MSC8157E Reference Manual

Broadband Wireless Access Six Core DSP With Security

MSC8157ERM
Rev 2, January 2012

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

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

North America:
Freescale Semiconductor, Inc.
Technical Sales and Commercial Support
Periferico Sur #8110
Col. El Mante
Tlaquepaque
Jalisco 45609
Mexico
1-800-521-6274 (US/Canada)
001 800 514 3392 (Mexico)
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

MSC8157ERM
Rev. 2
1/2012

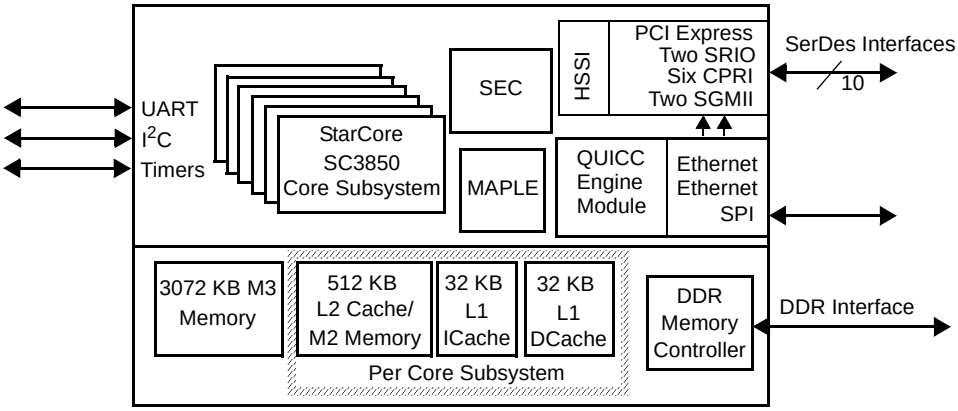
Contents

16 Serial RapidIO Controller and Enhanced Message Complex

17 PCI Express Controller

About This Book

The MSC8157E 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 support the 3G-LTE (FDD and TDD), HSPA+, LTE-Advanced, and WiMAX markets. 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 MSC8157E 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 3072 KB of shared M3 memory, one DDR-SDRAM controller to access up to 2 GB of DDR3/3L external memory, and a 32-channel direct memory access (DMA) controller optimized for DDR-SDRAM.

MAPLE-B2

The second-generation multi-accelerator platform engine (MAPLE-B2) provides Turbo or Viterbi decoding, Turbo encoding and rate matching, MIMO MMSE, IRC and ML equalization schemes, matrix operations, CRC insertion and check, DFT/iDFT and FFT/iFFT calculations, and Chip Rate acceleration.

Communications Processors and Interfaces

Includes a PCI Express interface, two serial RapidIO interfaces, six CPRI lanes, 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.

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

Before Using This Manual—Important Note

This manual describes the structure and function of the MSC8157E 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 MSC8157E. 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 MSC8157E 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 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	—	SWHR	RM	JPO	JH	—					RIO2	RIO1	RS	—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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 MSC8157E devices.
- **Chapter 6, *Boot Program***. Describes the bootloader program that loads and executes source code to initialize the MSC8157E 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 MSC8157E 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 MSC8157E modules; includes cross references to all registers discussed.
- **Chapter 10, *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 MSC8157E 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 and Enhanced Message Complex***. Describes the how the serial RapidIO interfaces and eMSG complex 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, *Common Public Radio Interface (CPRI) Complex***. Describes how the CPRI complex works and how to program it.
- **Chapter 19, *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 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 four timer modules, 16 identical 32-bit general purpose timers residing in two timer modules, and the 8 system watchdog timers and their sets 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 MSC8157E 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. 1149.6 documentation. The discussion covers the items that the standard requires to be defined and provides additional information specific to the MSC8157E 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 2 (MAPLE-B2)***. Describes the architecture, function, and register and memory structures used by the second-generation multi-accelerator platform engine (MAPLE-B2) for Channel Decoding/Encoding, Fourier Transforms, UMTS chip rate processing, OFDMA and SC-FDMA equalization and CRC algorithms. The MAPLE-B2 includes a quad-RISC programmable control processor, a second generation Programmable-System-Interface (PSIF2) that is a programmable controller with DMA capabilities, and ten accelerators.
- **Chapter 27, *Security Engine (SEC)***. Describes the architecture, function, and register and memory structures used for security algorithm processing.

Other MSC8157E Documentation

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

- *MSC8157E Data Sheet (*MSC8157E*)*. Details the signals, AC/DC characteristics, clock signal characteristics, package and pinout, and electrical design considerations of the MSC8157E 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.
- *MSC8157/8 Design Checklist* (AN4110). Identifies resources and provides guidance for developing applications using the MSC8157/8 DSP devices. It includes a check list for design phases of projects that incorporate the MSC8157/8 DSPs.
- *Differences Between the MSC8156 and the MSC8157 DSPs* (EB720). Indicates functional differences between the earlier generation MSC8156 and the MSC8157 DSPs.
- *Differences Between the MSC8157 and the MSC8158 DSPs* (EB723). Indicates functional differences between the devices in the MSC8157 DSP family.
- *Other documents.* Application Notes and Engineering Bulletins that cover various board layout and programming topics related to the StarCore DSP core and the MSC8157E 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	Oct 2011	Initial public release
1	Dec 2011	• **Chapter 5, *Reset***. – Removed the row for mode 22 in **Table 5-11**. • **Chapter 10, *SC3850 DSP Subsystem*** – Added new **Section 10.12**. • **Chapter 12, *DDR SDRAM Memory Controller*** – Added RC10 modification procedure after **Figure 12-11**. • **Chapter 15, *High Speed Serial Interface (HSSI) Subsystem*** – Added note to **Section 15.4**. – Added note after **Table 15-57** • **Chapter 16, *Serial RapidIO Controller and Enhanced Message Complex*** – Added note after the second bullet in **Section 16.1.4**. – Added **Section 16.1.7** and subsections. – Added **Section 16.1.8**. – Added note to **Section 16.3.1** about PDU segmentation. – Updated **Section 16.3.4**. – Updated **Section 16.3.8.2**. – Added note to **Section 16.3.8.3**. – Updated PWO and OPE rows in **Table 16-72**. – Added note after **Table 16-116**. – Updated SIZE row in **Table 16-134**, **Table 16-136**, and **Table 16-153**. – Added **Section 16.5**. • **Chapter 17, *PCI Express Controller*** – Added **Section 17.1.1.1** and **Section 17.1.1.2**. – Added note to **Section 17.1.2**. – Added information after **Table 17-14**. – Added **Section 17.4.8.1** and **Section 17.4.8.2**. – Added information after **Table 17-28**. – Added note after **Table 17-39**. – Added note after **Table 17-82**. – Added note after **Table 17-103**. – Added note after **Table 17-125**. • **Chapter 18, *Common Public Radio Interface (CPRI) Complex*** – Updated scrambling feature in **Section 18.1**. – Updated **Section 18.3.1.1**. – Updated **Section 18.3.1.3**. – Updated **Section 18.3.1.5.3**. – Updated **Section 18.3.3.6**. – Updated **Section 18.3.3.10** – Updated **Section 18.3.6.2**. – Updated **Section 18.4.1.13**. – Updated **Section 18.4.1.14**. – Updated RETHBS row in **Table 18-65** – Updated **Section 18.4.1.25**. – Updated ABORT row in **Table 18-127**. – Updated **Section 18.4.1.35**. – Updated **Section 18.4.4.7**. – Updated **Section 18.4.4.11**. • **Chapter 19, *QUICC Engine Subsystem*** – Added **Section 19.6.1** and subsections. – Added **Section 19.7.5** and subsections. • **Chapter 21, *Timers*** – Removed all references to special CPRI support. – Updated **Section 21.1.5** to add detailed procedure. – Added note to **Section 21.1.5.1**. • **Chapter 26, *Multi Accelerator Platform Engine, Baseband 2 (MAPLE-B2)*** – Updated **Section 26.4.3.5.1**, • **Chapter 27, *Security Engine (SEC)*** – Updated **Section 27.6.8** and **Section 27.7.4.4**.

Revision	Date	Change Description
2	Jan 2012	• **Chapter 5, *Reset*** – Updated rows for bit 17 and bit 16 in **Table 5-5**, **Table 5-7**, and **Table 5-9**. • **Chapter 8, *General Configuration Registers*** – Updated SerDes PLL designators in the descriptions for bits 19 and 18 in **Table 8-7**. • **Chapter 15, *High Speed Serial Interface (HSSI) Subsystem*** – Add EATTR settings in **Table 15-29**. – Updated the description of the LA[2–1] row in **Table 15-37**. – Split old **Section 15.10.52** into two: **Section 15.10.52** and **Section 15.10.53** to describe SRDB1RSTCTL and SRDB2RSTCTL individually. – Updated FRATE_SEL settings description in **Table 15-62**. • **Chapter 21, *Timers*** – Added an updated version of **Section 21.1.8**, *Special CPRI Support* which had been removed in Rev. 1.

1 Overview

The MSC8157E 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 support the rapidly changing and expanding broadband wireless markets, with support for 3G-LTE (FDD and TDD), HSPA+, LTE-Advanced, and WiMAX processing. The MSC8157E is carefully optimized for minimal cost, power, and area per channel. The highly flexible, fully-programmable and powerful MSC8157E broadband wireless access DSP offers tremendous processing power while maintaining a competitive price and high performance.

The highly integrated MSC8157E DSP device includes the following:

- Six StarCore SC3850 DSP subsystems each running at up to 1 GHz with an architecture optimized for wireless applications.
- One high-speed industry-standard DDR3 memory interface.
- Multi-Accelerator Platform Engine for Baseband 2 (MAPLE-B2) supports hardware acceleration for Turbo or Viterbi decoding, Turbo encoding and rate matching, MIMO MMSE, IRC and ML equalization schemes, matrix operations, CRC insertion and check, DFT/iDFT and FFT/iFFT calculations, and Chip Rate acceleration.
- High-Speed Serial Interface (HSSI) subsystem (10-lane) that supports
  - Two serial RapidIO interfaces
  - Two Gigabit serial Ethernet interfaces
  - Six CPRI channels
  - One PCI-Express controller
- QUICC Engine dual 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
- 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 MSC8157E 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 2 (MAPLE-B2)
— Two operating modes: 3G mode (3GLTE and UMTS standards) and WiMAX mode (**IEEE** 802.16e and 802.16m standards).
— Second Generation Programmable System Interface (PSIF2)
- 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.
- Four 128-bit master buses for data transfers from/to the system memory.
- Two 128-bit slave buses for the following purposes:
  — General purpose connection to CLASS, allowing any host or peripheral to access the MAPLE-B2 internal memories for placing job-descriptors in the PSIF2 internal memories or read/write data from/into PEs internal memories. This port is a read/write port and is referred to as MBus Slave0 port.
  — .Direct connection to CPRI for direct data transfer between the Antenna interface and MAPLE-B2. This port is a write only port and is referred to as MBus Slave1 port.
- 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 for WiMAX, UMTS and 3GLTE systems using an eTVPE module:
- Turbo decoder, rate-dematcher and HARQ combining acceleration

- Each decoder scalable with 1, 2, or 4 Radix 4 dual-recursion engines
- Binary and duo-binary codes
- Trellis termination and tail biting
- Various rate de-matching functions support for rate 1/3 code including sub-block de-interleaving and de-interlacing
- Max Log Map or Linear Log Map (MAX*)
- Non Linear Dynamics extrinsic factorization
- Programmable number of iterations
- Multiple stop conditions: CRC check, hard output compare and a-posteriori quality indication
- SIMD type of operation utilizing high level of hardware parallelism to provide high throughput and low latency channel decoding capabilities

— Turbo encoding for WiMAX OFDMA, 3GLTE DL/UL-SCH and UMTS systems using DEPE module:
- Information bits encoding and rate matching up to 1.8 Gbps for 3GLTE and WiMAX
- Code block CRC attachment and filler bits insertion for 3GLTE
- Bit randomization for WiMAX
- Transport block CRC attachment in 3GLTE in case of single code block.
- Information bits encoding and rate matching up to 900 Mbps for UMTS
- HS-DSCH and E-DCH (TDD and FDD) Turbo encoding and rate matching.
- CRC attachment for Transport Block $\leq$ 5090.
- Optional bit scrambling for Transport Block $\leq$ 5090 (FDD only).
- Transport Block generation assist in system memory using bit write granularity
- Transport Block segmentation assist using bit read granularity.

— Viterbi decoding for various technologies using eTVPE[1] module:
- Supports K =5,7,9
- Fully programmable polynomials
- Various rates/puncturing cases

— DFT/iDFT and FFT/iFFT processing using three eFTPE modules:
- Variable length FFT/iFFT processing of 128, 256, 512, 1024, 1536 and 2048 points.
- Variable length DFT/iDFT processing of the form, up to 1200 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 and 1200 points

1.Viterbi and Turbo decoding share memories and share throughput; 100% throughput of Viterbi and 100% throughput of Turbo cannot be achieved simultaneously.

- Variable input data size, supporting 16 bit {8I,8Q} or 32 bit {16I,16Q}
- Programmable guard band insertion and removal for iFFT and FFT processing.

– Programmable Cyclic Prefix insertion and removal.

- Pre-Multiplication (“array multiplication”) support by programmable complex values vector.
- Post-Multiplication support of all samples by programmable complex values vector.
- Phase rotation of input samples.
- Input data complex conjugate and re-ordering (1 to N samples reversed to N to 1)
- Zero padding of input data
- UMTS scrambled pilot samples generation in frequency domain
- Programmable scaling method, supporting one of the following methods:
- Automatic adaptive scaling using overflow detection between transform stages, and optional programmable overall scaling factor for the output data
- User defined scaling between transform stages
- Automatic or programmable input scale up for input data with small values, to increase calculation precision.

— CRC check and insertion using CRCPE:

- CRC check & report for UL processing
- CRC insertion for DL processing
- Up to 10 Gbps, supporting the following polynomials:
  - CRC24 with polynomial $D^{24} + D^{23} + D^6 + D^5 + D + 1$
  - CRC24 with polynomial $D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1$
  - CRC16 with polynomial $D^{16} + D^{12} + D^5 + 1$
  - CRC16 with polynomial $D^{16} + D^{15} + D^2 + 1$
  - CRC32 with polynomial $D^{32} + D^{26} + D^{23} + D^{22} + D^{16} + D^{12} + D^{11} + D^{10} + D^8 + D^7 + D^5 + D^4 + D^2 + D + 1$
  - CRC18 with polynomial $D^{18} + D^{17} + D^{14} + D^{13} + D^{11} + D^{10} + D^8 + D^7 + D^6 + D^3 + D^2 + 1$
  - CRC12 with polynomial $D^{12} + D^{11} + D^{10} + D^8 + D^5 + D^4 + 1$
  - CRC6 with polynomial $D^6 + D^5 + D^3 + D^2 + D + 1$

— MMSE (Minimum Mean Square Error) / ZF (Zero Forcing)/ IRC (Interference Rejection Combining) Equalization

- Estimation of signal by MMSE/ZF/IRC equalizer using following inputs:

- Observations, channel estimation, noise and transmitted layers covariance matrices and previously detected layer (for cancellation)
- Inputs precision 32 bit {16I,16Q} with block floating point 8 bit scaling factor.
- Supporting various Rx Antenna (1,2,4,8) and MIMO layers (1,2,3,4) configurations
- Optional Channel Estimation interpolation with user defined weights.
- Embedded support for advanced receivers with iterative interference cancellation schemes with internal single layer cancellation, rank reduction, layer discarding and MIMO layers covariance matrix.
- Optional support for full noise & interference covariance matrix, with variable granularity, up to different matrix per sub-carrier (RE).
- Flexible output format supporting 32 bit {16I,16Q} with optional block floating point representation using 8 bit scaling factor per element or per group of elements.

— Maximum Likelihood estimation implemented with QRD-M tree search

- Estimation of signal by QRD-M equalizer using following inputs:
- observations, channel estimation, noise variance.
- Supporting various Rx Antenna (1,2,4,8[1]) and MIMO layers (1,2,3,4) configurations.
- Optional Channel Estimation interpolation with user defined weights.
- Configurable search width, with M up to 64[2], providing performance-latency trade-off.
- LLR output generation.

— Matrix Inversion

- Support for up to 4×4 matrix inversion.
- Programming model optimized for batch job of multiple matrix inversions.
- High precision floating-point arithmetic
- Input samples are received in 32 bit {16I,16Q} block floating point (BLF) with 8-bit scale factor.
- Internal calculations performed using custom floating-point (FLP) format: 40 bit {20I,20Q} mantissa and 8-bit exponent.

— Downlink Chip Rate Processing:

- Capacity of up to 512 Physical Channels including MIMO, STTD, TSTD and Closed Loop Mode 1 operation per channel.
- Input data precision of 16 bit {8I,8Q} complex symbols.

1. In case of 8 Rx antenna system, QR decomposition is executed by DSP cores externally to MAPLE-B2 and QR matrices are input to MLD equalization and detection.
2. M=64 supported for up to 2 layers, M=32 for higher number of layers

- Spreading with SF 4,8,16,32,64,128, and 256 using internally generated channelization codes.
- Scrambling using internally generated, up to 32 independent codes, including support for compressed mode codes.
- Supporting SF 1 special channels with spreading, scrambling bypass
- Programmable complex gains per Physical Channel, with differentiation between data and control information, supporting various slot formats.
- Physical channels combining and flexible assignment to up to 16 virtual antenna's.
- Optional Beam Forming operation on combined Physical Channels.
- Output data precision of 32 bit {16I,16Q} complex chips.

— Uplink Batch Processing - for data and control channels with variable spreading factors

- Capacity of up to 384 Physical Channels with up to 2144 total fingers from up to 24 antenna streams with max 512 chips delay spread.
- Optional pre-de-spreading support with up to 80 Physical Channels with SF 4
- Input data precision of 16 bit {8I,8Q} complex chips.
- Optional Internal interpolation of x2 oversampled input stream, up to x16 resolution using programmable 8 tap polyphase filter.
- Despreading with SF 2,4,8,16,32,64,128,256 using internally generated channelization codes.
- Descrambling by short or long codes, with up to 384 different, internally generated, scrambling codes.
- Optional fingers combining using programmable weights.
- Optional frequency correction functionality using programmable correction factor.
- Multiple output formats:
  - 16 bit fixed point or custom 22 bit (16-bit mantissa and 6-bit exponent) floating point formats for fingers combining option.
  - complex 32 bit {16I,16Q} for fingers combining bypass option.

— Uplink Fast Processing - for (E)DPCCH processing

- Capacity of up to 400 Physical channels with up to 3200 total fingers from up to 24 antenna streams.
- Input data precision of 16 bit {8I,8Q} complex chips.
- Optional Internal interpolation of x2 oversampled input stream, up to x16 resolution using programmable 8 tap polyphase filter.
- Descrambling by short or long codes with internally generated scrambling codes.
- Despreading using SF 256 for DPCCH and E-DPCCH channels.
- Optional correlation with pilot sequence.

- Programmable slot format and early/on-time/late processing for various fields of control channels.
- Output 16 bit I and 16 bit Q.
- Internal commands FIFO for flexible updates of fingers and channels association.
- Latency of 68 chips including processing and write back of results to system memory

— PN code generator
- Short or Long codes generation based on programmable init values
- Generates scrambling code or scrambling code multiplied by Hadamard code with programmable spreading factor and OVSF.
- Two output formats:
- 16 bit {8I,8Q} format, with throughput of up to 4 Gsamples/s
- 2 bit {1I, 1Q} format, with throughput of up to 32 Gsamples/s

— Easily initialized and configured with minimal intervention:
- Software-friendly buffer descriptor handshake mechanism and task assignment
- Externally accessible memories and registers for debug purposes
- Internal, high throughput, DMA capabilities to fetch the input data and output the results to system memory
- Internal memory used for all module processing

— Multi-core support: Multiple configurable descriptor rings with support for high and low priority tasks
— System notification can generate RapidIO doorbells or interrupts on task completion
— Programmable customization including processing management and scheduling: Second Generation of Programmable System Interface (PSIF2)
— 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 controller, MAPLE-B2, 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.
  - 3072 Kbyte shared triple-bank, triple-port M3 memory. Power supply of two banks (2048 Kbyte) can be turned off to reduce power dissipation.
  - 96 Kbyte boot ROM accessible from the cores.
- Clocks
  - Four input clocks:
    - Global input clock.
    - Two differential input clocks (one per each SerDes PLL).
    - Optional DDR clock
  - Six PLLs:
    - Three system PLLs
    - Two SerDes PLLs
    - DDR Controller PLL
  - Clock ratios selected during reset via reset configuration pins.
  - Clock modes user-configurable after reset.
- One DDR Controller supporting:
  - Up to 667 MHz clock rate (1333 MHz data rate).
  - Supports DDR3 devices
  - Programmable timing
  - Support for a 64-bit data interface (72 bits including ECC), up to 1333 MHz data rate
  - Support for a 32-bit data interface (40 bits including ECC), up to 1333 MHz data rate

- Full ECC support for single-bit error correction and multi-bit error detection up to the maximum specified data rates
- Two banks of memory via two chip selects. Each chip select supports up to 2 Gbytes, but memory total cannot exceed 2 Gbytes.
- DRAM chip configurations from 64 Mbits to 4 Gbits with x8/x16 data ports
- Support burst lengths of 4 (burst chop) and On the Fly
- Sleep mode support for self-refresh SDRAM
- On-die termination support
- Supports auto refreshing
- Support for SODIMMs

■ 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)

- Ten multiplexed SerDes lanes
- Serial RapidIO Subsystem
  - Two Serial RapidIO ports supporting x1/x2/x4 operation up to 5 Gbaud with a RapidIO enhanced messaging unit (eMSG) and two RapidIO DMA units.
  - Each x1/x2/x4 Serial RapidIO endpoint operates at 1.25/2.5/3.125/5 Gbaud and complies with the following parts of Specification 2.1 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 1x LP-serial specification)
- Part VIII (error management extension specification)

- Each Serial RapidIO port supports read, write, messages, doorbells, data streaming 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 Enhanced Messaging Unit supports:
  - RapidIO Interconnect Specification 1.3, Part 2: Message Passing Logical Specification.
  - RapidIO Interconnect Specification 1.3, Part 10: Data Streaming Logical Specification.
  - RapidIO Interconnect Specification 2.1, Part 10: Stream Management Flow Control. Basic stream management flow control (XON/XOFF) using extended header message format.
  - 64 outbound queues allowing multi-core environment.
  - 16 concurrent inbound reassembly operations. One additional reserved reassembly for inbound unit 0 to carry session management protocol.
  - Multi unicast.
- 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

- Common Public Radio Interface (CPRI) Controller

- Supports v4.1 of the CPRI standard
- Up to 6 lanes
- Supports 1.2288 Gbaud, 2.4576 Gbaud, 3.072 Gbaud, 4.9152 Gbaud and 6.144 Gbaud
- Supports scrambling
- Daisy-chain capability that allows cascading devices according to pre-determined user configuration. Chaining can be done for each lane using the CPRI controller. Chaining lanes does not require core intervention or internal device bandwidth overhead.
- Input power can be disabled for system power reduction if CPRI is not required.

— PCI-Express Controller

- Complies with the *PCI Express Base Specification, Revision 2.0*
- Supports root complex (RC) and endpoint (EP) configurations
- 32- and 64-bit address support
- x4, x2, and x1 link support
- Supports 2.5 Gbaud, 5.0 Gbaud.
- 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

■ 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.

- 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 four 16-bit timers; cascadable timers; count up/down; programmable count modulo; count once or repeatedly; counters are preloadable; compare registers can be preloaded; counters can share available inputs; separate prescaler for each counter; each counter has capture and compare capability; any of the following clock sources: system clock or external clock input
  - Two TIMER_32B modules, each with four 32-bit timers; cascadable timers; count up/down; programmable count modulo; count once or repeatedly; counters are preloadable; compare registers can be preloaded; counters can share available inputs; separate prescaler for each counter; each counter has capture and compare capability; any of the following clock sources: system clock 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. 1149.6.
- 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 MSC8157E device is manufactured using CMOS 45 nm SOI technology.
- Flip Chip-Plastic Ball Grid Array (FC-PBGA), 783-ball, 1 mm pitch, 29 mm × 29 mm

1.2 Block Diagram

A block diagram of the MSC8157E is shown in **Figure 1-1**.

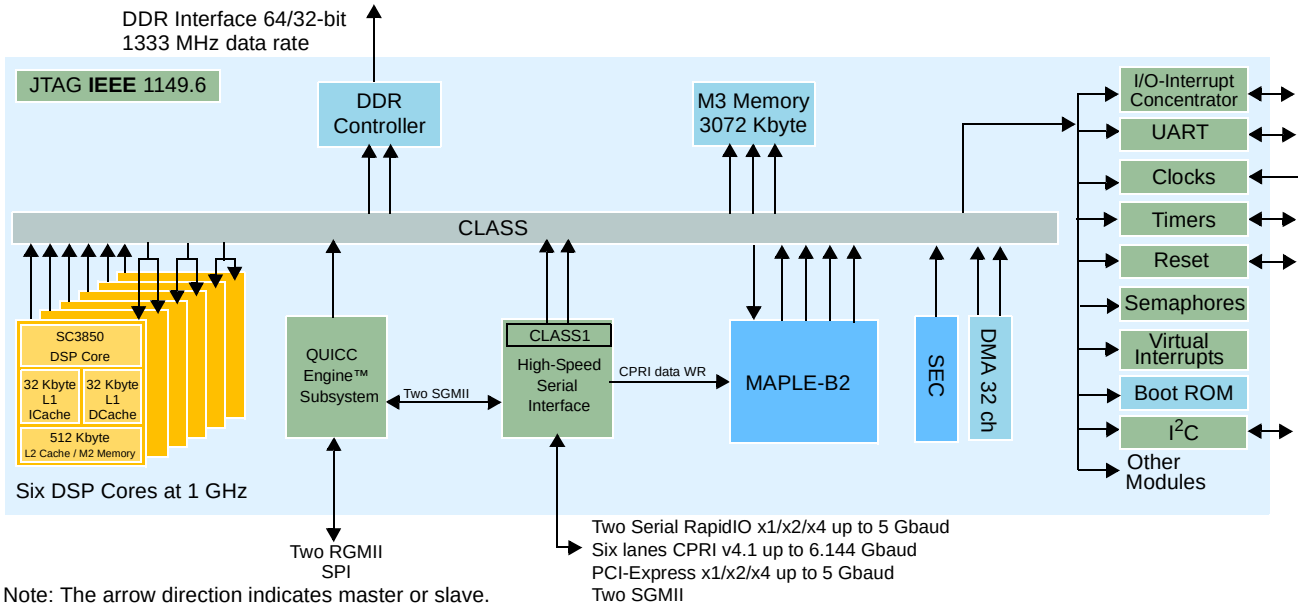


Figure 1-1. MSC8157E Block Diagram

1.3 Architecture

The MSC8157E architecture is carefully optimized to achieve the maximum channel density for a given device area, power, and cost. Also, the MSC8157E 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.

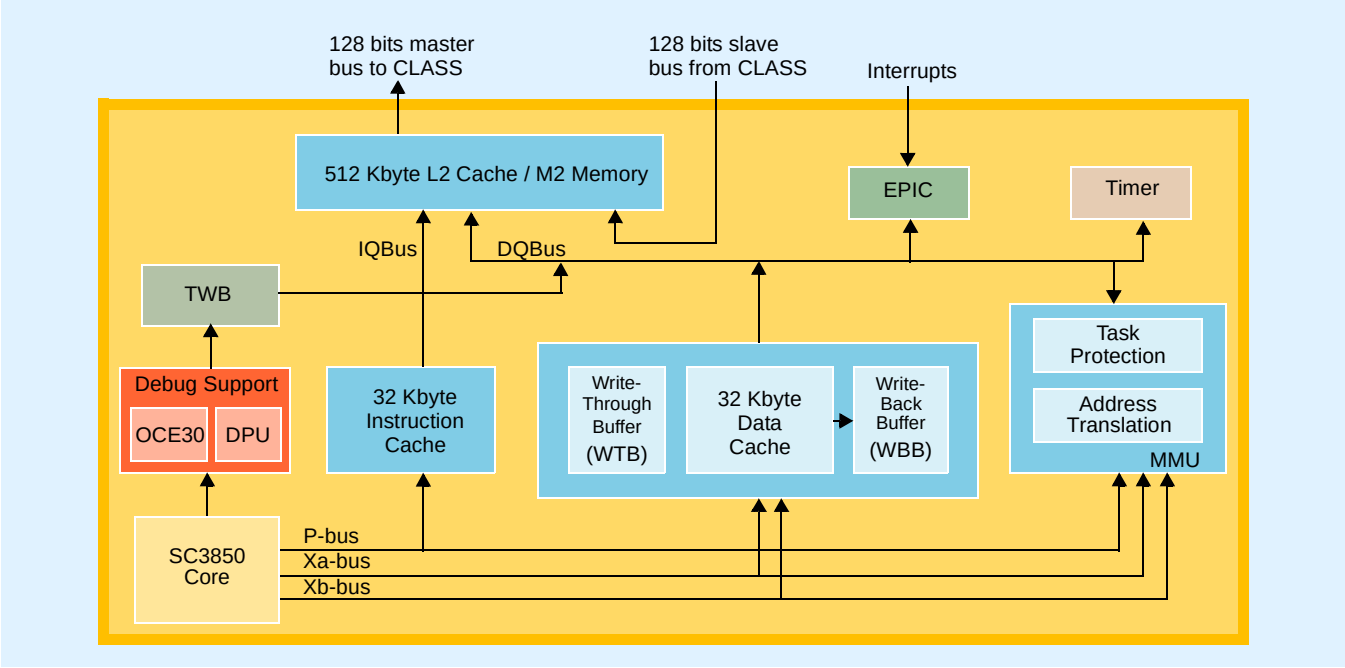


Figure 1-2. StarCore SC3850 DSP Subsystem Block Diagram

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 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.

1.4.2 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.

1.4.3 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.

1.4.4 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.

1.4.5 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.

1.4.6 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

1.4.7 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.

1.4.8 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-B2

Decoding/Encoding, Fourier Transforms, UMTS chip rate processing, OFDMA and SC-FDMA equalization and CRC algorithms. The MAPLE-B2 consists of second generation Programmable-System-Interface (PSIF2), a programmable controller with DMA capabilities. Eight accelerators are attached to it:

- eTVPE (Enhanced Turbo/Viterbi Processing-Element). Accelerates Turbo Decoding, Rate-De-Matching and HARQ combining.
- Three eFTPE (enhanced FFT/DFT Processing-Element). Accelerates various sizes of Fourier transforms and various pre/post transform processing.
- DEPE (Turbo Encoder Processing-Element). Accelerates Turbo Encoding and Rate Matching.
- EQPE (Equalization Processing-Element). Accelerates MMSE and MLD types of equalization algorithms and Matrix Inversion.
- CRPE (Chip Rate Processing-Element). Accelerates Downlink and Uplink UMTS chip rate processing
- CRCPE (CRC Processing-Element). Accelerates CRC attachment or check.

In addition to the eight accelerators, MAPLE-B2 provides virtual processing elements which enable chained functionality of multiple processing elements:

- Convolution or Filtering functionality utilized by combination of eFTPE and EQPE.
- Turbo Decoding and Re-encoding including rate matching using combination of eTVPE and DEPE.
- MMSE equalization combined with DFT and/or IDFT functions utilizing eFTPE and EQPE.

1.6 Security Engine (SEC)

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) 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 Chip Level Arbitration and Switching System (CLASS) is the central internal interconnect system for the MSC8157E 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 667 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 fifteen CLASS initiators are:

- Six SC3850 core subsystems (initiator ports 0–5)
- Four MAPLE bridges (initiator ports 6–7 and ports 13–14)
- Two HSSI ports shared by two Serial RapidIO controllers, the PCI Express controller, six CPRI controllers, and two SGMIIs (initiator ports 8 and 12)
- Peripherals bridge shared by the SEC, SPI, RGMII, SGMII, and JTAG interface (initiator port 9)
- Two DMA ports (initiator ports 10–11)

The ten CLASS targets are:

- Configuration Control and Status Registers (CCSR) (target port 0)
- Two DDR ports (target ports 1 for writes and 2 for reads)
- MAPLE module (target port 3)
- Three core subsystem bridges (two core subsystems per bridge) (target ports 4–6)
- Three M3 memory ports (target ports 7–9)

1.8 M3 Memory

The 3072 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 memory is divided into three 1 MB blocks. One block is always on when the chip is powered up; the other two blocks can be powered down. The M3 memory has three 128-bit wide ports and runs at 667 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 the upper 2048 MB to reduce power consumption.

1.9 Clocks

The MSC8157E device has four input clocks:

- A global input clock.
- Two differential input clocks (one per SerDes PLL).
- MCLKIN (optional) input for the DDR memory

The MSC8157E device includes six PLLs:

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

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 Controller (DDRC)

The DDR SDRAM interface is useful when the channel storage size is relatively big and also when more channels are required to supplement the internal memory. When the MSC8157E 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 applications. However, it is not large enough for memory intensive application, especially when high channel densities are required. For such applications, the MSC8157E can interface with JEDEC-compliant 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 has a separate strobe per byte. Two logical banks (chip select) are supported, each with logical programmable bank start and end addresses. 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 10-port serial communications subsystem that supports multiplexing of the following serial interfaces:

- Two x1/x2/x4 Serial RapidIO ports
- One x1/x2/x4 PCI Express port
- Two SGMII ports
- Six x1 CPRI ports

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

- One CLASS1 non-blocking fabric to interface between the HSSI and the system CLASS module.
- One 8-port OCN fabric with two dedicated DMA controllers and two OCN to MBus bridges to provide high-speed memory access with no core intervention for the two Serial RapidIO port controllers and the PCI Express controller. The bridges connect to the CLASS 1 module.
- Two Serial RapidIO port controllers (port 1 and 0) with enhanced messaging unit (eMSG). The RapidIO complex includes an AXI bridge to MBus to connect to the CLASS1 fabric and an eMSG MBus expander and normalizer to connect to the two Serial RapidIO port controllers. The RapidIO complex and eMSG details are provided in **Chapter 16**, *Serial RapidIO Controller and Enhanced Message Complex*.
- One PCI Express controller with a bridge to the OCN fabric. The PCI Express controller details are provided in **Chapter 17**, *PCI Express Controller*
- Six CPRI controllers, described in detail in **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*.
- One Protocol Converter module to link the two Serial RapidIO ports, one PCI Express port, six CPRI ports and the two SGMII ports to the SerDes PHY.
- One 10-channel SerDes PHY that multiplexes the RapidIO, PCI Express, CPRI and SGMII signals for external connection.

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.

1.12.1 CLASS1

Based on the same design as the system CLASS fabric, CLASS1 uses a fully pipelined low latency design and is the interconnect between the HSSI complex and the system CLASS. Like the CLASS, CLASS1 is a non-blocking, full-fabric interconnect that allows any initiator to access any target in parallel with another initiator-target couple. CLASS1 uses per-target prioritized round-robin arbitration, highly optimized to the target characteristics. CLASS1 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 Six CLASS1 initiators are:

- OCN-to-MBus bridge 0 (initiator port 0)
- OCN-to-MBus bridge 1 (initiator port 3)
- AXI-to-MBus bridge 0 (initiator port 5)
- AXI-to-MBus bridge 1 (initiator port 2)
- CPRI MBus port 0 (initiator port 4)
- CPRI MBus port 1 (initiator port 1)

The two CLASS1 targets are:

- HSSI port 0 (target port 0). Connects to CLASS initiator port 8.
- HSSI port 1 (target port 1). Connects to CLASS initiator port 12.

1.12.2 OCN Fabric

The On-Chip Network (OCN) fabric is a non-blocking high speed interconnect used for embedded system devices. The MSC8155E DSP HSSI uses an 8-port OCN to connect between the Serial RapidIO Controllers, PCI Express Controller, the two OCN-to-MBus bridges (O2M[0–1]) that connect to the CLASS1 module, and the two supporting dedicated DMA controllers. The OCN requires no programming and provides a seamless interface for the HSSI.

1.12.3 OCN-to-MBus (O2M) Bridges

The O2M bridges provide an interface between the OCN fabric and the CLASS1 module that connects to the system CLASS. The bridges support all mandatory MBus and OCN interface protocol procedures. The bridges provide initiator interfaces to access the target CLASS ports, formats OCN packets, negotiates with the OCN arbiter, and transmits/receives associated transactions.

1.12.4 DMA Controllers

The MSC8157E includes two dedicated DMA controllers that transfer blocks of data between the serial RapidIO controller/PEX Controller and the local address space independent from the DSP cores.

1.12.5 Serial RapidIO Complex

The Serial RapidIO complex includes two ports (SRIO0 and SRIO1) that perform the serial data transfers between the SerDes PHY and the DSP system, an enhanced messaging unit (eMSG),

and the interface bridges that connect between SRIO0 and SRIO1, the eMSG module, and the CLASS1 module. Configuration and functional details for this complex are provided in **Chapter 16**, *Serial RapidIO Controller and Enhanced Message Complex*. Transfers through the SRIO modules can be tracked by the performance monitor. See **Section 25.3**, *Performance Monitor*, on page 25-27 for details.

1.12.6 PCI Express Controller

HSSI includes one PCI Express controller that supports 1x/2x/4x link configuration. It is interfaced to the SerDes Phy through Protocol converter module and connects to the OCN fabric through PEX to OCN bridge. The functional and configuration details are provided in **Chapter 17**, *PCI Express Controller***.**

1.12.7 Protocol Converter

The protocol converter interfaces the Serial RapidIO, PCI Express, CPRI and SGMII signals to the SerDes PHY. It is programmed internally by selections make in the General Configuration Registers and the power-on reset configuration (see **Section 15.10**, *SerDes PHY Interfaces*, on page 15-30).

1.12.8 SerDes PHY Interfaces

The HSSI includes one 10-port SerDes interface that multiplexes the two Serial RapidIO ports, the PCI Express port, six CPRI lanes, and the two SGMII ports from the QUICC Engine subsystem. Multiplexing configuration is determined by the reset configuration word settings (see **Chapter 5**, *Reset* for details).

1.13 QUICC Engine Subsystem

The MSC8157E 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.

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 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.

1.15 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.

1.16 Timers

The MSC8157E device contains 16 identical 16-bit timers divided into four groups and 8 identical 32-bit timers divided into two groups. Each of the 16-bit groups (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. Each of the 32-bit groups (TMR_32b) contains four identical 32-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 eight general purpose 32-bit timers divided into two groups. each with the same basic registers supported by the 16-bit timers. The MSC8157E device also includes 8 software watchdog timers. Each of the software watchdog timers can be used by any of the cores within MSC8157E as well as by an external host.

1.17 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.

1.18 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.

1.19 I^2C Interface

The inter-integrated circuit (I^2C) controller enables the MSC8157E 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.

1.20 GPIOs

The MSC8157E 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). 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. GPIO ports do not have internal pull-up resistors. The dedicated MSC8157E peripheral functions multiplexed with the GPIO ports are grouped to maximize the usefulness of the ports in the greatest number of MSC8157E applications.

1.21 Boot Options

The boot program in the internal boot ROM initializes the MSC8157E after it completes a reset sequence. The MSC8157E 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.

1.22 JTAG

The dedicated user-accessible test access port (TAP) is fully compatible with **IEEE** Std. 1149.6. The MSC8157E device supports circuit-board test strategies based on this standard. For details on the standard, refer to the standard documentation.

1.23 Developer Environment

Freescale supplies a complete set of DSP development tools for the MSC8157E 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 MSC8157E architecture.

1.23.1 Tools

The MSC8157E 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 MSC8157*. A complete system for developing and debugging real-time hardware and software.

1.23.2 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 1-1**.

Table 1-1. Application Software Modules

Application	Modules
Baseband	3G-LTE evolving kernels library.
	Optimized FFT kernels, MMSE, QR Decomposition and other matrix multiplication operations.
	WCDMA including symbol rate and chiprate library.
	WiMAX solution supporting Wave 1 features with future extension to Wave 2 and beyond.
Device Drivers and Example Code	MAPLE-B2 driver, DMA driver, Serial RapidIO driver, PCI-Express, CPRI, 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.

2 SC3850 Core Overview

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
- 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
- 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.

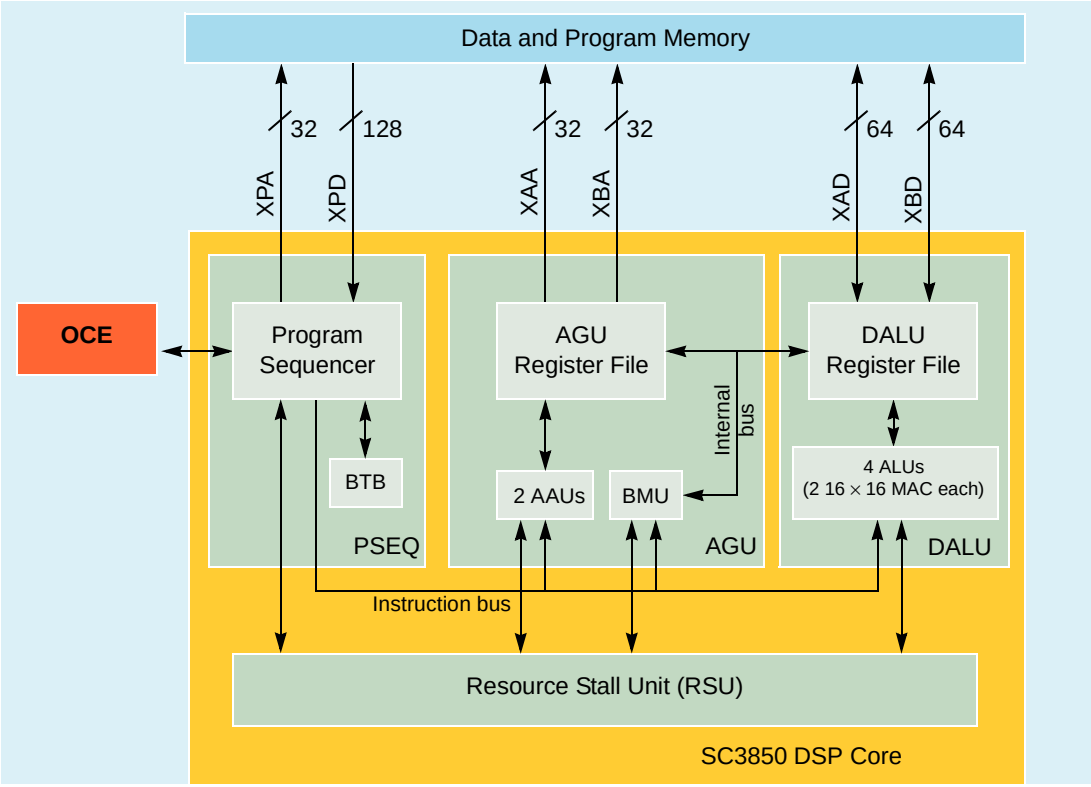


Figure 2-1. SC3850 DSP Core Block Diagram

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.

3 External Signals

The MSC8157E 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. MSC8157E Functional Signal Groupings

Functional Group	Detailed Description
Power and ground	**Table 3-2** on **page 3-4**
Clock	**Table 3-3** on **page 3-5**
Reset and Configuration	**Table 3-4** on **page 3-6**
DDR Memory Controller	**Table 3-5** on **page 3-11**
SerDes Multiplexers (Serial RapidIO controllers, PCI Express interface, CPRI, and SGMII signals)	**Table 3-6** on **page 3-12**
CPRI	**Table 3-8** on **page 3-13**
Ethernet	**Table 3-9** on **page 3-16**
Serial peripheral interface (SPI)	**Table 3-10** on **page 3-18**
GPIOs and maskable Interrupts	**Table 3-11** on **page 3-19**
Timers	**Table 3-12** on **page 3-24**
UART	**Table 3-13** on **page 3-26**
I^2C	**Table 3-14** on **page 3-27**
External DMA Interface	**Table 3-15** on **page 3-27**
$\overline{\text{NMI}}$/$\overline{\text{INT\_OUT}}$/$\overline{\text{NMI\_OUT}}$/$\overline{\text{CP\_TX\_INT}}$/$\overline{\text{CP\_RX\_INT}}$	**Table 3-16** on **page 3-28**
OCE module and JTAG Test Access Port	**Table 3-17** 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:

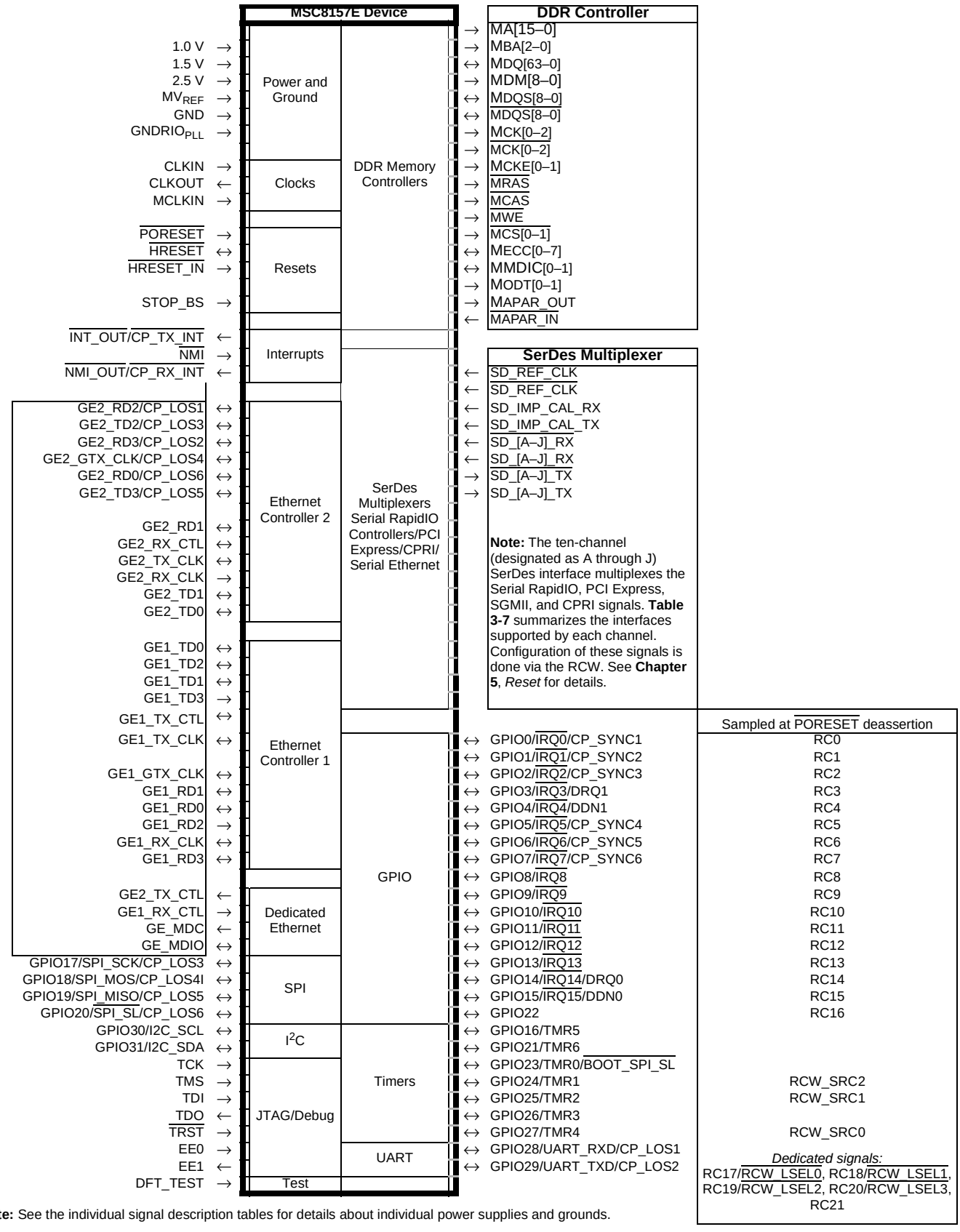
- Ethernet configurations (RGMII/SGMII) are defined by the Ethernet register settings (see the *QUICC Engine Block Reference Manual with Protocol Interworking* for details). Selection of RGMII or SGMII is configured by the QECR (see **Section 8.2.8**, *QUICC Engine Control Register (QECR)*, on page 8-17).
- SerDes Multiplexing. Serial RapidIO interfaces, PCI Express interface, CPRI lanes, and SGMII sharing on the 10-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 20**, *GPIO* for details).

The PCI Express, serial RapidIO interfaces, CPRI channels, and the Ethernet SGMIIs share the ten SerDes lanes. Access to the SerDes interface block 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 CPRI, SPI, timers, UART, and I^2C signals. The specific function is selected through configuration of the GPIO registers (see **Chapter 20**, *GPIO* for details).

Figure 3-1 summarizes the various MSC8157E external signal multiplexing options.



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

Figure 3-1. MSC8157E External Signals

3.1 Power Signals

Table 3-2. Power and Ground Inputs

Nominal Voltage	Signal Name	Symbol	Description
1.0 V	VDD	V_{DD}	**Main Power** A dedicated well-regulated power source for the device. This supply serves the 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 2 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, 1 Mbyte of M3 memory remains active.
	CRPEVDD	V_{DDCRPE}	**Power for MAPLE-B2 Chip Rate Processing Element** A dedicated well-regulated power source for the MAPLE-B2 chip rate processing element (CRPE). 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.
	CPRIVDD	V_{DDCPRI}	**Power for CPRI** A dedicated well-regulated power source for the CPRI subsystem (all controllers). 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.
	MAVDD	V_{DDPLLM}	**DDR PLL Power** A dedicated well-regulated power for the DDR Phase Lock Loop (PLL).
	PLL0_AVDD	V_{DDPLL0}	**System PLL 0 Power** A dedicated well-regulated power for the system Phase Lock Loop (PLL).
	PLL1_AVDD	V_{DDPLL1}	**System PLL 1 Power** A dedicated well-regulated power for the system Phase Lock Loop (PLL).
	PLL2_AVDD	V_{DDPLL2}	**System PLL 2 Power** A dedicated well-regulated power for the system Phase Lock Loop (PLL).
	SXCVDD	V_{DDSXC}	**SerDes Core Power** A dedicated well-regulated power for the SerDes core circuitry.
	SD_PLL1_AVDD	V_{DDPLL}	**SerDes PLL 1 Power** A dedicated well-regulated power for the first SerDes Phase Lock Loop (PLL).
	SD_PLL2_AVDD	V_{DDPLL}	**SerDes PLL2 Power** A dedicated well-regulated power for the second SerDes Phase Lock Loop (PLL).
1.5 V	GVDD	V_{DDDDR}	**SSTL15 IO Driver Power** A dedicated power source for the DDR controller DRAM interface buffers. Provide adequate external decoupling capacitors. The external decoupling capacitors recommendations are listed in the *MSC8157E Technical Data Sheet*.
	SXPVDD	V_{DDSXP}	**SerDes Pad Power** A dedicated well-regulated power for the SerDes pad circuitry.

Table 3-2. Power and Ground Inputs (Continued)

Nominal Voltage	Signal Name	Symbol	Description
GVDD × 0.5 V	MVREF	MV_{REF}	**SSTL Reference Power** A reference power level for the DDR controller 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 *MSC8157E 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 *MSC8157E Technical Data Sheet*.
0 V	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}.
	SD_PLL1_AGND	GND_{SDPLL1}	**SerDes PLL 1 Ground** Ground dedicated for SerDes PLL 1 use. The connection should be provided with an extremely low-impedance path to ground.
	SD_PLL2_AGND	GND_{SDPLL2}	**SerDes PLL 2 Ground** Ground dedicated for SerDes PLL 2 use. The connection should be provided with an extremely low-impedance path to ground.
	SXCVSS	GND_{SXC}	**SerDes Core Ground** A ground for the SerDes Core circuitry.
	SXPVSS	GND_{SXP}	**SerDes Pad Ground** A ground for the SerDes Pad circuitry.

Note: The external decoupling capacitors recommendations are listed in the *MSC8157E Technical Data Sheet*.

3.2 Clock Signals

Table 3-3. Clock Signals

Signal Name	Type	Signal Description
CLKIN	Input	**Clock In** Primary clock input to the MSC8157E PLLs.
CLKOUT	Output	**Clock Out** The bus clock output.
MCLKIN	Input	**DDR Clock In (optional)** Optional clock input to the DDR PLL. This clock can be used in part of the clock modes if required.

3.3 Reset and Configuration Signals

Table 3-4. Reset and Configuration Signals

Signal Name	Type	Signal Description
PORESET	Input	**Power-On Reset** When asserted, this line causes the MSC8157E to enter power-on reset state. Internally, this signal also resets the TAP and debugging modules. The power-on reset flow resets the MSC8157E device, configures various device attributes including its clock modes, and drives HRESET as an open-drain output.
HRESET	Input/ Output	**Hard Reset** When asserted as an input, this signal causes the MSC8157E 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 HRESET as open-drain output. This signal requires an external pull-up resistor. The signal is tri-stated after the hard reset flow is complete.
HRESET_IN	Input	**Hard Reset In** Asserted as an input. When asserted, this signal causes the MSC8157E 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. This signal requires an external pull-up resistor. The signal is tri-stated after the hard reset flow is complete. Unlike HRESET, it does not drive a reset output and therefore can be used to reset a single device in a multi-device system without affecting the other devices.
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 HRESET/HRESET_IN 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*).
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/HRESET_IN 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 20**, *GPIO*.
TMR4	Input/ Output	**Timer 4** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *GPIO*. For functional details, see **Chapter 21**, *Timers*.

Table 3-4. Reset and Configuration Signals (Continued)

Signal Name	Type	Signal Description
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/HRESET_IN 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 20**, *GPIO*.
TMR2	Input/ Output	**Timer 2** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *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/HRESET_IN 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 20**, *GPIO*.
TMR1	Input/ Output	**Timer 1** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *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 20**, *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 configuration details, see **Chapter 20**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.
CP_SYNC[1–3]	Input	**CPRI Controller 1–3 Synchronization Signal** Synchronization signals for CPRI controllers 1–3. Selected through GPIO configuration. For configuration details, see **Chapter 20**, *GPIO*. For functional details, see **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*.

Table 3-4. 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 20**, *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 20**, *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[5–7]	Input	**Reset Configuration Word Bit 5–7** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
GPIO[5–7]	Input/ Output	**General-Purpose Input Output 5–7** Three of the 32 GPIO pins used as GPIO or as a dedicated input or output. For details, see **Chapter 20**, *GPIO*.
IRQ[5–7]	Input	**Interrupt Request 5–7** External lines that can request a service routine via the internal interrupt controller. Selected through GPIO configuration. For configuration details, see **Chapter 20**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.
CP_SYNC[4–6]	Input	**CPRI Controller 4–6 Synchronization Signal** Synchronization signals for CPRI controllers 4–6. Selected through GPIO configuration. For configuration details, see **Chapter 20**, *GPIO*. For functional details, see **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*.

Table 3-4. Reset and Configuration Signals (Continued)

Signal Name	Type	Signal Description
RC[8–13]	Input	**Reset Configuration Word Bit 8–13** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
GPIO[8–13]	Input/ Output	**General-Purpose Input Output 8–13** Six of the 32 GPIO pins used as GPIO or as a dedicated input or output. For details, see **Chapter 20**, *GPIO*.
IRQ[8–13]	Input	**Interrupt Request 8–13** External lines that can request a service routine via the internal interrupt controller. Selected through GPIO configuration. For configuration details, see **Chapter 20**, *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 20**, *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*,
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 20**, *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 20**, *GPIO*.
TMR5	Input/ Output	**Timer 5** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *GPIO*. For timer functional details, see **Chapter 21**, *Timers*.

Table 3-4. Reset and Configuration Signals (Continued)

Signal Name	Type	Signal Description
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.
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/HRESET_IN is asserted. All other signal drivers connected to these inputs must be tri-stated while HRESET/HRESET_IN 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

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

Table 3-5. Memory Controller Signals

Signal Name	Type	Description
MA[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.
MBA[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.
MDQ[63–0]	Input/ Output	**Data Bus** The MSC8157E device drives the bus during write cycles and the external memory drives the bus during read cycles.
MDM[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.
MDQS[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.
$\overline{\text{MDQS[8–0]}}$	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.
MECC[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.
MCK[2–0]	Output	**DDR Clock Out** The DDR clock output. Each signal is part of a differential pair.
$\overline{\text{MCK[2–0]}}$	Output	**DDR Clock Out Inverted** The inverted DDR clock. Each signal is part of a differential pair.
MCKE[1–0]	Output	**Clock Enable** When asserted, this signal enables the DDR clock for the DDR DRAM.
$\overline{\text{MRAS}}$	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.
$\overline{\text{MCAS}}$	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.
$\overline{\text{MWE}}$	Output	**Write Enable** Connects to DDR DRAM $\overline{\text{WE}}$ input.

Table 3-5. Memory Controller Signals (Continued)

Signal Name	Type	Description
$\overline{\text{MCS[0–1]}}$	Output	**Chip Select 0–1** Enables specific memory devices or peripherals connected to the bus.
MMDIC[0–1]	Input/Output	**Driver Impedance Calibration** These lines are used for automatic calibration of the DDR I/O.
MODT[0–1]	Output	**On-Die Termination** Memory controller outputs for the ODT to the SDRAM. Each signal represents the corresponding chip select.
MAPAR_OUT	Output	**Parity Output** If enabled, drives the parity bit for the bus.
$\overline{\text{MAPAR\_IN}}$	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, CPRI, and SGMII Interfaces

Refer to **Chapter 5**, *Reset*; **Chapter 17**, *PCI Express Controller*; **Chapter 16**, *Serial RapidIO Controller*, **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*, and the *QUICC Engine Block Reference Manual with Protocol Interworking* for configuration information.

Table 3-6. SerDes Multiplexed Signals

Signal Name	Type	Description
SD_IMP_CAL_RX	Input	**SerDes Receiver Impedance Control Signal** Receiver impedance calibration control signal.
SD_IMP_CAL_TX	Input	**SerDes Transmitter Impedance Control Signal** Transmitter impedance calibration control signal.
SD_REF_CLK	Input	**SerDes Reference Clock** Reference clock signal. Each signal is part of a differential pair. For SGMII operation, this provides the input clock.
$\overline{\text{SD\_REF\_CLK}}$	Input	**SerDes Reference Clock Inverted** Inverted reference clock signal. Each signal is part of a differential pair. For SGMII operation, this provides the input clock.
$\overline{\text{SD\_[A–J]\_TX}}$	Output	**SerDes Lane A to J Transmit Data Inverted** Inverted serial data output. Each signal is part of a differential pair.
SD_[A–J]_TX	Output	**SerDes Lane A to J Transmit Data** Serial data output. Each signal is part of a differential pair.
$\overline{\text{SD\_[A–J]\_RX}}$	Input	**SerDes Lane A to J Receive Data Inverted** Inverted serial data output. Each signal is part of a differential pair.
SD_[A–J]_RX	Input	**SerDes Lane A to J Receive Data** Serial data output. Each signal is part of a differential pair.
Note: For proper definition of serial RapidIO modes (x1/x2/x4), PCI Express (x1/x2/x4), CPRI, and SGMII, configure the interfaces using the Reset Configuration Word settings. For details, see **Chapter 5**, *Reset*. **Table 3-7** lists the interfaces supported by each SerDes lane.		

Table 3-7. Summary of Multiplexed Signal Support by Channel

SerDes Lane	Multiplexed Signals					
	Serial RapidIO 1	Serial RapidIO 2	PCI Express	SGMII1	SGMII2	CPRI
A	—	0	0	—	—	—
B	—	0 or 1	1	—	X	—
C	0 or 1	0 or 2	2	X	—	—
D	0 or 1	1 or 3	3	—	X	—
E	0 or 2	1 or 2	—	X	—	6
F	1 or 3	0 or 3	—	—	X	5
G	0 or 2	1	—	X	—	4
H	1 or 3	0	—	—	X	3
I	0 or 2	—	0	X	—	2
J	1 or 3	—	1	—	X	1

Note: Interface support by channel is indicated by the following:
1. — indicates that the channel does not support the interface.
2. A number indicates which data lines the channel supports.
3. X indicates that the channel supports the designated SGMI connection.

3.6 CPRI Signals

Table 3-8. CPRI Signals

Signal Name	Type	Signal Description
RC[0–2]	Input	**Reset Configuration Word Bit 0–2** Sampled during the assertion of $\overline{\text{PORESET}}$ 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 20**, *GPIO*.
$\overline{\text{IRQ}}$[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 configuration details, see **Chapter 20**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.
CP_SYNC[1–3]	Input	**CPRI Controller 1–3 Synchronization Signal** Synchronization signals for CPRI controllers 1–3. Selected through GPIO configuration. For configuration details, see **Chapter 20**, *GPIO*. For functional details, see **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*.

Table 3-8. CPRI Signals (Continued)

Signal Name	Type	Signal Description
RC[5–7]	Input	**Reset Configuration Word Bit 5–7** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
GPIO[5–7]	Input/ Output	**General-Purpose Input Output 5–7** Three of the 32 GPIO pins used as GPIO or as a dedicated input or output. For details, see **Chapter 20**, *GPIO*.
IRQ[5–7]	Input	**Interrupt Request 5–7** External lines that can request a service routine via the internal interrupt controller. Selected through GPIO configuration. For configuration details, see **Chapter 20**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.
CP_SYNC[4–6]	Input	**CPRI Controller 4–6 Synchronization Signal** Synchronization signals for CPRI controllers 4–6. Selected through GPIO configuration. For configuration details, see **Chapter 20**, *GPIO*. For functional details, see **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*.
GE2_RD2	Input	**Ethernet 2 Receive Data 2** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
CP_LOS1	Input	**CPRI Optical Loss of Signal Indication for Lane 1** This is the default option. The signal can also be configured on GPIO28.
GPIO28	Input/ Output	**General-Purpose Input Output 28** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
UART_RXD	Input/ Output	**UART Receive Data** For details, see **Chapter 19**, *UART*.
CP_LOS1	Input	**CPRI Optical Loss of Signal Indication for Lane 1** This is an optional configuration. The signal can also be configured on GE2_RD2.
GE2_RD3	Input	**Ethernet 2 Receive Data 3** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
CP_LOS2	Input	**CPRI Optical Loss of Signal Indication for Lane 2** This is the default option. The signal can also be configured on GPIO29.
GPIO29	Input/ Output	**General-Purpose Input Output 29** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
UART_TXD	Output	**UART Transmit Data** For details, see **Chapter 19**, *UART*.
CP_LOS2	Input	**CPRI Optical Loss of Signal Indication for Lane 2** This is an optional configuration. The signal can also be configured on GE2_RD3.
GE1_TD2	Output	**Ethernet 1 Transmit Data 2** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
CP_LOS3	Input	**CPRI Optical Loss of Signal Indication for Lane 3** This is the default option. The signal can also be configured on GPIO17.

Table 3-8. CPRI Signals (Continued)

Signal Name	Type	Signal Description
GPIO17	Input/ Output	**General-Purpose Input Output 17** One of 32 GPIOs. For details, see **Chapter 20**, *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.
CP_LOS3	Input	**CPRI Optical Loss of Signal Indication for Lane 3** This is an optional configuration. The signal can also be configured on GE2_TD2.
GE2_TX_CTL	Output	**Ethernet 2 Transmit Control** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
CP_LOS4	Input	**CPRI Optical Loss of Signal Indication for Lane 4** This is the default option. The signal can also be configured on GPIO18.
GPIO18	Input/ Output	**General-Purpose Input Output 18** One of 32 GPIOs. For details, see **Chapter 20**, *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.
CP_LOS4	Input	**CPRI Optical Loss of Signal Indication for Lane 4** This is an optional configuration. The signal can also be configured on GE2_GTX_CLK.
GE2_TD3	Output	**Ethernet 2 Transmit Data 3** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
CP_LOS5	Input	**CPRI Optical Loss of Signal Indication for Lane 5** This is the default option. The signal can also be configured on GPIO19.
GPIO19	Input/ Output	**General-Purpose Input Output 19** One of 32 GPIOs. For details, see **Chapter 20**, *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.
CP_LOS5	Input	**CPRI Optical Loss of Signal Indication for Lane 5** This is an optional configuration. The signal can also be configured on GE2_TD3.
GE2_TD2	Output	**Ethernet 2 Transmit Data 2** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
CP_LOS6	Input	**CPRI Optical Loss of Signal Indication for Lane 6** This is the default option. The signal can also be configured on GPIO20.

Table 3-8. CPRI Signals (Continued)

Signal Name	Type	Signal Description
GPIO20	Input/ Output	**General-Purpose Input Output 20** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*. Valid in all modes.
$\overline{\text{SPI\_SL}}$	Input	**SPI Select** $\overline{\text{Enable}}$ 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.
CP_LOS6	Input	**CPRI Optical Loss of Signal Indication for Lane 6** This is an optional configuration. The signal can also be configured on GE2_RD0.

3.7 Ethernet Signals

The RGMII signals are listed and described in **Table 3-9**.

Table 3-9. Ethernet Signals

Signal Name	Type	Description
GE1_RD3	Input	**Ethernet 1 Receive Data 3** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE1_RD2	Input	**Ethernet 1 Receive Data 2** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE1_RX_CLK	Input	**Ethernet 1 Receive Clock** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE1_RD0	Input	**Ethernet 1 Receive Data 0** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE1_GTX_CLK	Output	**Ethernet 1 Output Transmit Clock** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE1_RD1	Input	**Ethernet 1 Receive Data 1** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE1_TX_CLK	Input	**Ethernet 1 Transmit Clock** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE1_TD3	Output	**Ethernet 1 Transmit Data 3** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE1_TX_CTL	Output	**Ethernet 1 Transmit Control** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE1_TD1	Output	**Ethernet 1 Transmit Data 1** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE1_TD0	Output	**Ethernet 1 Transmit Data 0** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE1_TD2	Output	**Ethernet 1 Transmit Data 2** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
CP_LOS3	Input	**CPRI Optical Loss of Signal Indication for Lane 3** This is the default option. The signal can also be configured on GPIO17.

Table 3-9. Ethernet Signals (Continued)

Signal Name	Type	Description
GE2_TD0	Output	**Ethernet 2 Transmit Data 0** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE2_RX_CLK	Input	**Ethernet 2 Receive Clock** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE2_TD1	Output	**Ethernet 2 Transmit Data 1** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE2_TX_CLK	Input	**Ethernet 2 Transmit Clock** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE2_RD1	Input	**Ethernet 2 Receive Data 1** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE2_RX_CTL	Input	**Ethernet 2 Receive Control** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE2_TD3	Output	**Ethernet 2 Transmit Data 3** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
CP_LOS5	Input	**CPRI Optical Loss of Signal Indication for Lane 5** This is the default option. The signal can also be configured on GPIO19.
GE2_GTX_CLK	Output	**Ethernet 2 Output Transmit Clock** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE2_RD0	Input	**Ethernet 2 Receive Data 0** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE2_RD3	Input	**Ethernet 2 Receive Data 3** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
CP_LOS2	Input	**CPRI Optical Loss of Signal Indication for Lane 2** This is the default option. The signal can also be configured on GPIO29.
GE2_RD2	Input	**Ethernet 2 Receive Data 2** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
CP_LOS1	Input	**CPRI Optical Loss of Signal Indication for Lane 1** This is the default option. The signal can also be configured on GPIO28.
GE2_TD2	Output	**Ethernet 2 Transmit Data 2** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
CP_LOS6	Input	**CPRI Optical Loss of Signal Indication for Lane 6** This is the default option. The signal can also be configured on GPIO20.
GE2_TX_CTL	Output	**Ethernet 2 Transmit Control** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
CP_LOS4	Input	**CPRI Optical Loss of Signal Indication for Lane 4** This is the default option. The signal can also be configured on GPIO18.
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*.

Table 3-9. Ethernet Signals (Continued)

Signal Name	Type	Description
GE_MDIO	Input/ Output	**Ethernet Management Data Input/Output** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.

3.8 Serial Peripheral Interface (SPI) Signal Summary

Table 3-10. SPI Signals

Signal Name	Type	Signal Description
GPIO23	Input/ Output	**General-Purpose Input Output 23** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
TMR0	Input/ Output	**Timer 0** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *GPIO*. For timer functional details, see **Chapter 21**, *Timers*.
BOOT_SPI_SL	Input	**SPI Select During Boot** Enables the input to the SPI slave in single master mode during the boot sequence. In a multi-master environment, BOOT_SPI_SL detects an error when more one master is operating. Assertion of BOOT_SPI_SL while it is master causes an error. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GPIO20	Input/ Output	**General-Purpose Input Output 20** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*. Valid in all modes.
SPI_SL	Input	**SPI Select** Enable input to the SPI slave in single master mode. In multi-master environment, SPI_SL detects an error when more one master is operating. Assertion of an SPI_SL, while it is master, causes an error.
CP_LOS6	Input	**CPRI Optical Loss of Signal Indication for Lane 6** This is an optional configuration. The signal can also be configured on GE2_RD0.
GPIO19	Input/ Output	**General-Purpose Input Output 19** One of 32 GPIOs. For details, see **Chapter 20**, *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.
CP_LOS5	Input	**CPRI Optical Loss of Signal Indication for Lane 5** This is an optional configuration. The signal can also be configured on GE2_TD3.
GPIO18	Input/ Output	**General-Purpose Input Output 18** One of 32 GPIOs. For details, see **Chapter 20**, *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.
CP_LOS4	Input	**CPRI Optical Loss of Signal Indication for Lane 4** This is an optional configuration. The signal can also be configured on GE2_GTX_CLK.

Table 3-10. SPI Signals (Continued)

Signal Name	Type	Signal Description
GPIO17	Input/ Output	**General-Purpose Input Output 17** One of 32 GPIOs. For details, see **Chapter 20**, *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.
CP_LOS3	Input	**CPRI Optical Loss of Signal Indication for Lane 3** This is an optional configuration. The signal can also be configured on GE2_TD2.

3.9 GPIO and Maskable Interrupt Signal Summary

Table 3-11. 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 20**, *GPIO*.
I2C_SDA	Input/ Output	**I^{2}C-Bus Data Line** This is the data line for the I^{2}C bus. For details, see **Chapter 23**, *I2C*.
GPIO30	Input/ Output	**General-Purpose Input Output 30** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
I2C_SCL	Input/ Output	**I^{2}C-Bus Clock Line** This the clock line for the I^{2}C bus. For details, see **Chapter 23**, *I2C*.
GPIO29	Input/ Output	**General-Purpose Input Output 29** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
UART_TXD	Output	**UART Transmit Data** For details, see **Chapter 19**, *UART*.
CP_LOS2	Input	**CPRI Optical Loss of Signal Indication for Lane 2** This is an optional configuration. The signal can also be configured on GE2_RD3.
GPIO28	Input/ Output	**General-Purpose Input Output 28** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
UART_RXD	Input/ Output	**UART Receive Data** For details, see **Chapter 19**, *UART*.
CP_LOS1	Input	**CPRI Optical Loss of Signal Indication for Lane 1** This is an optional configuration. The signal can also be configured on GE2_RD2.

Table 3-11. GPIO and Maskable Interrupt Summary (Continued)

Signal Name	Type	Description
GPIO27	Input/ Output	**General-Purpose Input Output 27** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
TMR4	Input/ Output	**Timer 4** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *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 $\overline{\text{HRESET}}$/$\overline{\text{HRESET\_IN}}$ is asserted.
GPIO26	Input/ Output	**General-Purpose Input/Output 26** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*. .
TMR3	Input/ Output	**Timer 3** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *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 20**, *GPIO*.
TMR2	Input/ Output	**Timer 2** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *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 $\overline{\text{HRESET}}$/$\overline{\text{HRESET\_IN}}$ is asserted.
GPIO24	Input/ Output	**General-Purpose Input Output 24** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
TMR1	Input/ Output	**Timer 1** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *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 $\overline{\text{HRESET}}$/$\overline{\text{HRESET\_IN}}$ is asserted.

Table 3-11. GPIO and Maskable Interrupt Summary (Continued)

Signal Name	Type	Description
GPIO23	Input/ Output	**General-Purpose Input Output 23** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
TMR0	Input/ Output	**Timer 0** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *GPIO*. For timer functional details, see **Chapter 21**, *Timers*.
BOOT_SPI_SL	Input	**SPI Select During Boot** Enables the input to the SPI slave in single master mode during the boot sequence. In a multi-master environment, BOOT_SPI_SL detects an error when more one master is operating. Assertion of BOOT_SPI_SL while it is master causes an error. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GPIO22	Input/ Output	**General-Purpose Input Output 22** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
GPIO21	Input/ Output	**General-Purpose Input Output 21** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
TMR6	Input/ Output	**Timer 6** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 22**, *GPIO*. For functional details, see **Chapter 21**, *Timers*.
GPIO20	Input/ Output	**General-Purpose Input Output 20** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
SPI_SL	Input	**SPI Select** Enable input to the SPI slave in single master mode. In multi-master environment, SPI_SL detects an error when more one master is operating. Assertion of an SPI_SL, while it is master, causes an error. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
CP_LOS6	Input	**CPRI Optical Loss of Signal Indication for Lane 6** This is an optional configuration. The signal can also be configured on GE2_RD0.
GPIO19	Input/ Output	**General-Purpose Input Output 19** One of 32 GPIOs. For details, see **Chapter 20**, *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*.
CP_LOS5	Input	**CPRI Optical Loss of Signal Indication for Lane 5** This is an optional configuration. The signal can also be configured on GE2_TD3.
GPIO18	Input/ Output	**General-Purpose Input Output 18** One of 32 GPIOs. For details, see **Chapter 20**, *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*.
CP_LOS4	Input	**CPRI Optical Loss of Signal Indication for Lane 4** This is an optional configuration. The signal can also be configured on GE2_GTX_CLK.

Table 3-11. GPIO and Maskable Interrupt Summary (Continued)

Signal Name	Type	Description
GPIO17	Input/ Output	**General-Purpose Input Output 17** One of 32 GPIOs. For details, see **Chapter 20**, *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*.
CP_LOS3	Input	**CPRI Optical Loss of Signal Indication for Lane 3** This is an optional configuration. The signal can also be configured on GE2_TD2.
GPIO16	Input/ Output	**General-Purpose Input Output 16** One of 32 GPIOs. For details, see **Chapter 20**, *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.
TMR5	Input/ Output	**Timer 5** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 22**, *GPIO*. For functional details, see **Chapter 21**, *Timers*.
GPIO15	Input/ Output	**General-Purpose Input Output 15** One of the 32 GPIOs. For details, see **Chapter 20**, *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 20**, *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.

Table 3-11. GPIO and Maskable Interrupt Summary (Continued)

Signal Name	Type	Description
GPIO[13–8]	Input/ Output	**General-Purpose Input Output 13–8** Nine of the 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
$\overline{\text{IRQ}}$[13–8]	Input	**Interrupt Request 13–8** External lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
RC[13–8]	Input	**Reset Configuration Word Bit 13–8** Sampled during the assertion of $\overline{\text{PORESET}}$ to set part of the bits of the Reset Configuration Word Registers.
GPIO[7–5]	Input/ Output	**General-Purpose Input Output 7–5** Nine of the 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
$\overline{\text{IRQ}}$[7–5]	Input	**Interrupt Request 7–5** External lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
RC[7–5]	Input	**Reset Configuration Word Bit 7–5** Sampled during the assertion of $\overline{\text{PORESET}}$ to set part of the bits of the Reset Configuration Word Registers.
CP_SYNC[6–4]	Input/ Output	**CPRI Controller 4–6 Synchronization Signal** Synchronization signals for CPRI controllers 4–6. Selected through GPIO configuration. For configuration details, see **Chapter 20, *GPIO***. For functional details, see **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*.
GPIO4	Input/ Output	**General-Purpose Input Output 4** One of the 32 GPIOs. For details, see **Chapter 20**, *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 20**, *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.

Table 3-11. GPIO and Maskable Interrupt Summary (Continued)

Signal Name	Type	Description
GPIO[2–0]	Input/ Output	**General-Purpose Input Output 2–0** Three of the 32 GPIOs. For details, see **Chapter 20**, *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.
CP_SYNC[3–1]	Input/ Output	**CPRI Controller 1–3 Synchronization Signal** Synchronization signals for CPRI controllers 1–3. Selected through GPIO configuration. For configuration details, see **Chapter 20, *GPIO***. For functional details, see **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*.

3.10 Timer Signals

Table 3-12 describes the signals in this group.

Table 3-12. Timer Signals

Signal Name	Type	Description
TMR6	Input/ Output	**Timer 6** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 22**, *GPIO*. For functional details, see **Chapter 21**, *Timers*.
GPIO21	Input/ Output	**General-Purpose Input Output 21** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
TMR5	Input/ Output	**Timer 5** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 22**, *GPIO*. For functional details, see **Chapter 21**, *Timers*.
GPIO16	Input/ Output	**General-Purpose Input Output 16** One of 32 GPIOs. For details, see **Chapter 20**, *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.

Table 3-12. Timer Signals (Continued)

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 configuration details, see **Chapter 20**, *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 20**, *GPIO*.
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 $\overline{\text{HRESET}}$/$\overline{\text{HRESET\_IN}}$ is asserted.
TMR3	Input/ Output	**Timer 3** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *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 20**, *GPIO*.
TMR2	Input/ Output	**Timer 2** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *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 20**, *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 $\overline{\text{HRESET}}$/$\overline{\text{HRESET\_IN}}$ is asserted.
TMR1	Input/ Output	**Timer 1** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *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 20**, *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 $\overline{\text{HRESET}}$/$\overline{\text{HRESET\_IN}}$ is asserted.

Table 3-12. 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 configuration details, see **Chapter 20**, *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 20**, *GPIO*.
BOOT_SPI_SL	Input	**SPI Select During Boot** Enables the input to the SPI slave in single master mode during the boot sequence. In a multi-master environment, BOOT_SPI_SL detects an error when more one master is operating. Assertion of BOOT_SPI_SL while it is master causes an error. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.

3.11 UART Signals

Table 3-13. UART Signals

Signal Name	Type	Description
UART_TXD	Output	**UART Transmit Data** Selected through the GPIO configuration. For configuration details, see **Chapter 20, *GPIO***. For functional details, see **Chapter 19, *UART***.
GPIO29	Input/ Output	**General-Purpose Input Output 29** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*.
CP_LOS2	Input	**CPRI Optical Loss of Signal Indication for Lane 2** This is an optional configuration. The signal can also be configured on GE2_RD3.
UART_RXD	Input/ Output	**UART Receive Data** Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *GPIO*. For functional details, see **Chapter 19**, *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 20**, *GPIO*.
CP_LOS1	Input	**CPRI Optical Loss of Signal Indication for Lane 1** This is an optional configuration. The signal can also be configured on GE2_RD2.

3.12 I2C Signals

Table 3-14. I2C Signals

Signal Name	Type	Description
I2C_SDA	Input/ Output	**I2C-Bus Data Line** This is the data line for the I2C bus. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *GPIO*. For functional details, see **Chapter 23**, *I2C*.
GPIO31	Input/ Output	**General-Purpose Input Output 31** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*. Valid in all modes.
I2C_SCL	Input/ Output	**I2C-Bus Clock Line** This the clock line for the I2C bus. Selected through the GPIO configuration. For configuration details, see **Chapter 20**, *GPIO*. For functional details, see **Chapter 23**, *I2C*.
GPIO30	Input/ Output	**General-Purpose Input/Output 30** One of 32 GPIOs. For details, see **Chapter 20**, *GPIO*. Valid in all modes.

3.13 External DMA Signals

Table 3-15. 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 20**, *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 20**, *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.

Table 3-15. External DMA Signals (Continued)

Signal Name	Type	Description
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 20**, *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.
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 20**, *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.

Table 3-16. Other Interrupt Signals

Signal Name	Type	Description
$\overline{\text{INT\_OUT}}$	Output	**Interrupt Output** An open-drain output driven from the MSC8157E virtual interrupt 24. Assertion of this output indicates that an unmasked interrupt is pending in the MSC8157E internal interrupt controller.
$\overline{\text{CP\_TX\_INT}}$	Output	**CPRI Transmit Interrupt** CPRI Transmit Interrupt. For Details see **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*.
$\overline{\text{NMI}}$	Input	**Non-Maskable Interrupt** An external device may assert this line to generate a non-maskable interrupt to the MSC8157E device.
$\overline{\text{NMI\_OUT}}$	Output	**Non-Maskable Interrupt Output** An open-drain pin driven from the MSC8157E virtual interrupt 25. Assertion of this output indicates that a non-maskable interrupt is pending in the MSC8157E internal interrupt controller, waiting to be handled by an external host.
$\overline{\text{CP\_RX\_INT}}$	Output	**CPRI Receive Interrupt** CPRI Receive Interrupt. For Details see **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*.

3.14 OCE Event and JTAG Test Access Port Signals

The MSC8157E 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 MSC8157E supports the standard set of test access port (TAP) signals defined by **IEEE** Std. 1149.6 Test Access Port and Boundary-Scan Architecture specification and described in **Table 3-17**.

Table 3-17. JTAG TAP Signals

Signal Name	Type	Signal Description
DFT_TEST	Input	**Manufacturing Test Mode** Always connect to VSS. This is not a ground pin.
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 $\overline{\text{PORESET}}$.

4 Chip-Level Arbitration and Switching System (CLASS)

The Chip Level Arbitration and Switching System (CLASS) is the central internal interconnect system for the MSC8157E 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 667 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 fifteen CLASS initiators are:

- Six SC3850 core subsystems (initiator ports 0–5)
- Four MAPLE bridges (initiator ports 6–7 and ports 13–14)
- One 10-lane port shared by two Serial RapidIO controllers, the PCI Express controller, six CPRI controllers, and two SGMIIs (connects to two initiator ports 8 and 12)
- Peripherals bridge shared by the SEC, SPI, RGMII, SGMII, and JTAG interface (initiator port 9)
- Two DMA ports (initiator ports 10–11)

The ten CLASS targets are:

- Configuration Control and Status Registers (CCSR) (target port 0)
- Two DDR ports (target ports 1 for writes and 2 for reads)
- MAPLE module (target port 3)
- Three core subsystem bridges (two core subsystems per bridge) (target ports 4–6)
- Three M3 memory ports (target ports 7–9)

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

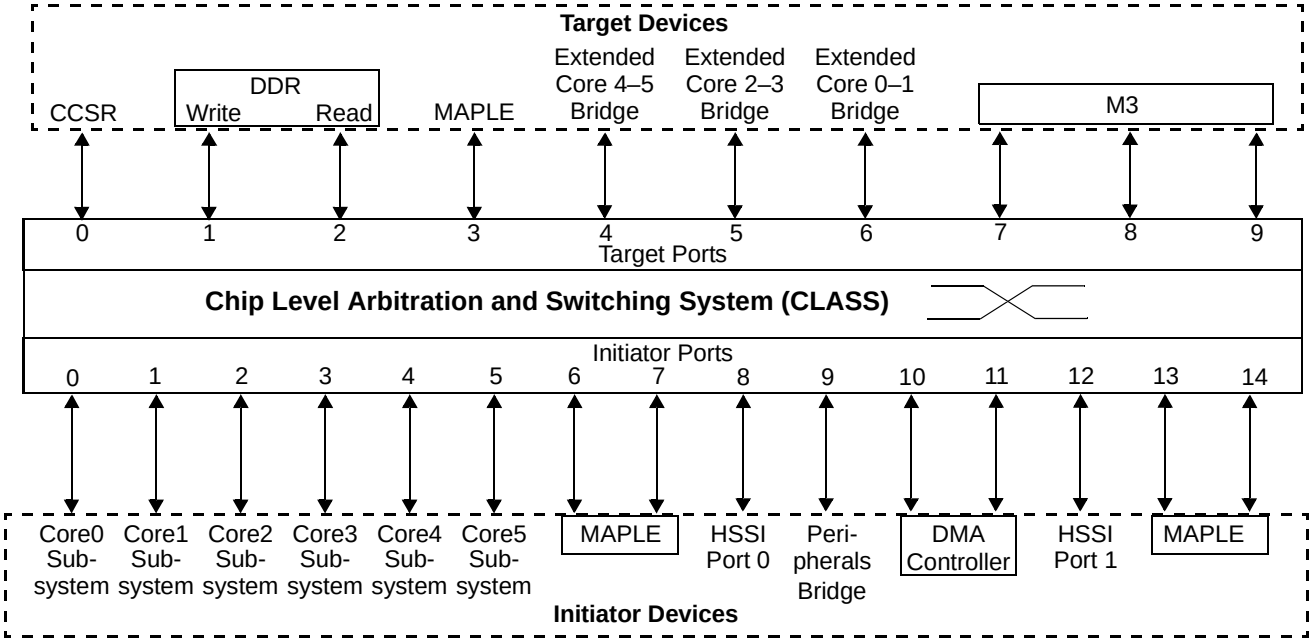


Figure 4-1. CLASS Initiators and Targets in the MSC8157E Device

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 15 initiators and 10 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 **4.7.27**, *CLASS Arbitration Control Register (C0ACR)*). Late arbitration is enabled by default.

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 **4.7.3**, *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 **4.7.2**, *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 4.7** for programming details.

4.3 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.

4.4 CLASS Debug Profiling Unit

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

4.4.1 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.

4.4.2 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.

4.4.3 Event Selection

Events are selected using a combination of the CLASS watch point and profiling registers. **Table 4-1** 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 4.7** for the register details.

Table 4-1. 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.							

Table 4-1. 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 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 Target Switch (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.							
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.							

Table 4-1. 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-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.							
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							

Table 4-1. 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
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)							

4.4.4 Debug and Profiling Events

The CLASS generates two event interrupts:

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

4.5 CLASS Reset

The CLASS implements 2 kinds of reset:

- Synchronous hard reset.
- Synchronous soft reset.

4.5.1 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 **4.7**, *Programming Model*.

4.5.2 Hard Reset

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

4.6 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 MAPLE-B2 to access itself.
- Core subsystem 0 can only access its own M2 space and the M2 space of core subsystems 2, 3, 4, and 5.
- Core subsystem 1 can only access its own M2 space and the M2 space of core subsystems 2, 3, 4, and 5.
- Core subsystem 2 can only access its own M2 space and the M2 space of core subsystems 4 and 5.
- Core subsystem 3 can only access its own M2 space and the M2 space of core subsystems 4 and 5.
- Core subsystems 4 and 5 can only access their own M2 space.

4.7 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 4-15**)
- CLASS Priority Auto Upgrade Value Registers (see **page 4-16**)
- CLASS Priority Auto Upgrade Control Registers (see **page 4-17**)
- CLASS Error Address Registers (see **page 4-18**)
- CLASS Error Extended Address Registers (see **page 4-19**)
- CLASS Initiator Profiling Configuration Registers (see **page 4-20**)
- CLASS Initiator Watch Point Control Registers (see **page 4-22**)
- CLASS Arbitration Weight Registers (see **page 4-23**)
- CLASS Start Address Decoder x (see **page 4-24**)
- CLASS End Address Decoder x (see **page 4-25**)
- CLASS Attributes Decoder x (see **page 4-26** and **page 4-28**)
- CLASS IRQ Status Register (see **page 4-29**)
- CLASS IRQ Enable Register (see **page 4-30**)
- CLASS Target Profiling Configuration Register (see **page 4-30**)
- CLASS Profiling Control Register (see **page 4-32**)
- CLASS Watch Point Control Register (see **page 4-33**)
- CLASS Watch Point Access Configuration Register (**page 4-34**)
- CLASS Watch Point Extended Access Configuration Register (see **page 4-35**)
- CLASS Watch Point Address Mask Register (see **page 4-36**)
- CLASS Profiling Time Out Register (see **page 4-37**)
- CLASS Target Watch Point Control Register (see **page 4-38**)
- CLASS Profiling IRQ Status Register (see **page 4-39**)
- CLASS Profiling IRQ Enable Register (see **page 4-40**)
- CLASS Profiling Reference Counter Register (see **page 4-40**)
- CLASS Profiling General Counter Registers (see **page 4-41**)
- CLASS Arbitration Control Register (see **page 4-42**)

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

4.7.1 CLASS Priority Mapping Registers (C0PMRx)

C0PMR[0–14] 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

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

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.

Table 4-2 lists the C0PMRx bit field descriptions.

Table 4-2. 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
— 3–2	0	Reserved. Write to 0 for future compatibility.	

Table 4-2. C0PMRx Bit Descriptions (Continued)

Name	Reset	Description	Settings
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

4.7.2 CLASS Priority Auto Upgrade Value Registers (C0PAVRx)

C0PAVR[0–14] 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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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 4-3 lists the C0PAVRx bit field descriptions.

Table 4-3. C0PAVRx Bit Descriptions

Name	Reset	Description
— 31–16	0	Reserved. Write to 0 for future compatibility.
AUV 15—0	0	**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.

4.7.3 CLASS Priority Auto Upgrade Control Registers (C0PACRx)

C0PACR[0–14] 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 4-4 lists the C0PACRx bit field descriptions.

Table 4-4. 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.

4.7.4 CLASS Error Address Registers (C0EARx)

C0EAR[0–14] 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 4-5 lists the C0EARx bit field descriptions.

Table 4-5. 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-B2 port 0.
- C0EAR7 = Address generated by MAPLE-B2 port 1.
- C0EAR8 = Address generated by HSSI port 0.
- C0EAR9 = Address generated by peripheral bridge.
- C0EAR10 = DMA port 0.
- C0EAR11 = DMA port 1.
- C0EAR12 = Address generated by HSSI port 1
- C0EAR13 = Address generated by MAPLE-B2 port 2.
- C0EAR14 = Address generated by MAPLE-B2 port 3.

4.7.5 CLASS Error Extended Address Registers (C0EEARx)

C0EEAR[0–14] 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 4-6 lists the C0EEARx bit field descriptions.

Table 4-6. 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.	

Table 4-6. C0EEARx Bit Descriptions (Continued)

Name	Reset	Description	Settings
SRC_ID 8–4	0	**Source ID** Identifies the source ID of the initiator that caused the error.	0x00 Core subsystem 0 0x01 Core subsystem 1 0x02 Core subsystem 2 0x03 Core subsystem 3 0x04 Core subsystem 4 0x05 Core subsystem 5 0x06 MAPLE-B2 port 0 0x07 MAPLE-B2 port 1 0x08 DMA port 0 0x09 DMA port 1 0x0B QUICC Engine subsystem 0x0C Serial RapidIO Port 0 0x0D Serial RapidIO Port 1 0x11 CPRI read 0x12 CPRI write 0x13 eMSG read 0x14 eMSG write 0x15 MAPLE-B2 port 2 0x16 MAPLE-B2 port 3 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.	

4.7.6 CLASS Initiator Profiling Configuration Registers (C0IPCRx)

C0IPCR[0–14] 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 4-7 lists the C0IPCRx bit field descriptions.

Table 4-7. 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 Target 8 bandwidth 11001 Target 9 bandwidth 11010– 11111 reserved

Table 4-8. 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 port 0
9	SEC, Ethernet, SPI, or Debug
10	DMA port 0
11	DMA port 1
12	HSSI port 1
13	MAPLE port 2
14	MAPLE port 3

4.7.7 CLASS Initiator Watch Point Control Registers (C0IWPCRx)

C0IWPCR[0–14] 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 4-9 lists the C0IWPCRx bit field descriptions.

Table 4-9. 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.

4.7.8 CLASS Arbitration Weight Registers (C0AWRx)

C0AWR[0–14] 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																
x = 0–9, 12–14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x = 10–11	0	0	0	0	0	0	0	0	0	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																
x = 0–9, 12–14	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
x = 10–11	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

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 4-10 lists the C0AWRx bit field descriptions.

Table 4-10. C0AWRx Bit Descriptions

Name	Reset	Description
— 31–4	0	Reserved. Write to 0 for future compatibility.
WEIGHT 3–0	*For C0AWR[0–9] and C0AWR[12–14]:* 1111 *For C0AWR[10–11]:* 0111	**Weight** Contains the arbitration weight assigned to the associated initiator.

The reset values be changed by the designer according to the application requirements. See **Table 4-30** for recommended initial settings for the CLASS Arbitration Control Register (C0ACR).

4.7.9 CLASS Start Address Decoder x (C0SADx)

C0SAD1 CLASS Start Address Decoders Offset 0xC00 +x*0x04
C0SAD2
C0SAD7
C0SAD8
C0SAD9

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																
SAD1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
SAD2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SAD7	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
SAD8	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
SAD9	0	0	0	0	0	0	0	0	0	0	0	0	1	1	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																
SAD1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SAD2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SAD7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SAD8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
SAD9	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

C0SADx configure the address decoding of CLASS toward the DDR controller (C0SAD1/C0SAD2) and the M3 memory (C0SAD[7–9]). 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.

Note: C0SAD2 is valid only if C0ATD1[SPRW] is clear (0).

These registers are reset by a hardware reset only. **Table 4-26** lists the C0SADx bit field descriptions.

Table 4-11. C0SADx Bit Descriptions

Name	Reset	Description
— 31–24	0	Reserved. Write to 0 for future compatibility.
SA[35–12] 23–0	Port 1 = 0x040000 (DDR start) Port 2 = 0x000000 (DDR start) Port 7 = 0x0C0000 (M3_0 start) Port 8 = 0x0C0100 (M3_1 start) Port 9 = 0x0C0200 (M3_2 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.

4.7.10 CLASS End Address Decoder x (C0EADx)

C0EAD1 CLASS End Address Decoders Offset 0xC40 + x*0x04
C0EAD2
C0EAD7
C0EAD8
C0EAD9

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																
EAD1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
EAD2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EAD7	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
EAD8	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
EAD9	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

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																
EAD1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
EAD2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EAD7	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
EAD8	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
EAD9	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1

C0EADx configure the address decoding of CLASS toward the DDR controller (C0EAD1/C0EAD2) and the M3 memory (C0EAD[7–9]). 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.

Note: C0EAD2 is valid only if C0ATD1[SPRW] is clear (0).

These registers are reset by a hardware reset only. **Table 4-26** lists the C0EADx bit field descriptions.

Table 4-12. C0EADx Bit Descriptions

Name	Reset	Description
— 31–24	0	Reserved. Write to 0 for future compatibility.
EA[35–12] 23–0	Port 1 = 0x0BFFFF (DDR end) Port 2 = 0x000000 (DDR end) Port 7 = 0x0C00FF (M3_0 end) Port 8 = 0x0C01FF (M3_1 end) Port 9 = 0x0C02FF (M3_2 end)	**End Address 35–12** The 24 msb of the end address of the specified port window. The lsbs are all ones. 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.

4.7.11 CLASS Attributes Decoder 1 (C0ATD1)

C0ATD1 CLASS Attributes Decoder 1 Offset 0xC84

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
	—											SPRW	—			DEN
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

C0ATD1 controls the functionality of each specific decoder for CLASS toward the DDR controller ports. In addition to containing the bit that enables/disables the port 1, this register also contains a bit to determine whether the DDR access uses a direct connection or uses split read/write access split between the two DDR ports.

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.

This registers is reset by a hardware reset only. **Table 4-26** lists the C0ATD1 bit field descriptions.

Table 4-13. C0ATD1 Bit Descriptions

Name	Reset	Description	Settings
— 31–5	0	Reserved. Write to 0 for future compatibility.	
SPRW 4	1	**Split Read/Write** Determines whether reads and writes both occur on a port or they are split between two ports. **Note:** **Table 4-14** summarizes the DDR access configuration options.	0 DDR read and write accesses use both target ports1 and 2. Address decoding for each port is done by the respective register sets: C0SAD1/C0EAD1 are used for port 1 and C0SAD2/C0EAD2 are used for port 2. Use of this mode may improve performance for applications with an unbalanced proportion of reads and writes toward DDR, or it can be used to divide core access to DDR (for example, some of the cores can use the low DDR address space and the other cores can use the high DDR address space). 1 DDR reads and writes go through different ports (DDR read uses target 2 and DDR writes use target 1). The start and end addresses for both ports are controlled by C0SAD1 and C0EAD1.
— 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 this register when there are open transactions being handled by the CLASS to the specified target controlled by the register.

Table 4-14. DDR Access Configurations

Scenario	C0ATD1[DEN]	C0ATD2[DEN]	COATD1[SPRW]	Result
1	0	0	x	No accesses can be made to DDR
2	0	1	x	All DDR accesses use Target2 as defined by C0SAD2/C0EAD2. (no benefit)
3	1	0	0	All DDR accesses use Target1 as defined by C0SAD1/C0EAD1. (no benefit)
4	1	x	1	Writes to DDR use Target 1. Reads from DDR use Target 2. Both ports are defined by C0SAD1/C0EAD1.
5	1	1	0	Some accesses use Target 1 (defined by C0SAD1/C0EAD1). Some accesses use Target 2 (defined by C0SAD2/C0EAD2). Use for cases of unbalanced reads/writes or to split address space among different masters. **Note:** Target2 has a higher priority than Target1.

4.7.12 CLASS Attributes Decoder x (C0ATDx)

C0ATD2 CLASS Attributes Decoders Offset 0xC80 + X*0x04
C0ATD7
C0ATD8
C0ATD9

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																
ATD2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ATD7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
ATD8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
ATD9	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 controller (C0ATD2 if COATD1[SPWR] is cleared) and the M3 memory (C0ATD[7–9]). 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 4-26** lists the C0ATDx bit field descriptions.

Table 4-15. C0ATDx Bit Descriptions

Name	Reset	Description	Settings
— 31–1	0	Reserved. Write to 0 for future compatibility.	
DEN 0	C0ATD2 = 0 C0ATD[7–9] = 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.

Note: C0ATD2 is valid only if C0ATD1[SPRW] is clear (0).

4.7.13 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
	—	AEI14	AEI13	AEI12	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 4-16 lists the C0ISR bit field descriptions.

Table 4-16. C0ISR Bit Descriptions

Name	Reset	Description	Settings
— 31–15	0	Reserved. Write to 0 for future compatibility.	
AEI[14–0] 14–0	0	**Address Error Interrupt 14–0** 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.

4.7.14 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
	—	AEIE14	AEIE13	AEIE12	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 4-17 lists the C0IER bit field descriptions.

Table 4-17. C0IER Bit Descriptions

Name	Reset	Description	Settings
— 31–15	0	Reserved. Write to 0 for future compatibility.	
AEIE[14–0] 14–0	0	**Address Error Interrupt Enable** Used to enable/disable the address error interrupt for an initiator.	0 Interrupt masked. 1 Interrupt enabled.

4.7.15 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 4-18 lists the C0TPCR bit field descriptions.

Table 4-18. 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.
TN 11–8	0	**Target Number** Indicates the number of selected target.	0000 CCSR 0001 DDR write 0010 DDR read 0011 MAPLE 0100 Core 4 and 5 Bridge 0101 Core 2 and 3 Bridge 0110 Core 0 and 1 Bridge 0111 M3 port 0 1000 M3 port 1 1001 M3 port 2 All other values reserved.
— 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.

4.7.16 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 4-19** lists the C0PCR bit field descriptions.

Table 4-19. 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.

4.7.17 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 4-20** lists the C0WPCR bit field descriptions.

Table 4-20. 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.	

Table 4-20. C0WPCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
SPVE 3	0	**Supervisor Access Compare Enable** Enables/disables supervisor access comparison.	0 Supervisor type compare disabled. 1 Supervisor type compare with C0WRACR enabled.
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.

4.7.18 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 4-21** lists the C0WPACR bit field descriptions.

Table 4-21. 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.			

4.7.19 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 4-22** lists the C0WPEACR bit field descriptions.

Table 4-22. 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	0	Reserved. Write to 0 for future compatibility.	
EATTR 28–24	0	**EOT Attributes** Defines the EOT attributes to monitor.	0x0– 0x7 Reserved 0x8 Target port 0 CCSRs 0x9 Target port 1 DDR write 0xA Target port 2 DDR read 0xB Target port 3 MAPLE-B2 0xC Target port 4 Cores 4 and 5 0xD Target port 5 Cores 2 and 3 0xE Target port 6 Cores 0 and 1 0xF Target port 7 M3 memory port 0 0x10 Target port 8 M3 memory port 1 0x11 Target port 9 M3 memory port 2 All other values reserved.
— 23–16	0	Reserved. Write to 0 for future compatibility.	

Table 4-22. 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 HSSI Port 0 0x09 Peripherals Bridge 0x0A DMA Port 0 0x0B DMA Port 1 0x0C HSSI Port 1 0x0D MAPLE Port 2 0x0E MAPLE Port 3
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.

4.7.20 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 4-23** lists the C0WPAMR bit field descriptions.

Table 4-23. 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.

4.7.21 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 4-24** lists the C0PTOR bit field descriptions.

Table 4-24. 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.

4.7.22 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 4-16 lists the C0TWPCR bit field descriptions.

Table 4-25. 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.

4.7.23 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 4-26** lists the C0PISR bit field descriptions.

Table 4-26. 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.

4.7.24 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 4-27** lists the C0PIER bit field descriptions.

Table 4-27. 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.

4.7.25 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 4-28** lists the C0PRCR bit field descriptions.

Table 4-28. C0PRCR Bit Descriptions

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

4.7.26 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 4-29** lists the C0PGCR bit field descriptions.

Table 4-29. C0PGCR Bit Descriptions

Name	Reset	Description
CNT 31–0	0	**Counter** Holds the counter value of the selected measurement. **Table 4-1** lists the measurements counted by each counter for each configuration combination.

4.7.27 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
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
	—						LA9	LA8	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
Boot	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

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 about the winner according to the MDBW parameter and the byte count of the winner access. When Late Arbitration mode is disabled, the arbiter makes a decision every clock cycle.The register is reset by a hard reset only. **Table 4-30** lists the C0ACR bit field descriptions.

Table 4-30. 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.
— 27–10	0	Reserved. Write to 0 for future compatibility.	
LA[9–0] 9–0	1	**Late Arbitration 9–0** Enables/disables late arbitration mode for the associated arbiter. **Note:** As with the arbitration weight (see **Section 4.7.8**, *CLASS Arbitration Weight Registers (C0AWRx)*, on page 4-23), the default value for this field does not yield optimal performance. Therefore, the boot program after reset, writes a value of 0x3FF 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.

5 Reset

The reset and control signals provide many options for MSC8157E 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 MSC8157E 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 MSC8157E device.

5.1 Reset Operations

The MSC8157E has several inputs to the reset logic:

- Power-on reset ($\overline{\text{PORESET}}$)
- External hard reset ($\overline{\text{HRESET}}$, $\overline{\text{HRESET\_IN}}$) (See **Chapter 3**, *External Signals*for detailed signal descriptions)
- Software watchdog reset
- RapidIO reset
- Software hard 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.

Note: If the device is reset by any source, always reload all data into M3 and M2 memory before continuing operation.

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 or HRESET_IN)	HRESET is a bidirectional I/O pin. HRESET is an open-drain pin. HRESET_IN is an input that permits a hard reset of an individual device without invoking a hard reset on other devices in a chain. The MSC8157E can detect an external assertion of HRESET or HRESET_IN only if it occurs while the MSC8157E is not asserting reset.
Software watchdog reset	After the MSC8157E 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.
Software hard reset	A hard reset sequence can be initialized by writing to a memory mapped register (RCR)

5.1.2 Reset Actions

The MSC8157E 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. 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. **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 4-14 for details)	Reset Configuration Words Loaded	Other Internal Logic	HRESET or HRESET_IN Driven	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

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 MSC8157E device. When $\overline{\text{PORESET}}$ is deasserted, the MSC8157E starts the configuration process. The MSC8157E asserts $\overline{\text{HRESET}}$ 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 MSC8157E 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.

5.1.4 Detailed Power-On Reset Flow

The detailed power-on reset ($\overline{\text{PORESET}}$) flow for the MSC8157E is as follows:

1. The user asserts $\overline{\text{PORESET}}$ (and optionally $\overline{\text{HRESET}}$ or $\overline{\text{HRESET\_IN}}$).
2. Power is applied to meet the specifications in the *MSC8157E Technical Data Sheet*.
3. The user asserts $\overline{\text{PORESET}}$ (and optionally $\overline{\text{HRESET}}$ or $\overline{\text{HRESET\_IN}}$) 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 *MSC8157E 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 $\overline{\text{TRST}}$.

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 MSC8157E is now in its ready state.

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

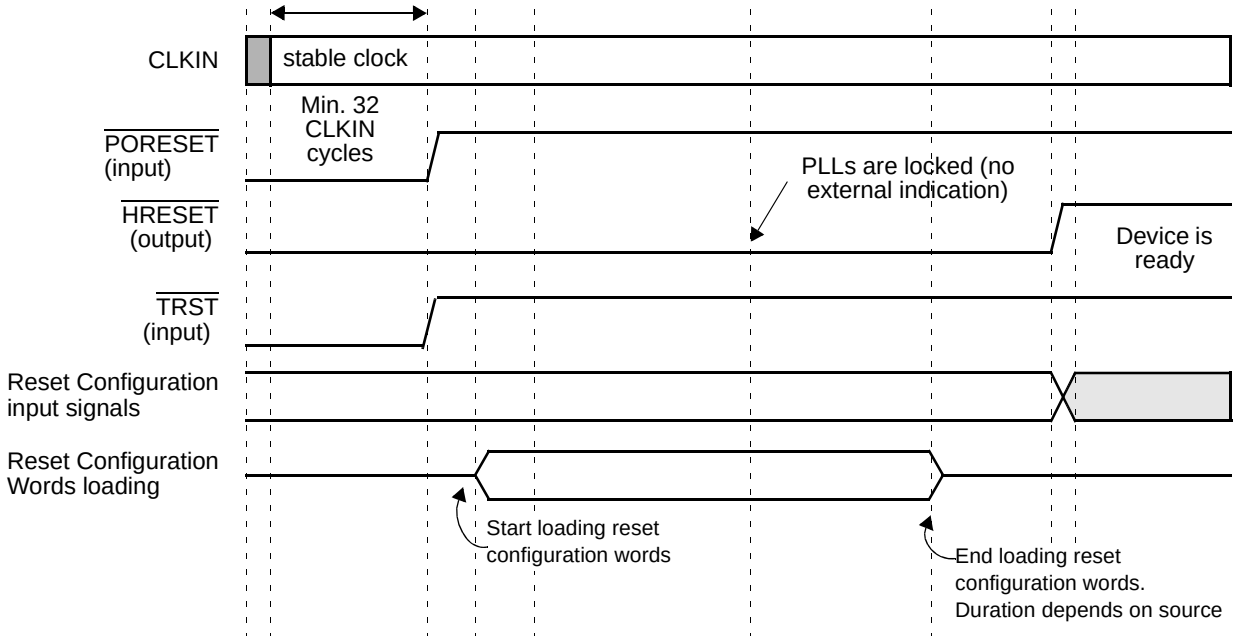


Figure 5-1. Power-On Reset Flow

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.

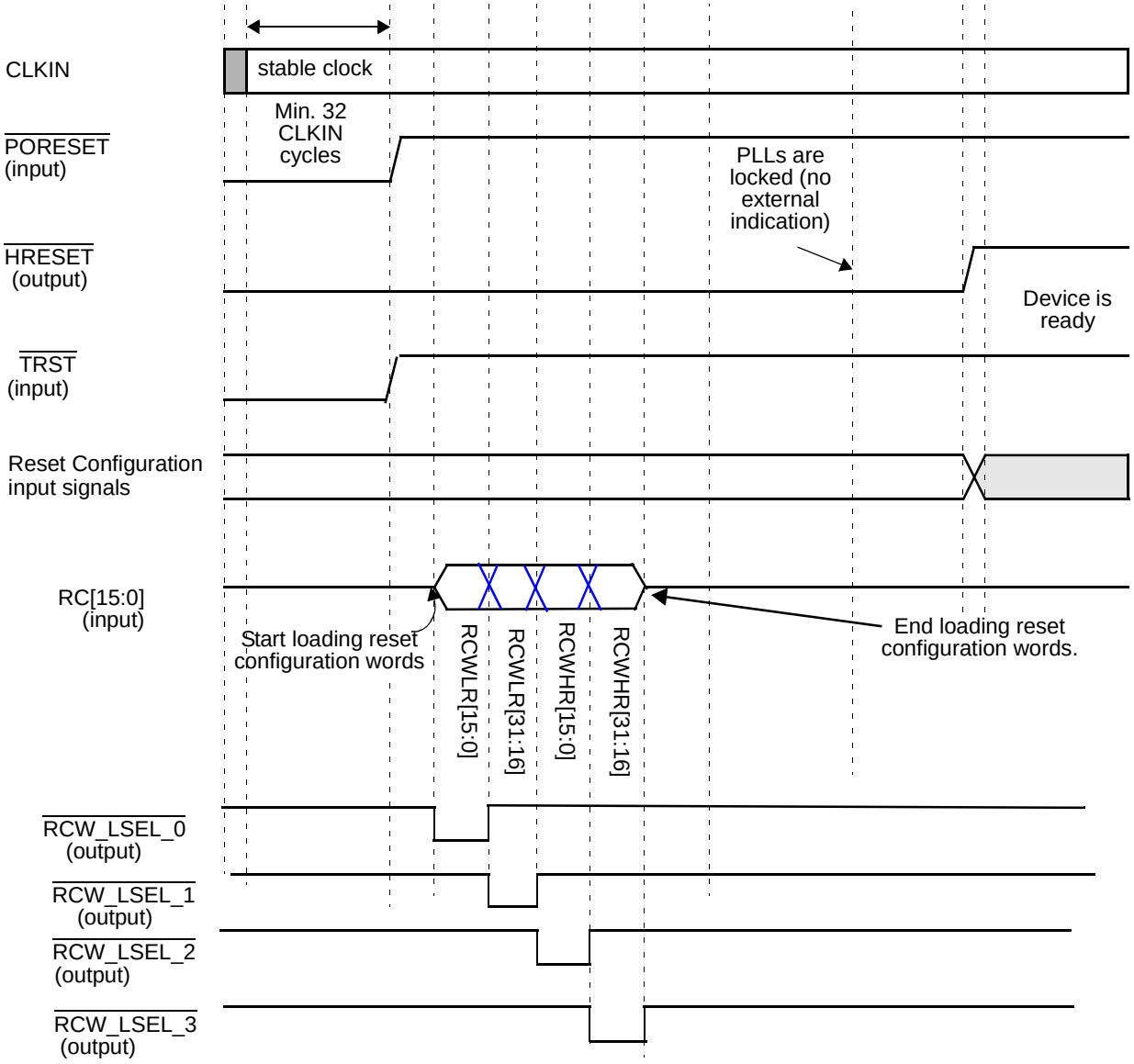


Figure 5-2. Power-on Reset Flow for RCW_SRC = 000

5.1.5 HRESET Flow

The HRESET flow may be initiated externally by asserting HRESET/ HRESET_IN or internally when the MSC8157E detects a reason to assert HRESET. In both cases, the device continues asserting HRESET throughout the HRESET flow. For HRESET, the hard reset sequence time is 4908 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 MSC8157E immediately configures the device. For the HRESET_IN signal, the user asserts the signal in the same way as HRESET, but the deassertion is not automatic and is determined by the user; the signal is not held low by the reset block to the end of the reset sequence. After the configuration sequence completes, the MSC8157E releases the HRESET signal and exits the HRESET flow. Use an external pull-up resistor to deassert the signal. After deassertion is detected, the device waits for a 16-cycle period before testing the presence of an external (hard) 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 MSC8157E 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.

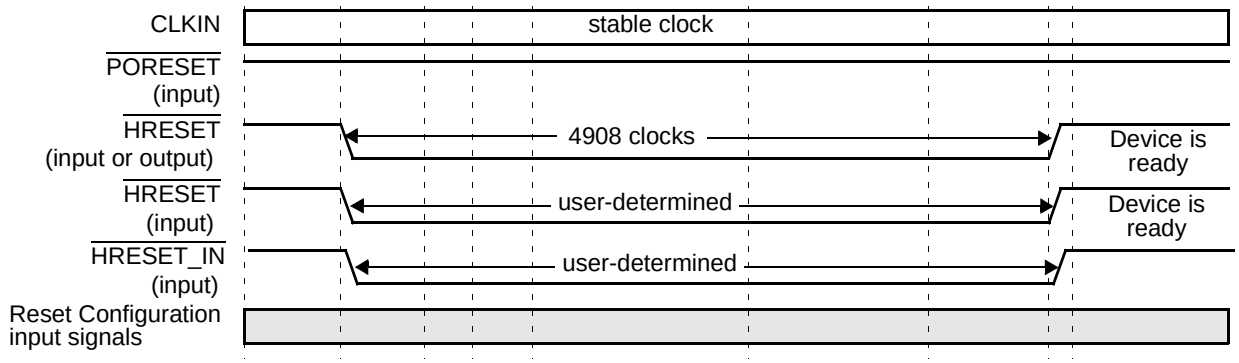


Figure 5-3. Hard Reset Flow

5.2 Reset Configuration

The MSC8157E 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 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 HRESET. Depending on the configuration signal values, the MSC8157E 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 *MSC8157E 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 MSC8157E 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	Reserved.
010	Reset configuration word is loaded from an I^2C EEPROM in 16-bit addressing mode. MSC8157E 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 $\overline{\text{HRESET}}$.

Table 5-4. Selecting Reset Configuration Input Signals

CLKIN Frequency	RCW_SRC[0–2]	Reset Sequence Duration in CLKIN Cycles	Duration in ms
133.33 MHz	000, 011-101	21169	0.159
133.33 MHz	010 **Note:** Loading the RCW via I^2C is the slower method for loading the RCW.	264424	1.98

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 MSC8157E 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 MSC8157E 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.

Note: The reset configuration word cannot be loaded from an I^2C EEPROM in 8-bit addressing mode.

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

When a MSC8157E 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 24**, *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 MSC8157E 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 uses 0b1010000 for the EEPROM calling address. A reset target 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 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 5-10.

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 MSC8157E 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**.

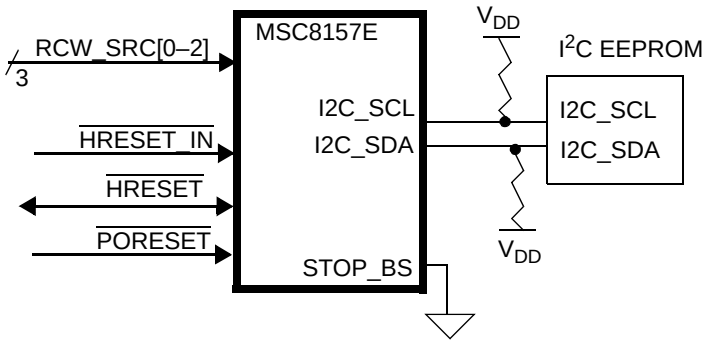


Figure 5-4. Single Device I²C Reset Configuration

5.2.5.1.5 Loading Multiple Devices From a Single I²C EEPROM

When the MSC8157E device shares the I²C EEPROM device with other MSC8157E 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 *MSC8157E 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.

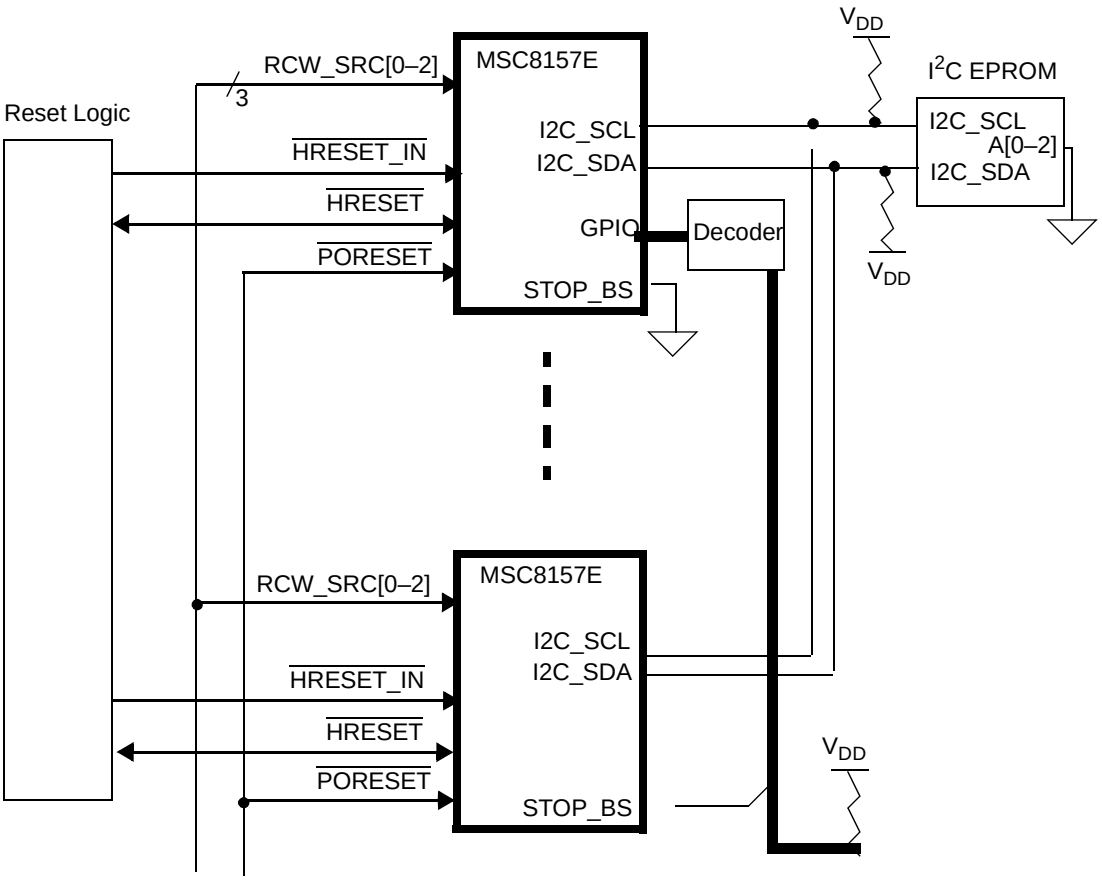


Figure 5-5. Multi Device I^2C Reset Configuration Hardware

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

When the MSC8157E 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. .

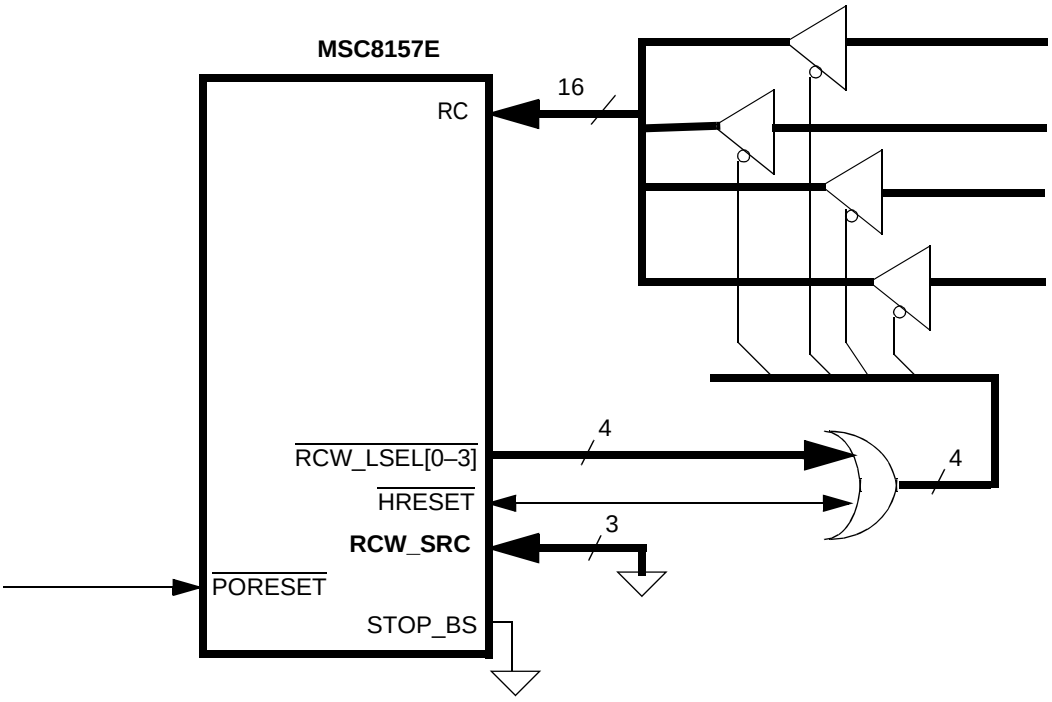


Figure 5-6. Multiplexed External Pins Reset Configuration

Figure 5-2 on page 5-5 shows the timing of the gating signals and indicates which RCW bits are loaded by each lane signal. See Table 3-4 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 MSC8157E device is configured to use the reduced RCW, the MSC8157E 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**.

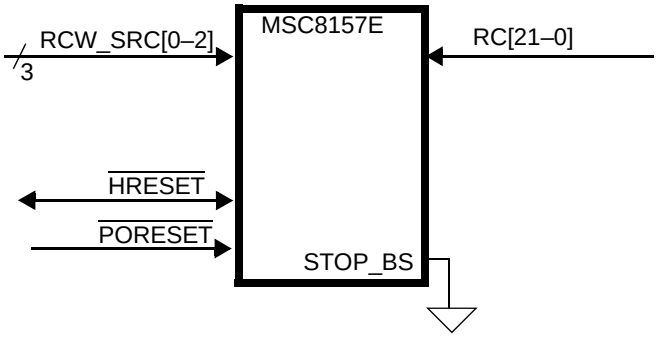


Figure 5-7. External Pins Reduced Reset Configuration

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.
28–22	SP	0, RC[21–18], RC4, RC17	SP. See **Table 5-9** for details.
21–20	R1FREQ	RC[16–15]	Rapid IO Controller 1 frequency
19	—	0	Reserved. Should be cleared.
18	PFREQ	0	PCI Express Controller frequency
17	SCLK1	RC14	SerDes PLL2 reference clock. • 0 selects 100 Mhz (SRIO/PCI Express/SGMII) • 1 selects 125 MHz (SRIO/PCI Express/SGMII)/122.88 MHz (CPRI). See **Table 5-9** for details and limitations.
16	SCLK0	RC14	SerDes PLL1 reference clock. • 0 selects 100 Mhz (SRIO/PCI Express/SGMII) • 1 selects 125 MHz (SRIO/PCI Express/SGMII). See **Table 5-9** for details and limitations.
15-13	CFREQ	000	CPRI Controllers frequency
12–11	R2FREQ	RC [16-15]	Rapid IO Controller 2 frequency
10	P3V	0	PLL3 visibility
9-6	—	0000	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	—	00	Reserved. Should be cleared.
30	RC	0	PCI Express End-Point/Root Complex Select
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 5-12**.
23	RIO	1	RapidIO access enabled.
22	RPT	RC6	RapidIO pass-through enable bit.
21	RHE	0	RapidIO host mode disabled.
20	SBETH		Enable Simple Boot Over Ethernet
19	—	0	Reserved. Should be cleared.
18	RM	0	Not reset master.
17	BP	0	Boot patch disabled.
16–11	—	0000000	Reserved. Should be cleared.
10	R1A	0	RapidIO Controller 1 does not accept all.
9	R2A	0	RapidIO Controller 2 does not accept all.
8–3	DEVID	00, RC[3–0]	DEVID[5–4] = 00, DEVID[3–0] = RC[3–0]
2	RIO2EB	0	Enable RapidIO Controller 2 Enumeration Boundary Reached
1	RIO1EB	0	Enable RapidIO Controller 1 Enumeration Boundary Reached
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 MSC8157E 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.
28–22	SP	0110000	SP. See **Table 5-9** for details.

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

Bits	Name	Value	Meaning
21–20	R1FREQ	00	Rapid IO Controller 1 frequency
19	—	0	Reserved. Should be cleared.
18	PFREQ	0	PCI Express Controller frequency
17	SCLK1	RC14	SerDes PLL2 reference clock. • 0 selects 100 Mhz (SRIO/PCI Express/SGMII) • 1 selects 125 MHz (SRIO/PCI Express/SGMII)/122.88 MHz (CPRI). See **Table 5-9** for details and limitations.
16	SCLK0	RC14	SerDes PLL1 reference clock. • 0 selects 100 Mhz (SRIO/PCI Express/SGMII) • 1 selects 125 MHz (SRIO/PCI Express/SGMII). See **Table 5-9** for details and limitations.
15-13	CFREQ	000	CPRI Controllers frequency
12–11	R2FREQ	00	Rapid IO Controller 2 frequency
10	P3V	0	PLL3 visibility
9-6	—	0000	Reserved. Should be cleared
5–0	MODCK	000000 101000	Clock Mode

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	—	0	Reserved. Should be cleared.
30	RC	0	PCI Express End-Point/Root Complex Select
29	EWDT	0	Disable Watch Dog Timers
28	PRDY	0	PCI Express not ready.
27–24	BPRT	0000	See for details **Table 5-12**.
23	RIO	1	RapidIO access is enabled.
22	RPT	0	RapidIO pass-through is disabled.
21	RHE	0	RapidIO host mode is disabled.
20	SBETH	0	Enable Simple Boot Over Ethernet
19	—	0	Reserved. Should be cleared.
18	RM	0	Not reset master
17	BP	0	Boot patch disabled.
16–11	—	000000	Reserved. Should be cleared
10	R1A	0	RapidIO Controller 1 does not accept all.
9	R2A	0	RapidIO Controller 2 does not accept all.
8-3	DEVID	000000	DEVID[5–4] = 00, DEVID[3–0] = RC[3–0]
2	RIO2EB	0	Enable RapidIO Controller 2 Enumeration Boundary Reached

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

Bits	Name	Value	Meaning
1	RIO1EB	0	Enable RapidIO Controller 1 Enumeration Boundary Reached
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 5-16**.
- Reset Configuration Word High Register (RCWHR), **page 5-20**.
- 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		—	SP							R1FREQ		—	PFREQ	SCLK1	SCLK0
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
	CFREQ			R2FREQ		P3V	—				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

Note: In the event that an unsupported protocol or invalid configuration is entered, the protocol defaults to a hard-coded option (SP = 0110000).

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.	Used with P3V to determine source. See **Table 5-10** for details.
— 29	0	Reserved. Write to one for future compatibility.	
SP 28–22	0	**SerDes Protocol** Selects the SerDes protocols to use.	See **Table 5-11** for setting descriptions.s

Table 5-9. RCWLR Bit Descriptions (Continued)

Name	Reset	Description	Settings
R1FREQ 21–20	0	**SRIO1 Frequency** Selects the SRIO1 frequency.	00 5 GHz. 01 3.125 GHz 10 2.5 GHz 11 1.25 GHz
— 19	0	Reserved. Write to one for future compatibility.	
PFREQ 18	0	**PCI Express Frequency** Selects the PCI Express frequency	0 5 GHz 1 2.5 GHz
SCLK1 17	0	**SerDes PLL2 Reference Clock** Selects the SerDes PLL2 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 except for CPRI; 122.88 MHz works for all frequencies of CPRI.	0 SerDes reference clock = 100 MHz (SRIO/PCI Express/SGMII). 1 SerDes reference clock = 125 MHz.(SRIO/PCI Express/SGMII)/122.88 MHz for CPRI,
SCLK0 16	0	**SerDes PLL1 Reference Clock** Selects the SerDes PLL1 reference clock. 100 MHz clock can work for all protocols and frequencies except for 3.125 Gbaud RapidIO and CPRI; 125 MHz works for all protocols and frequencies except for CPRI.	0 SerDes reference clock = 100 MHz (SRIO/PCI Express/SGMII). 1 SerDes reference clock = 125 MHz (SRIO/PCI Express/SGMII).
CFREQ 15–13	0	**CPRI Frequency** Selects the CPRI frequency.	000 6.1440 GHz 001 4.9152 GHz 010 3.0720 GHz 011 2.4576 GHz 100 1.2288 GHz. 101 reserved 110 reserved 111 Power down.
R2FREQ 12–11	0	**SRIO2 Frequency** Selects the SRIO2 frequency.	00 5 GHz. 01 3.125 GHz 10 2.5 GHz 11 1.25 GHz
P3V 10	0	**DDRPLL Selection** Controls use of the DDRPLL with CLKOUT.	See **Table 5-10** for details.
— 9–6	0	Reserved. Write to one for future compatibility.	
MODCK 5–0	0	**Clock Mode** Defines the clock operating mode.	See **Chapter 7**, *Clocks*.

Table 5-10. CLKO Selection

RCWLR[CLKO]	RCWLR[P3V]	CLKOUT Source
00	0	PLL0 output / 16
01	0	PLL1 output / 10
10	0	PLL2 output / 10
11	0	always low
00	1	DDRPLL output / 8
01	1	reserved

Table 5-10. CLKO Selection

RCWLR[CLKO]	RCWLR[P3V]	CLKOUT Source
10	1	reserved
11	1	reserved

Table 5-11. MSC8157E HSSI Multiplexing

RCWLR[SP] Protocol Select	Mode	SerDes Lanes										Exceptions
		A	B	C	D	E	F	G	H	I	J	
0000000	0	—	—	—	—	—	—	—	—	—	—	None
0000001	1	PEX(0)	PEX(1)	PEX(2)	PEX(3)	CPRI #6	CPRI #5	CPRI #4	CPRI #3	CPRI #2	CPRI #1	None
0000101	2	PEX(0)	PEX(1)	SGMII #1	SGMII #2	CPRI #6	CPRI #5	CPRI #4	CPRI #3	CPRI #2	CPRI #1	None
0000110	3	PEX(0)	RIO #2(1)	RIO #2(0)	SGMII #2	CPRI #6	CPRI #5	CPRI #4	CPRI #3	CPRI #2	CPRI #1	1, 2
0000111	4	PEX(0)	RIO #2(0)	SGMII #1	SGMII #2	CPRI #6	CPRI #5	CPRI #4	CPRI #3	CPRI #2	CPRI #1	1, 3
0001010	6	RIO #2(0)	RIO #2(1)	RIO #2(2)	RIO #2(3)	CPRI #6	CPRI #5	CPRI #4	CPRI #3	CPRI #2	CPRI #1	None
0001011	7	RIO #2(0)	RIO #2(1)	SGMII #1	SGMII #2	CPRI #6	CPRI #5	CPRI #4	CPRI #3	CPRI #2	CPRI #1	1, 3
0001100	8	RIO #2(0)	RIO #2(1)	SGMII #1	RIO #1(0)	CPRI #6	CPRI #5	CPRI #4	CPRI #3	CPRI #2	CPRI #1	1, 2, 3
0001101	9	RIO #2(0)	RIO #2(1)	RIO #1(1)	RIO #1(0)	CPRI #6	CPRI #5	CPRI #4	CPRI #3	CPRI #2	CPRI #1	2
0001110	10	RIO #2(0)	SGMII #2	SGMII #1	RIO #1(0)	CPRI #6	CPRI #5	CPRI #4	CPRI #3	CPRI #2	CPRI #1	1, 2, 3
0010001	11	PEX(0)	PEX(1)	PEX(2)	PEX(3)	SGMII #1	SGMII #2	CPRI #4	CPRI #3	CPRI #2	CPRI #1	None
0010011	12	PEX(0)	PEX(1)	RIO #2(0)	RIO #2(1)	SGMII #1	SGMII #2	CPRI #4	CPRI #3	CPRI #2	CPRI #1	1, 3
0010100	13	PEX(0)	PEX(1)	RIO #2(0)	RIO #2(1)	RIO #1(0)	RIO #1(1)	CPRI #4	CPRI #3	CPRI #2	CPRI #1	1, 2, 3
0010110	14	PEX(0)	PEX(1)	RIO #1(0)	RIO #1(1)	RIO #1(2)	RIO #1(3)	CPRI #4	CPRI #3	CPRI #2	CPRI #1	1, 3
0011101	15	RIO #2(0)	RIO #2(1)	RIO #2(2)	RIO #2(3)	RIO #1(0)	SGMII #2	CPRI #4	CPRI #3	CPRI #2	CPRI #1	1, 2, 3
0011110	16	RIO #2(0)	RIO #2(1)	RIO #2(2)	RIO #2(3)	RIO #1(0)	RIO #1(1)	CPRI #4	CPRI #3	CPRI #2	CPRI #1	2
0011111	17	RIO #2(0)	RIO #2(1)	SGMII #1	SGMII #2	RIO #1(0)	RIO #1(1)	CPRI #4	CPRI #3	CPRI #2	CPRI #1	1, 2, 3
0100011	18	PEX(0)	PEX(1)	PEX(2)	PEX(3)	SGMII #1	SGMII #2	RIO #1(0)	RIO #1(1)	CPRI #2	CPRI #1	1, 3

Exceptions

1. RapidIO interface cannot run at 3.125 GHz.
2. If 3.125 GHz RapidIO interface is required, both RapidIO controllers must run at the same frequency.
3. Boot from RapidIO interface at 3.125 GHz is not supported.
4. Boot from RapidIO interface at frequencies other than 3.125 GHz is not supported.

Table 5-11. MSC8157E HSSI Multiplexing (Continued)

RCWLR[SP] Protocol Select	Mode	SerDes Lanes										Exceptions
		A	B	C	D	E	F	G	H	I	J	
0100100	19	PEX(0)	PEX(1)	PEX(2)	PEX(3)	RIO #1(0)	RIO #1(1)	RIO #1(2)	RIO #1(3)	CPRI #2	CPRI #1	1, 3
0100101	20	PEX(0)	PEX(1)	SGMII #1	SGMII #2	RIO #2(1)	RIO #2(0)	RIO #1(0)	RIO #1(1)	CPRI #2	CPRI #1	1, 2, 3
0101000	21	RIO #2(0)	RIO #2(1)	RIO #2(2)	RIO #2(3)	SGMII #1	SGMII #2	CPRI #4	CPRI #3	CPRI #2	CPRI #1	1, 3
0110000	23	PEX(0)	PEX(1)	SGMII #1	SGMII #2	RIO #1(0)	RIO #1(1)	RIO #1(2)	RIO #1(3)	CPRI #2	CPRI #1	1, 3
0111101	24	RIO #2(0)	RIO #2(1)	RIO #2(2)	RIO #2(3)	SGMII #1	SGMII #2	RIO #1(0)	RIO #1(1)	CPRI #2	CPRI #1	1, 2, 3
1000001	25	RIO #2(0)	RIO #2(1)	RIO #2(2)	RIO #2(3)	RIO #1(0)	RIO #1(1)	RIO #1(2)	RIO #1(3)	CPRI #2	CPRI #1	2
1000010	26	PEX(0)	PEX(1)	PEX(2)	PEX(3)	RIO #2(1)	RIO #2(0)	RIO #1(0)	RIO #1(1)	RIO #1(2)	RIO #1(3)	1, 2, 3
1000101	27	PEX(0)	PEX(1)	PEX(2)	PEX(3)	SGMII #1	SGMII #2	RIO #2(1)	RIO #2(0)	RIO #1(0)	RIO #1(1)	1, 2, 3
1010001	28	PEX(0)	PEX(1)	PEX(2)	PEX(3)	SGMII #1	RIO #2(0)	RIO #1(0)	RIO #1(1)	RIO #1(2)	RIO #1(3)	1, 2, 3
1010011	29	PEX(0)	PEX(1)	PEX(2)	PEX(3)	SGMII #1	SGMII #2	RIO #1(0)	RIO #1(1)	RIO #1(2)	RIO #1(3)	1, 3
1010100	30	PEX(0)	PEX(1)	PEX(2)	PEX(3)	SGMII #1	SGMII #2	RIO #1(0)	RIO #1(1)	RIO #1(2)	RIO #1(3)	1, 4
1010101	31	PEX(0)	PEX(1)	SGMII #1	SGMII #2	RIO #2(1)	RIO #2(0)	RIO #1(0)	RIO #1(1)	RIO #1(2)	RIO #1(3)	1, 3
1100110	32	RIO #2(0)	RIO #2(1)	RIO #2(2)	RIO #2(3)	RIO #1(0)	RIO #1(1)	RIO #1(2)	RIO #1(3)	SGMII #1	SGMII #2	1, 2, 3

Note: The frequency of each protocol is configured in the RCWLR as shown below. Refer to the reference manual for more details on the protocol frequency.

- RCWLR[R1FREQ] selects the frequency for the RapidIO port 1
- RCWLR[R2FREQ] selects the frequency for the RapidIO port 2
- RCWLR[PFREQ] selects the frequency for the PCI Express
- RCWLR[CFREQ] selects the frequency for the CPRI
- RCWLR[SCLK1, SCLK2] select the SerDes reference clock

Exceptions	
	1. RapidIO interface cannot run at 3.125 GHz. 2. If 3.125 GHz RapidIO interface is required, both RapidIO controllers must run at the same frequency. 3. Boot from RapidIO interface at 3.125 GHz is not supported. 4. Boot from RapidIO interface at frequencies other than 3.125 GHz is not supported.

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
	—					R1A	R2A	DEVID						RIO1 EB	RIO2 EB	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 5-12** defines the RCWHR bit fields.

Table 5-12. 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 **Chapter 17**, *PCI Express Controller* 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.
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 the RCWLR[SP] bit.	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 Reserved. 1001 SGMII2 without I^2C 1010 Reserved 1011 SGMII2 with I^2C 1100–1111 Reserved.

Table 5-12. RCWHR Bit Descriptions (Continued)

Name	Description	Settings
RIO 23	**RapidIO Host Access Enable** Enables RapidIO host access to internal memory after boot.	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**, *Simple Ethernet Boot*, on page 6-18 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.	
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.
— 16–11	Reserved. Write to zero for future compatibility.	
R1A 10	**RapidIO Controller 1 Accept All** Selects the reset value of P0PTAACR[AA]. When set, RapidIO controller 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 Controller 2 Accept All** Selects the reset value of P1PTAACR[AA]. When set, RapidIO controller 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).
RIO1EB 2	**Enable RapidIO Controller 1 Enumeration Boundary Reached** Enables/disables the enumeration boundary reached.	0 Enumeration Boundary Reached not enabled. 1 Enumeration Boundary Reached enabled.

Table 5-12. RCWHR Bit Descriptions (Continued)

Name	Description	Settings
RIO2EB 1	**Enable RapidIO Controller2 Enumeration Boundary Reached** Enables/disables the enumeration boundary reached.	0 Enumeration Boundary Reached not enabled. 1 Enumeration Boundary Reached enabled.
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	—	SWHR	RM	JPO	JH	—					RIO2	RIO1	RS	—
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 5-13** defines the RSR bit fields.

Table 5-13. 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 Reserved. 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.

Table 5-13. RSR Bit Descriptions (Continued)

Name	Reset	Description	Settings
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.	
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.
— 13	0	Reserved. Write to zero for future compatibility.	
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[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.
— 8–4	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.
RS 1	0	**Reset Status** When an external or internal reset event is detected, RS is set.	0 No reset event. 1 A reset event was detected.
— 0	0	Reserved. Write to zero for future compatibility.	

Note: The RSR accumulates reset events. For example, because a software watchdog timer expiration results in a hard reset, RSR[RS] is 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 5-14** defines the bit fields of RPR.

Table 5-14. 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
	—														SWHR	—
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 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 5-15** defines the RCR bit fields.

Table 5-15. RCR Bit Descriptions

Name	Reset	Description	Settings
— 31–2	0	Reserved. Write to zero for future compatibility.	
SWHR 1	0	**Software Hard Reset** Setting this bit cause the MSC8157E 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.
— 0	0	Reserved. Write to zero for future compatibility.	

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 5-16** defines the RCER bit fields.

Table 5-16. 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.

6 Boot Program

The boot program initializes the MSC8157E after it completes a reset sequence. The MSC8157E 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 0xC0000000–0xC00063FF (25 KB). Do not write to this area during boot loading. Always use at least an 8 Kbyte EEPROM for boot.

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 0xC0000004 in M3 memory (see **Section 6.5**, *Boot Errors*, on page 6-24 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 five 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.* The boot can load additional code from I^2C EEPROM.
- *Boot mode select (core 0)*. This part includes downloading of code from one of the MSC8157E bootable ports as defined by the RCWHR[BPRT] field.
- *Boot completion*. All cores complete the boot operation and jump to a user-specified address.

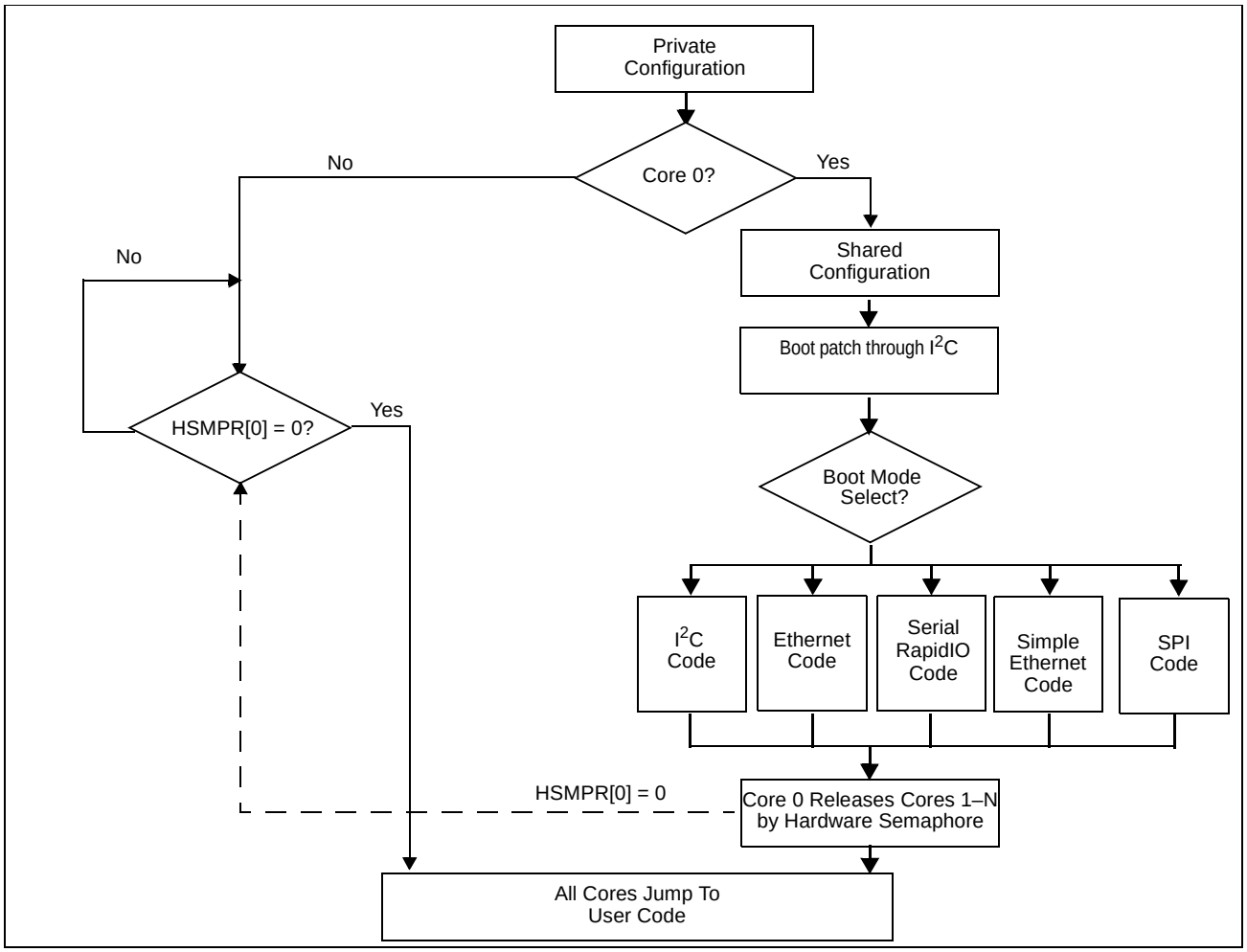


Figure 6-1. Boot Sequence Diagram

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 *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:

- In case of power down of M3 (upper 2MB) the boot code closes the appropriate class windows towards the relevant power domain.
- 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 L[A–J]GCR1 are set in order to disable the tri-state on the Serial RapidIO lanes based on RCWLR[SP].
  - The necessary bits of P0CCSR and P1CCSR (RapidIO) are set to enable the port.
- QUICC Engine module priority is set to be 1 with emergency not masked (that is SDMR[EB1_PR] = 01).

6.1.3 Patch Mode

- The Patch mode is enabled if RCWHR[BP] is set. See **Chapter 5**, *Reset* for details.
- Boot loads patch code from I^2C and executes in the same way like boot over I^2C.
- After the patch is executed, the boot code continues to load boot code from the boot port which is defined by RCWHR[BPRT].
- If the 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 MSC8157E can share the I^2C EEPROM device with other MSC8157E 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:

- For each EEPROM in the system, there must be at least one EEPROM master. The EEPROM master is also the reset master (RCWHR[RM])
- For each EEPROM, there can be 0 or more EEPROM slaves. An EEPROM slave is defined as a device that reads its 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 0x96 of the EEPROM.
- 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 0x18 of the EEPROM.
- Every EEPROM slave must also be a reset slave.
- There may be up to 15 reset slaves per EEPROM
- As a consequence of the conditions listed in 1–5, the limitations on the number of slaves is defined as $0 \le$ #EEPROM slaves $\le$ #reset slaves ≤ 15
- 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).

- EEPROM slaves must be numbered sequentially from 0 upward.
- 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.
- *For the case of multi device RCW only*: The EEPROM master can have its $\overline{\text{HRESET}}$ asserted without the reset slaves being in reset as well. Each reset slave may have its $\overline{\text{HRESET}}$ asserted without the master being reset
- *For the case of multi device RCW And boot using I^2C*: If the EEPROM master have its $\overline{\text{HRESET}}$ asserted, the EEPROM slaves must have their $\overline{\text{HRESET}}$ asserted as well. The EEPROM slaves can have their $\overline{\text{HRESET}}$ asserted without the master being reset. However, there must be an external logic that toggles the EEPROM slaves STOP_BS as the master performs during its boot sequence (The logic may be implemented using an FPGA or other implementation).
- 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 0x18 or 0x96 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 MSC8157E #0 is a reset master and MSC8157E #1 is a reset slave. The reset master uses {GPIO[0–3], GPIO[21]} to release the reset slaves. The MSC8157E 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.

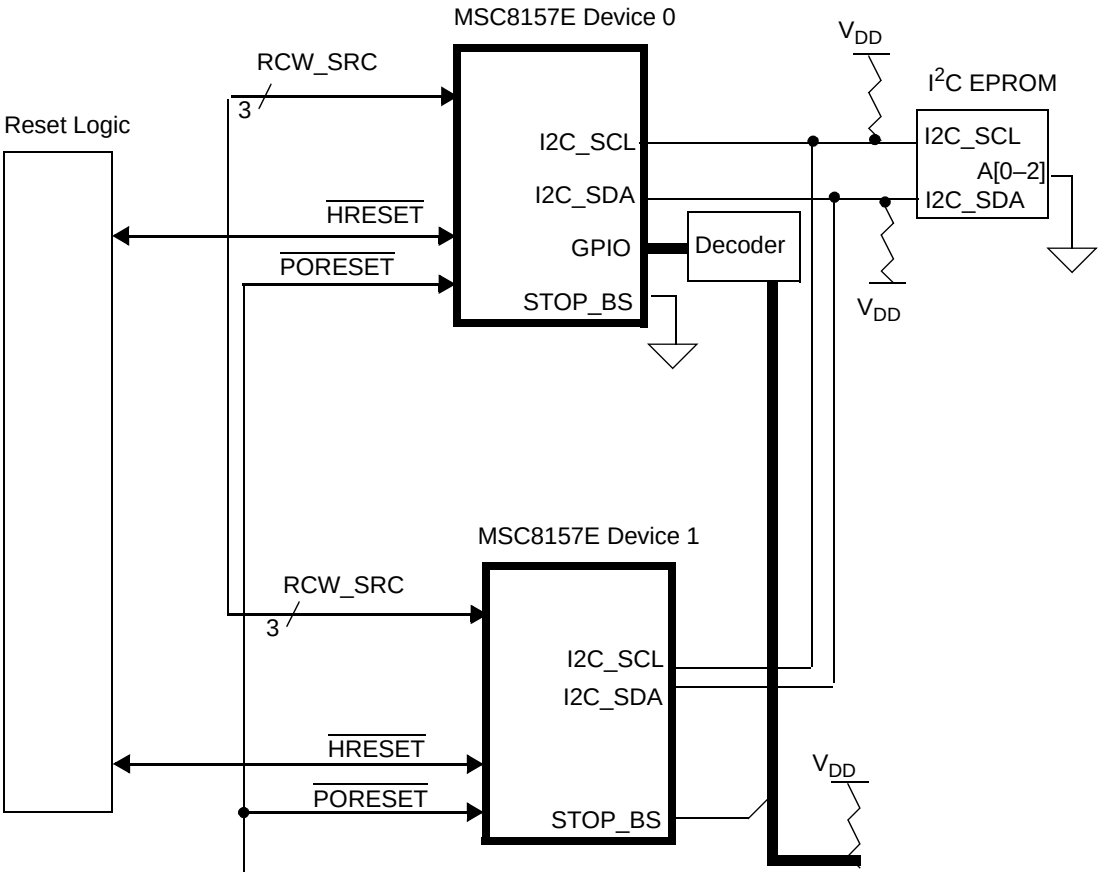


Figure 6-2. I^2C Multi Device System

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

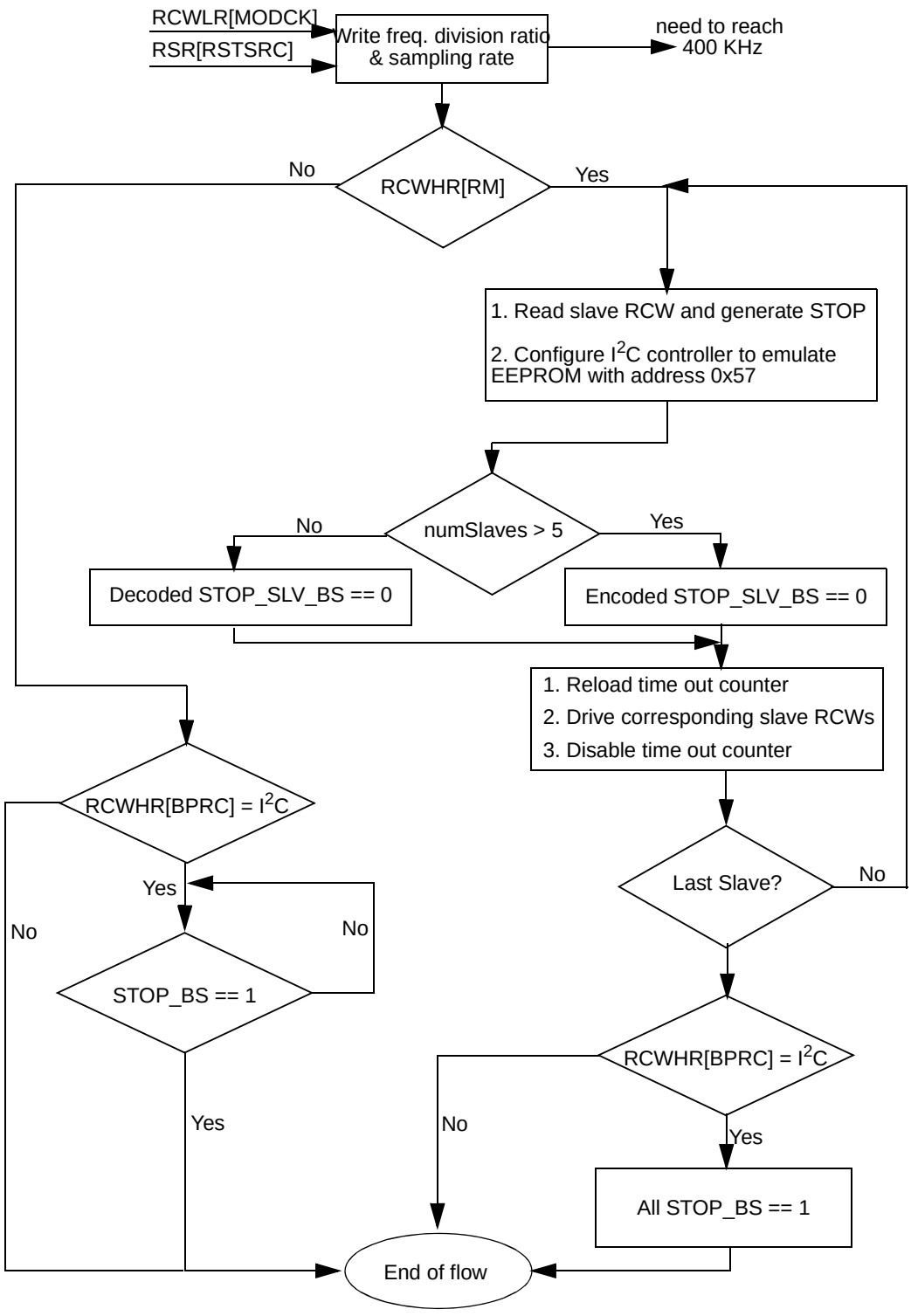


Figure 6-3. I^{2}C initialization and Multi Device Support

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

1. The MSC8157E reads RSR register to determine if the reset is PORESET. If it is not PORESET, this section of the boot is bypassed entirely.
2. The MSC8157E 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 MSC8157E 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 place in the I2C EEPROM (with address 0x50) and store them in M3.
   c. The reset master's 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). From this stage the reset slave starts reading its RCW from the reset master (that emulates an I^2C EEPROM with address 0x57)
   e. The reset master:
      Drive preamble 0xAA55AA
      Drive header 0xFFFFFF
      Drive RCWLR
      Drive header 0xFFFFFF
      Drive RCWHR
      Drive header 0x000000000000FF
      For the read request of the reset slave after which the reset slave generates a STOP condition on the bus.
   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 RCW the reset slave starts to run the boot program.
   If the MSC8157E is also an EEPROM slave, the boot waits until $\overline{\text{STOP\_BS}}$ is 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.

Note: If RSR[RSTSRC] is 0b100 or 0b101 (Hard coded RCW) than the RCWHR[BPRT] field is ignored and the boot port is *Serial RapidIO Interface without I²C support*. The RCWLR[SP] is 0b0110000 (x4 SRIO1)

6.2.1 I²C EEPROM

The MSC8157E boot expects the I²C EEPROM to be divided in to four sections:

- 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.
- Reserved.
- Boot configuration. This section starts at address 0x0097 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 in the device. This feature is most commonly used to configure registers 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.
- Boot code. This section starts at address 0x0218 of the EEPROM and contains the user code. The value in the Destination Address field must be 32-bit aligned.

Note: Although not all sections are used by all boot port options, the section addresses are fixed. However, if an I²C EEPROM is required in the system for any reason (RCW, configuration, and so on), addresses 0x000–0x216 must be valid.

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 ($\lfloor (64 \cdot 6)/(2 \cdot 4) \rfloor = 47 + 1$)

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	0	0	0	0	0	0	0	0	Preload Command
0x0012	0	0	0	0	0	0	0	0	
0x0013	0	0	0	0	0	0	0	0	
0x0014	0	0	0	0	0	0	0	0	
0x0015	0	0	0	0	0	0	0	0	
0x0016	0	0	0	0	0	0	0	0	
0x0017	1	1	1	1	1	1	1	1	
0x0018	$numResetSlaves \in [0-15]$								Number of Reset Slaves
0x0019	Reset Configuration Word Low [31–24]								Reset Configuration Word Low of Slave 1
0x001A	Reset Configuration Word Low [23–16]								
0x001B	Reset Configuration Word Low [15–8]								
0x001C	Reset Configuration Word Low [7–0]								
0x001D	Reset Configuration Word High [31–24]								Reset Configuration Word High of Slave 1
0x001E	Reset Configuration Word High [23–16]								
0x001F	Reset Configuration Word High [15–8]								
0x0020	Reset Configuration Word High [7–0]								
	.....								
	.....								

Figure 6-4. EEPROM Contents

Address	0	1	2	3	4	5	6	7	Description
0x0089	Reset Configuration Word Low [31–24]								**Reset Configuration Word Low of Slave 15**
0x008A	Reset Configuration Word Low [23–16]								
0x008B	Reset Configuration Word Low [15–8]								
0x008C	Reset Configuration Word Low [7–0]								
0x008D	Reset Configuration Word High [31–24]								**Reset Configuration Word High of Slave 15**
0x008E	Reset Configuration Word High [23–16]								
0x008F	Reset Configuration Word High [15–8]								
0x0090	Reset Configuration Word High [7–0]								
0x0091	0	0	0	0	0	0	0	0	**Reserved**
0x0092	0	0	0	0	0	0	0	0	
0x0093	0	0	0	0	0	0	0	0	
0x0094	0	0	0	0	0	0	0	0	
0x0095	0	0	0	0	0	0	0	0	
0x0096	0 ≤ numEEPROMSlaves ≤ numResetslaves ≤ 15								**Number of EEPROM Slaves**
0x0097									**Configuration/MAC Address First Byte**
0x0216									**Configuration Last Byte/ Device 0x3F Last MAC Address Byte**
0x0217	0	0	0	0	0	0	0	0	**Reserved**
0x0218	1	0	1	1	1	1	1	1	**Block Control (Block #0)**
0x0219	size[0–7]								**Block Size**
0x021A	size[8–15]								
0x021B	size[16–23]								
0x021C	NBA[31–24]								**Next Block Address**
0x021D	NBA[23–16]								
0x021E	NBA[15–8]								
0x021F	NBA[7–0]								
0x0220	DA[31–24]								**Destination Address**
0x0221	DA[23–16]								
0x0222	DA[15–8]								
0x0223	DA[7–0]								
	CODE								**Payload Data**

Figure 6-4. EEPROM Contents (Continued)

Address	0	1	2	3	4	5	6	7	Description
	Checksum[15–8]								**Checksum and $\overline{\text{Checksum}}$**
	Checksum[7–0]								
	$\overline{\text{Checksum}}$[15–8]								
	$\overline{\text{Checksum}}$[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 msb of the I^2C slave address are always 1010000. The I^2C controller expects a specific memory image when trying to read data during the EEPROM as shown in **Figure 6-5**.

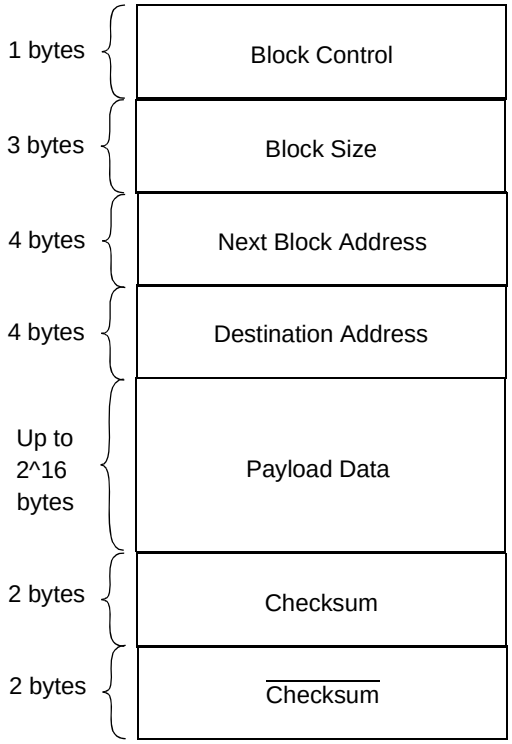


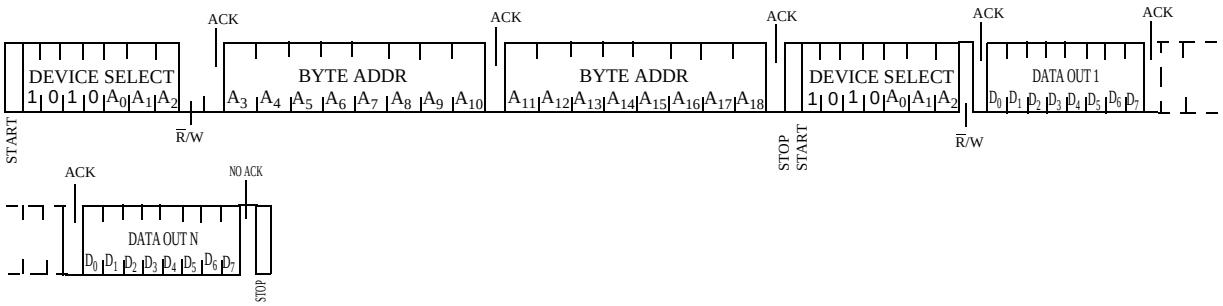
Figure 6-5. EEPROM Data Format

The I^2C memory image consists of:

- *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.
- *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}).
- *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.
- *Destination Address*. The address to which the payload data should be written.
- *Payload Data***.** Holds $1<=size<2^{16}$ bytes (up to 64 Kbytes) of data to be written to on-device memory according to the destination address.
- *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.
- $\overline{Checksum}$**.** A 2-byte field that holds bitwise-not of the Checksum.

The I^2C bootloader expects the 4 bytes of Checksum and $\overline{Checksum}$ regardless of the CSE value. If the Checksum is disabled, these 4 bytes are not checked. By using Checksum and $\overline{Checksum}$, the boot ensures that all values of the bits are real values and that there are no stuck signals. If both Checksum and $\overline{Checksum}$ 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} = 3'b000) 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 MSC8157E. After the ninth unacknowledged bit, the boot code generates a STOP condition. **Figure 6-6** describes the Software I^2C read access.



Note: A_0 and D_0 are the most significant bits.

Figure 6-6. I^2C Read Access

6.2.2 Ethernet

The MSC8157E 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 UEC0 (UCC1) and UEC1 (UCC3) depending on the values of RCWHR[BPRT] and RCWLR[SP]. For boot over Ethernet in SGMII mode the RCWLR[SP] field must have a valid SGMII option (See **Chapter 5**, *Reset* for details). Booting over Ethernet in RGMII mode is enabled on UEC0 (UCC1) only. **Figure 6-7** describes the Ethernet bootloader flow.

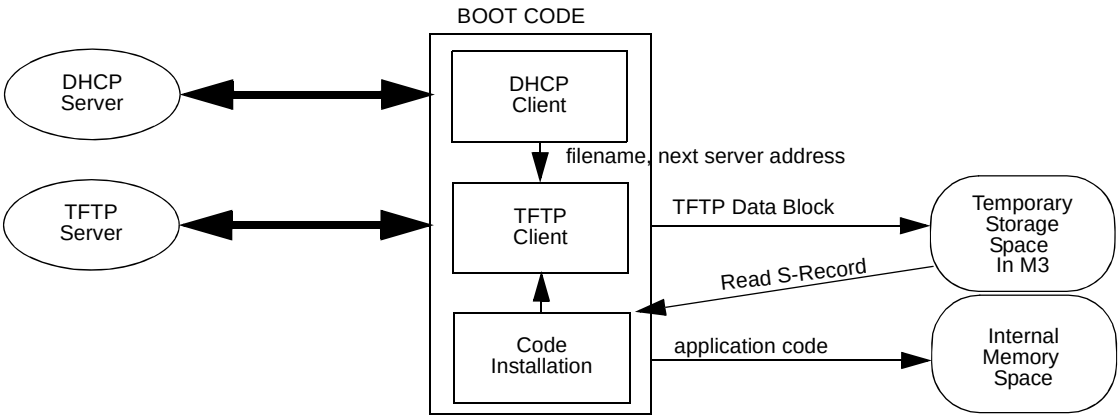


Figure 6-7. Ethernet Bootloader Flow

: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 6-8**.

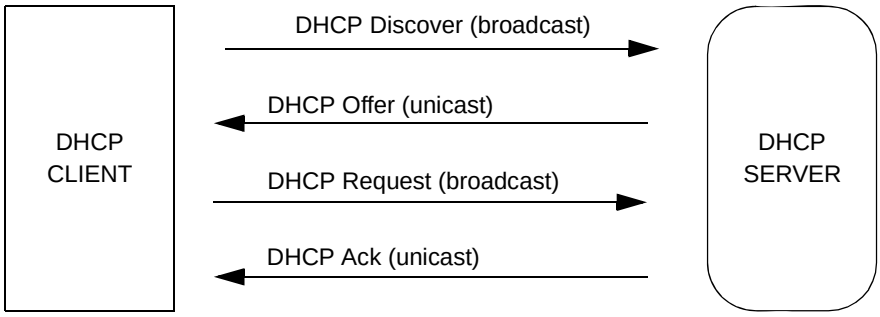


Figure 6-8. DHCP Transactions

- 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 MSC8157E 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.

6.2.2.2 TFTP Client

This implementation supports three types of messages: TFTP REQUEST, TFTP DATA and TFTP ACK. **Figure 6-9** describes the TFTP handshake.

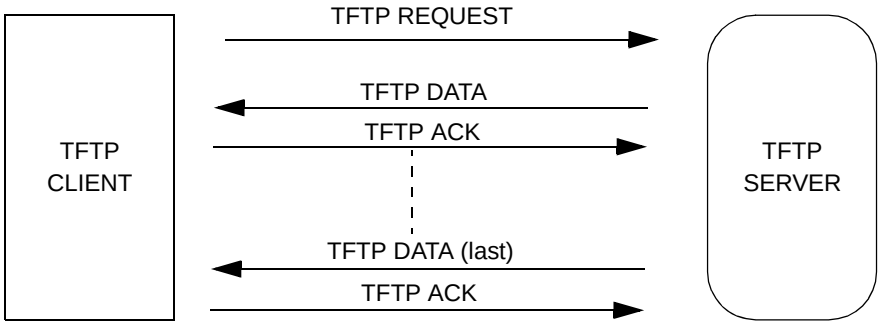


Figure 6-9. TFTP Transactions

- 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 Ethernet bootloader supports up to 128 Mbytes of S-Record file size. The S-Record file structure is described in **Figure 6-10**.

Figure 6-10. S-Record File Structure

Each line of an S-Record file corresponds to any of 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 and no 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
S70500000000FA

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 0xC0000010).
- **FA**. Checksum field.

Each S-Record line includes an address field that maps the lines data content to a memory location in MSC8157E. Core 0 moves the data to this address.

Note: Because the MSC8157E uses 32-bit addressing, use of S3 and S7 is recommended.

6.2.3 Simple Ethernet Boot

The MSC8157E supports a simple Ethernet boot mode. This mode is selected by setting the RCWHR[SBETH] bit during the $\overline{\text{PORESET}}$ sequence (See **Chapter 5**, *Reset* for details).

In this procedure, an Ethernet master waits for the MSC8157E boot program to finish its default initialization and then initializes the device by typically loading code and data to the on device memory.

- Supports RGMII @1000 and SGMII @1000Mbps full duplex.
- Each client could have its own unique Media Access Control (MAC) address. This MAC address may be based on RCWHR[DEVID], or be user defined and read from I^2C EEPROM.

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 MSC8157E 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> <Length><2 Bytes Data_Length><4 Bytes Address> <Data>.
- 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 0xC0000000.
- All cores jump to the address written in 0xC0000010. The value in this address should be written during the boot loading process.

Figure 6-11 describes the Simple Ethernet bootloader flow

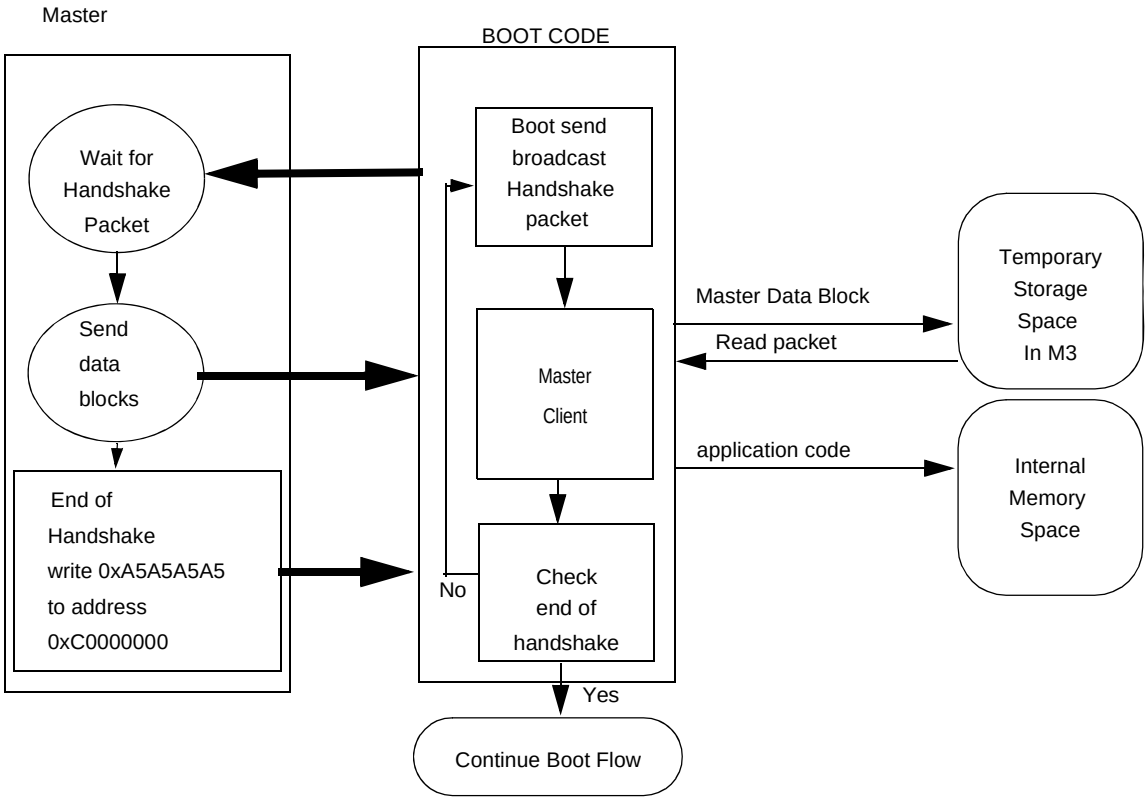


Figure 6-11. Ethernet Bootloader Flow

6.2.3.2 Simple Ethernet Boot Ports

Booting over Ethernet is enabled on UEC0 (UCC1) and UEC1 (UCC3) depending on the values of RCWHR[BPRT] and RCWLR[SP]. For simple boot over Ethernet in SGMII mode the RCWLR[SP] field must have a valid SGMII option (See **Chapter 5**, *Reset* for details).

Note: Booting over Ethernet in RGMII mode is enabled on UEC0 (UCC1) only.

6.2.3.3 Boot File Format

The Ethernet bootloader expects an application file in the format of a Simple Ethernet. The Simple Ethernet has the following format: <lengh><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	Length of data in bytes
<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 MSC8157E. Core 0 moves the data to this address.

The following code shows a typical Simple-Ethernet packet for End of handshake between Ethernet master and the MSC8157E boot:

```
1E7FD5000000
1E7FD5100000
002E
0004
C0000000
A5A5A5A5
```

The End of handshake packet which is sent by Ethernet Master consists of the following:

- **1E7FD5000000.** Destination MAC address (Default MSC8157E MAC address).
- **1E7FD5100000.** Source MAC address.
- 002E. The minimal length field according to IEEE 802.3 standard.
- 0004. Data length is 4 bytes.
- **C0000000.** The address at which the data field is to be loaded into memory. This is the Handshake address for the MSC8157E device.
- **A5A5A5A5.** Handshake data.

6.2.4 Serial RapidIO Interconnect

In this procedure a Serial RapidIO master waits for the MSC8157E 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 Interface Without I^{2}C Support

The boot code configures PxCCSR and L[A–J]GCR1 according to RCWLR[SP] and writes 0x17171717 to address 0xC0000000.
The flow is:

1. Disable port by writing to PxCCSR register.
2. Enable lanes by writing to the L[A–J]GCR1 registers.
3. Enable the port by writing to PxCCSR register.
4. Polling address 0xC0000000 until the serial RapidIO master writes 0xA5A5A5A5 to it, thereby indicating that it has finished its code loading.

Figure 6-12 describes the boot flow from Serial RapidIO interconnect**.**

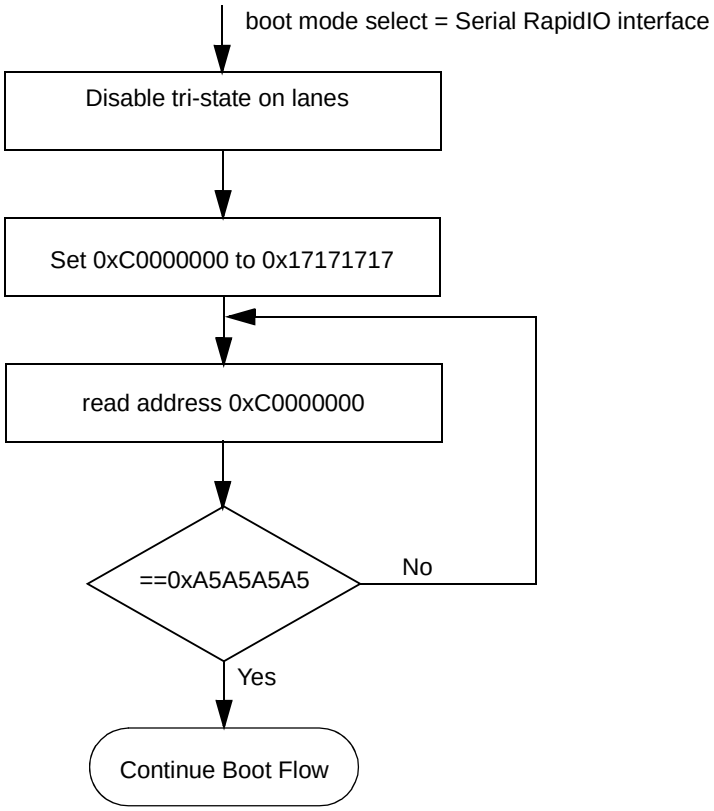


Figure 6-12. Serial RapidIO Interface Boot Flow

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 MSC8157E 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 21**, *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 CECR[RST]
- 0x900D900D is written to 0xC000000C in M3 to indicate that the boot has finished executing.
- All cores jump to the address written in 0xC0000010. 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 0xC0000004. 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 0x18 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 (0xC0000018–0xC00063FF)
0x003FEFF1	In patch mode the boot port is I^2C
0x0027EFFE	Lost arbitration on I^2C bus.
0x0027EFFD	Time-out on I^2C acknowledge (9th clock).
0x0027EFFC	Stuck I2C_SDA (I^2C bus).
0x00000000	Unexpected debug condition in the SC3850 Core (unexpected interrupt, EE0 asserted and so on)

7 Clocks

The clock circuits contain the following six PLLs:

- Three PLLs driven from a single CLKIN signal generated by a crystal-based oscillator that generates the clocks for most of the system as described in **Figure 7-1**
- One PLL driven from either from the CLKIN signal or from a dedicated MCLKIN signal that generates clocks for the DDR-SDRAM memory controller.
- Two PLLs that generate clocks for the SerDes interfaces. The selection of the individual clock frequencies for the interfaces is done by the settings of the Reset Configuration Word (see **Chapter 5**, *Reset* for details). For the CPRI lanes, the initial configuration determines the starting frequency, but the actual connection frequency is negotiated and adjusted by dividers in the CPRI framer block (see **Chapter 18**, *Common Public Radio Interface (CPRI) Complex* for details).

7.1 Clock Generation Components and Modes

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

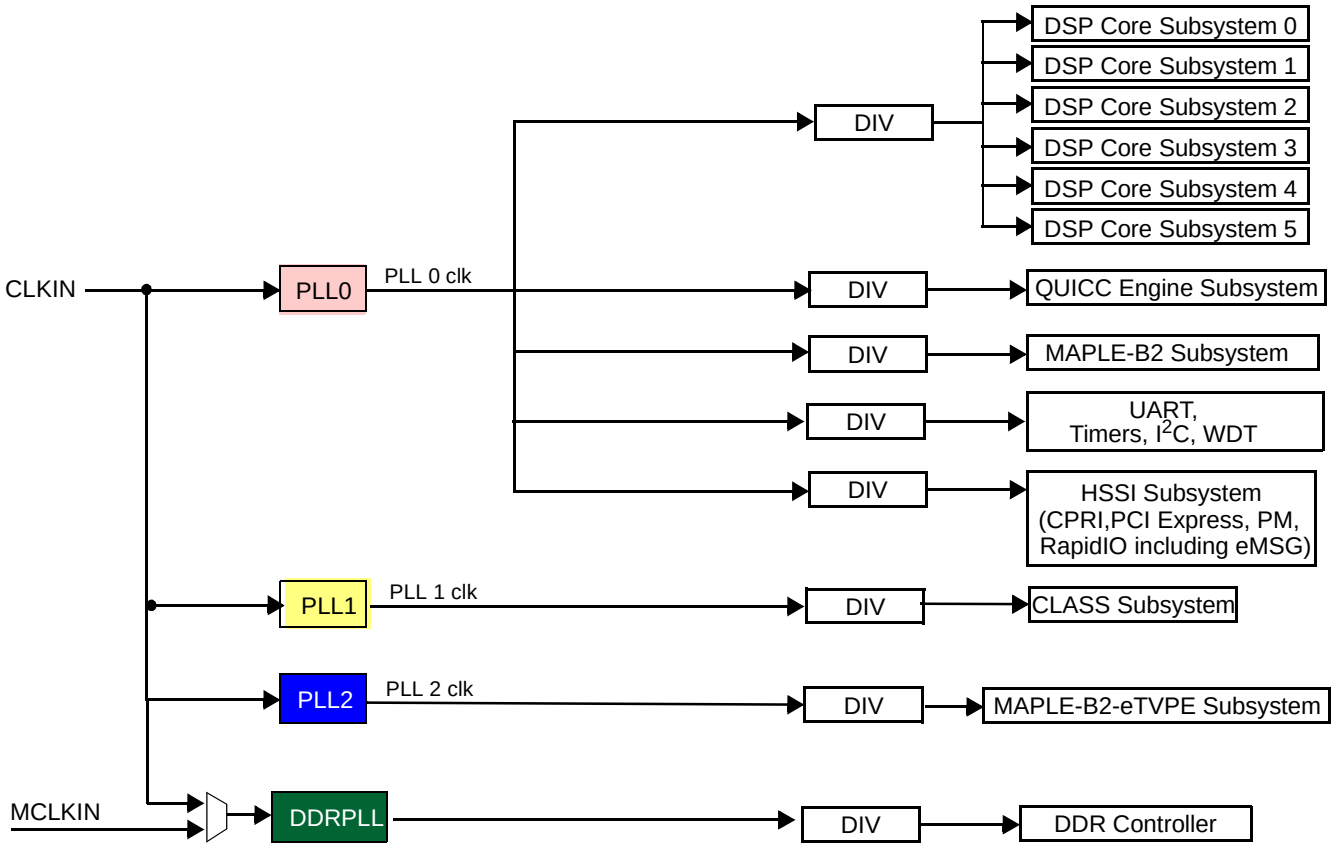


Figure 7-1. MSC8157E Clock Scheme

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 MSC8157E clock modes are listed in **Table 7-1**.

Table 7-1. MSC8157E Clock Modes

Mode	CLKIN	MCLKIN	PLL0	PLL1	PLL2	DDR PLL	CLASS	UART, Timers, I^2C, WDT	DSP Core Sub-systems	HSSI (CPRI, PCI Express, PM, RapidIO inc. eMSG)	QUICC Engine Sub-system	MAPLE-B2	eTVPE	DDR
0	133.33	n/a	2000	667	800	1333	667	500	1000	500	500	500	800	1333
40	133.33	133.33	2000	667	800	1333	667	500	1000	500	500	500	800	1333

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

2. In mode 40 the DDR PLL is fed by the MCLKIN

The CLKOUT pin can be driven from either PLL with selection determined by the value of RCWLR[CLKO] and RCWLR[P3V] (bits 31–30 and 10 of the low part of the reset configuration word—for details, see **Chapter 5**, *Reset*). The possible CLKOUT frequencies are listed in **Table 7-2**.

Table 7-2. MSC8157E CLK_OUT Frequencies

Mode	PLL0	PLL1	PLL2	DDRPLL	CLK_OUT from PLL0	CLK_OUT from PLL1	CLK_OUT from PLL2	CLKOUT2 from DDRPLL
0	2000	667	800	1333.3	125	66.7	80	166.6
40	2000	667	800	1333.3	125	66.7	80	166.6

7.2 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	PER DIS	—	HSSI DIS	QEDIS	MAPLE DIS	ETVPE DIS	TITDIS	—							
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.
PERDIS 14	0	**Peripherals Clock Domain Disable** Used to disable the Peripherals clock domain to conserve power.	0 UART, Timers, I^2C, WDT clock domain enabled. 1 UART, Timers, I^2C, WDT clock domain disabled.
13	0	Reserved. Write to zero for future compatibility.	
HSSIDIS 12	0	**HSSI Clock Domain Disable** Used to disable the HSSI clock domain to conserve power.	0 HSSI (CPRI, PCI Express, PM, RapidIO including eMSG) clock domain enabled. 1 HSSI (CPRI, PCI Express, PM, RapidIO including eMSG) 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.
ETVPDIS 9	0	**eTVPE Clock Domain Disable** Used to disable the eTVPE controller clock domain to conserve power.	0 eTVPE clock domain enabled. 1 eTVPE clock domain disabled.
TITDIS 8	0	**Core Subsystem Clock Domain Disable** Used to disable the Core Subsystem controller clock domain to conserve power.	0 Core Subsystem clock domain enabled. 1 Core Subsystem clock domain disabled.
— 7–0	0	Reserved. Write to zero for future compatibility.	

General Configuration Registers 8

The MSC8157E 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 8-40**
- General Interrupt Register 5 (GIR5), see **page 8-42**
- General Interrupt Enable Register 5 for Cores 0–5 (GIER5_[0–5]), see **page 8-44**
- General Control Register 11 (GCR11), see **page 8-46**
- General Control Register 13 (GCR13), see **page 8-47**
- General Status Register 8 (GSR8), see **page 8-48**
- DMA Request0 Control Register (GCR_DREQ0), see **page 8-49**
- DMA Request1 Control Register (GCR_DREQ1), see **page 8-53**
- DMA Done Control Register (GCR_DDONE), see **page 8-57**
- DDR Controller General Configuration Register (DDRC_GCR), see **page 8-60**
- Core Subsystem Slave Port General Configuration Register (CORE_SLV_GCR), see **page 8-62**
- QUICC Engine Input General Control Register (QE_PIO_IN_GCR), see **page 8-63**
- QUICC Engine Output General Status Register (QE_PIO_OUT_GSR), see **page 8-64**
- L2Q Arbitration Control of Cores 0 and 1 (MEX_T2_0_1_ARB), see **page 8-65**
- L2Q Arbitration Control of Cores 2 and 3 (MEX_T2_2_3_ARB), see **page 8-66**
- L2Q Arbitration Control of Cores 4 and 5 (MEX_T2_4_5_ARB), see **page 8-67**
- General Interrupt Register 6 (GIR6), see **page 8-68**
- General Interrupt Enable Register 6 for Cores 0–5 (GIER6_[0–5]), see **page 8-71**
- General Interrupt Register 7 (GIR7), see **page 8-74**
- General Interrupt Enable Register 7 for Cores 0–5 (GIER7_[0–5]), see **page 8-76**
- DDR View Through L2 Memory Core Subsystems 0–3 (L2MAP_0_3), see **page 8-79**
- DDR View Through L2 Memory Core Subsystems 4–5 (L2MAP_4_5), see **page 8-80**
- eMSG to QUICC Engine External Request Enable (CPCEER), see **page 8-81**
- Ethernet 1 High Resolution Delay Register (UCC1_DELAY_HR), see **page 8-84**
- Ethernet 2 High Resolution Delay Register (UCC3_DELAY_HR), see **page 8-86**
- General Interrupt Register 8 (GIR8), see **page 8-88**
- CPRI Interrupt to MAPLE External Request Enable (MAPLE_EXT_REQ_EN_1), see **page 8-89**

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
	MPL_TRB_SEL	PM_HSSI_EN	—		MPL_ETVPE_SYNC_BYPASS	MAPLE_SYNC_MODE_EN	—	
Type	R/W							
Reset	0	1	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
	MEX_DMM	DDRW_PIPE_LMT					DDRR_PIPE_LMT	
Type	R/W							
Reset	0	1	0	0	0	0	1	0

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

GCR1 configures various general functions for the MSC8157E device. **Table 8-1** lists the GCR1 bit field descriptions.

Table 8-1. GCR1 Bit Descriptions

Name	Reset	Description	Settings
MPL_TB_SEL 31	0	**MAPLE Trace Buffer Enable** Selects the trace buffers to output.	0 Output trace buffers 1 and 2. 1 Output trace buffers 3 and 4.
PM_HSSI_EN 30	1	**HSSI PM Enable** Enables/disables the PM signals coming out of the HSSI.	0 HSSI PM disabled. 1 HSSI PM enabled.
— 29–28	0	Reserved. Write to 0 for future compatibility.	
MPL_ETVPE_SYNC_BYPASS 27	0	**MAPLE ETVPE Sync Mode Enable** Enables/disables the ETVPE CDC logic by setting DCF sync bypass.	0 Sync mode disabled. 1 Sync mode enable.
MAPLE_SYNC_MODE_EN 26	0	**MAPLE Sync Mode Enable** Selects the signal for the internal clock multiplexer in MAPLE for MAPLE sync mode.	0 1
— 25–17	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.

Table 8-1. GCR1 Bit Descriptions (Continued)

Name	Reset	Description	Settings
MEX_DMM 15		**MBus Bridge Memory Mode** Determines the mode used by the MBus bridges between the CLASS and the core subsystems. In MSC8156 mode, the core subsystems can only access the M2 memory within its own subsystem. In MSC8157 mode, cores can access M2 memory as follows: • Core 0 can access M2 in subsystem 0, 2, 3, 4, and 5. • Core 1 can access M2 in subsystem 1, 2, 3, 4, and 5. • Core 2 can access M2 in subsystem 2, 4, and 5. • Core 3 can access M2 in subsystem 3, 4, and 5. • Core 4 can access M2 in subsystem 4. • Core 5 can access M2 in subsystem 5.	0 MSC8157 mode 1 MSC8156 mode
DDRW_PIPE_LMT 14–10	10000	**DDR Write Pipeline Limit** Specifies the DDR write pipeline depth.	
DDRR_PIPE_LMT 9–5	10000	**DDR Read Pipeline Limit** Specifies the DDR read pipeline depth.	
— 4–0	0	Reserved. Write to 0 for future compatibility.	

8.2.2 General Configuration Register 2 (GCR2)

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

Bit	31	30	29	28	27	26	25	24
	PRESCALE_OV_EN	SRIO_PRESCALE_CFG						
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 MSC8157E device. **Table 8-2** lists the GCR2 bit field descriptions.

Table 8-2. GCR2 Bit Descriptions

Name	Reset	Description	Settings
PRESCALE_OV_EN 31	0	**Prescale Configuration Source** Determines the source to use for the pre-scale configuration.	0 RCW 1 GCR
SRIO_PRESCALE_CFG 30–24	0	**Serial RapidIO Prescale Configuration** When GCR is selected as the source (GCR2[PRESCALE_OV_EN] = 1), provides the prescalers for the Serial RapidIO PHY wrapper.	
— 23–17	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 14.6.17**, *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 subsystem.	0 No debug request. 1 Debug request.
CORE4_DBG_REQ 4	0	**Core 4 Debug Request** Asserts a debug request to core 4 subsystem.	0 No debug request. 1 Debug request.
CORE3_DBG_REQ 3	0	**Core 3 Debug Request** Asserts a debug request to core 3 subsystem.	0 No debug request. 1 Debug request.
CORE2_DBG_REQ 2	0	**Core 2 Debug Request** Asserts a debug request to core 2 subsystem.	0 No debug request. 1 Debug request.
CORE1_DBG_REQ 1	0	**Core 1 Debug Request** Asserts a debug request to core 1 subsystem.	0 No debug request. 1 Debug request.
CORE0_DBG_REQ 0	0	**Core 0 Debug Request** Asserts a debug request to core 0 subsystem.	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_PSIF_PU	—			
Type	R							
Reset	0	0	0	1	0	0	1	1

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 MSC8157E 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 Second 1 MB Power Up** Reflects the M3 second 1 MB power up status.	0 Second 1 MB M3 powered down. 1 Second 1 MB KB M3 powered up.
M3_PU_0 13	0	**M3 First 1 MB Power Up** Reflects the M3 first 1 MB power up status.	0 First 1 MB M3 (1st half) powered down. 1 First 1 MB M3 (1st half) powered up.
MAPLE_PSIF_PU 12	1	**MAPLE Power Up (excluding TVPE and EQPE)** Reflects the MAPLE block power status except for TVPE and EQPE.	0 Powered down. 1 Powered up.
— 11–6	001100	Reserved. Write the reset value (0b0001100) for future compatibility.	
CORE_DBG_STS5 5	0	**Core Debug Status 5** Reflects the mode of core 5 subsystem.	0 Not in Debug mode. 1 In Debug mode.
CORE_DBG_STS4 4	0	**Core Debug Status 4** Reflects the mode of core 4 subsystem.	0 Not in Debug mode. 1 In Debug mode.
CORE_DBG_STS3 3	0	**Core Debug Status 3** Reflects the mode of core 3 subsystem.	0 Not in Debug mode. 1 In Debug mode.
CORE_DBG_STS2 2	0	**Core Debug Status 2** Reflects the mode of core 2 subsystem.	0 Not in Debug mode. 1 In Debug mode.
CORE_DBG_STS1 1	0	**Core Debug Status 1** Reflects the mode of core 1 subsystem.	0 Not in Debug mode. 1 In Debug mode.
CORE_DBG_STS0 0	0	**Core0 Debug Status 0** Reflects the mode of core 0 subsystem.	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
	HSSI_IPG_STOP_ACK	CPRI_IPG_STOP_ACK	—			DMA1_PD_STATUS	SRIO1_PD_STATUS	SE_CB_PD_STATUS
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SD_ISO_RING_POWERED	SD_FORCE_STOPPED	DMA0_PD_STATUS	SRIO0_PD_STATUS	PEX_PD_STATUS	OCN_PD_STATUS	EMSG_PD_STATUS	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	PEX_STOP_ACK	EMSG_STOP_ACK	—	SD2_RST_DONE	SERDES_ PD	SRIO_1_ IDLE	SRIO_1_OB_IDLE
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SD1_RST_DONE	—	SRIO_0_ IDLE	SRIO_0_OB_IDLE	PEX_OB_IDLE	PEX_IDLE	—	EMSG_IDLE
Type	R							
Reset	0	0	0	0	0	0	0	0

HSSI_SR controls part of the SerDes operation for the MSC8157E 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
HSSI_IPG_STOP_ACK 31	0	**HSSI Stop Acknowledge** Indicates whether the HSSI Stop request is acknowledged.	0 Not acknowledged. 1 Acknowledged.
CPRI_IPG_STOP_ACK 30	0	**CPRI Stop Acknowledge** Indicates whether the CPRI Stop request is acknowledged.	0 Not acknowledged. 1 Acknowledged.
— 29–27	0	Reserved. Write to 0 for future compatibility.	
DMA1_PD_STATUS 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_STATUS 25	0	**Serial RapidIO Port 1 Power Down Status** Indicates the power status of the serial RapidIO Port 1. Can be powered down by the Reset Control Word or GCR control.	0 Power up. 1 Power down.

Table 8-4. HSSI_SR Bit Descriptions (Continued)

Name	Reset	Description	Settings
SD_CB_PD_STATUS 24	x	**SerDes Control Bloc Power Down Status** Indicates the power status of the SerDes control block. The default value of this status bit depends on the value of the RCW[SP] field.	0 Power up. 1 Power down.
SD_ISO_RING_PD 23	x	**SerDes Control Isolation Ring Powered** Indicates the power status of the SerDes isolation ring. This can be powered down by the RCW or by GCR control. The default value of this status bit depends on the value of the RCW.	0 Power up. 1 Power down.
SD_FORCE_STOPPED 22	x	**SerDes Force Stop Status** Indicates whether the SerDes had a force stop. The default value of this status bit depends on the value of the RCW.	0 Normal. 1 Stopped.
DMA0_PD_STATUS 21	x	**DMA0 Power Down Status** Indicates the power status of DMA0. Can be powered down by the Reset Control Word or GCR control. The default value of this status bit depends on the value of the RCW[SP] field.	0 Power up. 1 Power down.
SRIO0_PD_STATUS 20	x	**Serial RapidIO Port 0 Power Down Status** Indicates the power status of the serial RapidIO Port 0. Can be powered down by the Reset Control Word or GCR control. The default value of this status bit depends on the value of the RCW[SP] field.	0 Power up. 1 Power down.
PEX_PD_STATUS 19	x	**PCI Express Power Down Status** Indicates the power status of the PCI Express controller. Can be powered down by the Reset Control Word or GCR control. The default value of this status bit depends on the value of the RCW[SP] field.	0 Power up. 1 Power down.
OCN_PD_STATUS 18	x	**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. The default value of this status bit depends on the value of the RCW[SP] field.	0 Power up. 1 Power down.
EMSG_PD_STATUS 17	x	**RapidIO Enhanced Messaging Unit Power Down Status** Indicates the power status of the RapidIO Enhanced Messaging Unit. Can be powered down by the Reset Control Word or GCR control. The default value of this status bit depends on the value of the RCW[SP] field.	0 Power up. 1 Power down.
SRIO1_STOP_ACK 16	0	**Serial RapidIO Port 1 Stop Acknowledge Status** Indicates the serial RapidIO Port 1 Stop Acknowledge status.	0 Not stopped. 1 Stop ACK issued.
SRIO0_STOP_ACK 15	0	**Serial RapidIO Port 0 Stop Acknowledge Status** Indicates the serial RapidIO Port 0 Stop Acknowledge status.	0 Not stopped. 1 Stop ACK issued.
PEX_STOP_ACK 14	0	**PCI Express Stop Acknowledge Status** Indicates the PCI Ex[press Stop Acknowledge status.	0 Not stopped. 1 Stop ACK issued.

Table 8-4. HSSI_SR Bit Descriptions (Continued)

Name	Reset	Description	Settings
EMSG_STOP_ACK 13	0	**RapidIO Enhanced Messaging Unit Stop Acknowledge Status** Indicates the EMSG Stop Acknowledge status.	0 Not stopped. 1 Stop ACK issued.
— 12	0	Reserved. Write to 0 for future compatibility.	
SERDES2_RST_DONE 11	0	**SERDES PLL2 Reset Done** Indicates whether the SERDES PLL2 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.
SERDES_PD 10	0	**SERDES Power Down Status** Indicates the power status of the SERDES PHY.	0 Power up. 1 Power down (PLL is not working).
SRIO1_IDLE 9	0	**Serial RapidIO Port 1 Idle** Indicates whether the Serial RapidIO Port 1 is idle, that is, no transactions in progress.	0 Active. 1 Idle.
SRIO1_OB_IDLE 8	0	**Serial RapidIO Port 1 Outbound Idle** Indicates whether the Serial RapidIO Port 1 outbound activity is idle, that is, no outbound transactions in progress.	0 Active. 1 Idle.
SERDES1_RST_DONE 7	0	**SERDES PLL1 Reset Done** Indicates whether the SERDES PLL1 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.
— 6	0	Reserved. Write to 0 for future compatibility.	
SRIO0_IDLE 5	0	**Serial RapidIO Port 0 Idle** Indicates whether the Serial RapidIO Port 0 unit is idle, that is, no transactions in progress.	0 Active. 1 Idle.
SRIO0_OB_IDLE 4	0	**Serial RapidIO Port 0 Outbound Idle** Indicates whether the Serial RapidIO Port 0 outbound activity is idle, that is, no outbound transactions in progress.	0 Active. 1 Idle.
PEX_IDLE 3	0	**PCI Express Idle** Indicates whether the Serial PCI Express unit is idle, that is, no transactions in progress.	0 Active. 1 Idle.
PEX_OB_IDLE 2	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.
— 1	0	Reserved. Write to 0 for future compatibility.	
EMSG_IDLE 0	0	**EMSG Idle** Indicates whether the EMSG 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
	—							
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
	—		DDR_QUEUE_DEPTH_DI		DDR_IO_REC_ADJ			—
Type	R/W							
Reset	0	0	0	0	1	1	0	0

Bit	7	6	5	4	3	2	1	0
	DDR_COP_DRAIN_QUEUES_FOR_SLEEP	DDR_COP_USE_POR_SAMPLED_VALUE	DDR_COP_TERMSEL_OVERRIDE_VALUE			DDR_COP_TERMSEL_OVERRIDE_EN	—	DDR_NO_WRAP_32
Type	R/W							
Reset	0	0	0	0	0	0	0	1

DDR_GCR controls the DDR operation the MSC8157E 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
— 31–14	0	Reserved. Write to 0 for future compatibility.	
DDR_QUEUE_DEPTH_DI 13–12	00	**DDR Input Queue Depth Select** Selects the depth of the input queue.	00 5 entries. 01 2 entries. 10 3 entries 11 4 entries
DDR_IO_REC_ADJ 11–9	110	**DDRC I/O Select Setting** Sets the value for the termination select override.	
— 8	0	Reserved. Write to 0 for future compatibility.	
DDR_COP_DRAIN_QUEUES_FOR_SLEEP 7	0	**DDRC Drain Queues for Sleep** Indicates whether to drain the queues when entering Sleep mode.	0 Do not drain input queue. 1 Drain input queue.
DDR_COP_USE_POR_SAMPLED_VALUE 6	0	**DDRC Use POR Sampled Value** Indicates whether to use the sampled value for driver impedance.	0 Do not use sampled value. 1 Use sampled value.

Table 8-5. DDR_GCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
DDR_COP_TERMSEL_OVERRIDE_VALUE 5–3	000	**DDRC Termination Select Override Value** Sets the value for the termination select override.	
DDR_COP_TERMSEL_OVERRIDE_EN 2	0	**DDRC Termination Select Override Enable** Disables/enables the termination select override.	0 Disable 1 Enable.
— 1	0	Reserved. Write to 0 for future compatibility.	
DDR_NO_WRAP_32 0	1	**DDRC No Wrap 32 Mode** Indicates where to wrap the memory.	0 Wrap occurs on 32-byte boundary. 1 Wrap occurs on 64-byte boundary.

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
	—		GPEX_ECC_ERR_INJECT	PEX2O_ECC_ERR_INJECT	DMA2OCN1_ECC_ERR_INJECT	DMA2OCN0_ECC_ERR_INJECT	SRIO1_DOZE	DMA2OCN1_ECC_D
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SRIO1_PD	EMSG_STOP	SRIO1_ECC_D	SD_SB_PD	SD_ISOR_PD	SD_DOZE	SRIO1_ECC_ERR_INJECT	SRIO0_ECC_ERR_INJECT
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SRIO0_DOZE	DMA2OCN0_ECC_D	SRIO0_PD	PEX_STOP	SRIO0_ECC_D	—	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
	PEX2O_ECC_D	GPEX_ECC_D	OCN_PD	EMSG_PD	EMSG_DOZE	EMSG_ECC_ERR_INJECT	—	
Type	R/W							
Reset	0	0	0	0	0	0	1	1

HSSI_CR1 controls various functions within the SerDes block for the MSC8157E 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
— 31–30	0	Reserved. Write to 0 for future compatibility.	
GPEX_ECC_ERR_INJECT 29	0	**PCI Express Multi-Bit ECC Error Inject** Indicates whether to inject a multibit error into the PCI Express controller.	0 Do not inject error. 1 Inject error.
PEX2O_ECC_ERR_INJECT 28	0	**PCI Express to OCN ECC Error Inject** Indicates whether to inject a multibit error into the PCI Express to OCN bridge.	0 Do not inject error. 1 Inject error.
DMA2OCN1_ECC_ERR_INJECT 27	0	**DMA to OCN Bridge1 Multi-Bit ECC Error Inject** Indicates whether to inject a multibit error into the DMA to OCN Bridge 1.	0 Do not inject error. 1 Inject error.
DMA2OCN0_ECC_ERR_INJECT 26	0	**DMA to OCN Bridge0 Multi-Bit ECC Error Inject** Indicates whether to inject a multibit error into the DMA to OCN Bridge 0.	0 Do not inject error. 1 Inject error.
SRIO1_DOZE 25	0	**Serial RapidIO Port 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 clock is stopped.	0 Normal operation. 1 Doze mode.
DMA2OCN1_ECC_D 24	0	**Disable DMA to OCN1 ECC** Enables/disables DMA to OCN1 bridge ECC.	0 ECC enabled. 1 ECC disabled.
SRIO1_PD 23	0	**Serial RapidIO Port 1 Power Down** Shuts down serial RapidIO Port 1 power.	0 Power up. 1 Power down.
EMSG_STOP 22	0	**EMSG Stop** Stops the RapidIO Enhanced Messaging Unit.	0 EMSG not stopped. 1 EMSG stopped.
SRIO1_ECC_D 21	0	**Disable Serial RapidIO Port 1 ECC** Enables/disables serial RapidIO Port 1 ECC.	0 ECC enabled. 1 ECC disabled.
SD_CB_PD 20	0	**SerDes Control Block Power Down** Shuts down SerDes control block power.	0 Power up. 1 Power down.
SD_ISOR_PD 19	0	**SerDes Isolation Ring Power Down** Shuts down the SerDes Isolation Ring power.	0 Power up. 1 Power down.
SD_DOZE 18	0	**SerDes 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 clock is stopped.	0 Normal operation. 1 Doze mode.
SRIO1_ECC_ERR_INJECT 17	0	**Serial RapidIO Port 1 Multi-Bit ECC Error Inject** Indicates whether to inject a multibit error into the Serial RapidIO Port 1.	0 Do not inject error. 1 Inject error.
SRIO0_ECC_ERR_INJECT 16	0	**Serial RapidIO Port 0 Multi-Bit ECC Error Inject** Indicates whether to inject a multibit error into the Serial RapidIO Port 0.	0 Do not inject error. 1 Inject error.

Table 8-6. HSSI_CR1 Bit Descriptions (Continued)

Name	Reset	Description	Settings
SRIO0_DOZE 15	0	**Serial RapidIO Port 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.
DMA2OCN0_ECC_D 14	0	**Disable DMA to OCN0 ECC** Enables/disables DMA to OCN0 bridge ECC.	0 ECC enabled. 1 ECC disabled.
SRIO0_PD 13	0	**Serial RapidIO Port 0 Power Down** Shuts down serial RapidIO Port 0 power.	0 Power up. 1 Power down.
PEX_STOP 12	0	**PCI Express Stop** Stops the PCI Express complex.	0 PCI Express not stopped. 1 PCI Express stopped.
SRIO0_ECC_D 11	0	**Disable Serial RapidIO Port 0 ECC** Enables/disables serial RapidIO Port 0 ECC.	0 ECC enabled. 1 ECC disabled.
— 10	0	Reserved. Write to 0 for future compatibility.	
PEX_PD 9	0	**PCI Express Power Down** Shuts down PCI Express clock 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 controller is stopped.	0 Normal operation. 1 Doze mode.
PEX2O_ECC_D 7	0	**Disable PCI Express to OCN ECC** Enables/disables PCI Express to OCN ECC.	0 ECC enabled. 1 ECC disabled.
GPEX_ECC_D 6	0	**Disable PCI Express ECC** Enables/disables PCI Express ECC.	0 ECC enabled. 1 ECC disabled.
OCN_PD 5	0	**OCN Power Down** Shuts down OCN fabric power.	0 Power up. 1 Power down.
EMSG_PD 4	0	**EMSG Power Down** Shuts down EMSG power.	0 Power up. 1 Power down.
EMST_DOZE 3	0	**EMSG 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 EMSG is stopped.	0 Normal operation. 1 Doze mode.
EMSG_ECC_ERR_INJECT 2	0	**EMSG Multi-Bit ECC Error Inject** Indicates whether to inject a multibit error into the EMSG.	0 Do not inject error. 1 Inject error.
— 1–0	0	Reserved. Write to 0 for future compatibility.	

Note: Doze mode is a special mode used by the MSC8157E 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
	—			CPRI_RESET	—			
Type	R/W							
Reset	0	0	0	1	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CPRI_LINK_RATE				PD_SRIO_1_TO_SD	PD_SRIO_0_TO_SD	HSSI_STOP	CPRI_STOP
Type	R/W							
Reset	1	0	1	0	0	0	0	0

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

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

HSSI_CR2 controls various functions within the SerDes block for the MSC8157E 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
— 31–29	0	Reserved. Write to 0 for future compatibility.	
CPRI_RESET 28	0	**CPRI Reset** Use to reset the CPRI controllers	0 Reset CPRI 1 Do not reset CPRI.
— 27–24	0	Reserved. Write to 0 for future compatibility.	
CPRI_LINK_RATE 23–20	1010	**CPRI Link Rate** Selects the CPRI link rate.	0010 1.2288 GHz 0100 2.4576 GHz 0101 3.072 GHz 1000 4.9152 GHz 1010 6.144 GHz 1111 CPRI powered down All other values reserved.
PD_SRIO_1_TO_SD 19	0	**SerDes Power Down** Powers down the SRIO1 pipe interface to SerDes if the SerDes PLL2 becomes unstable.	0 No action. 1 Power down SRIO1 pipe.
PD_SRIO__0_TO_SD 18	0	**SerDes Power Down** Powers down the SRIO0 pipe interface to SerDes if the SerDes PLL1 becomes unstable.	0 No action. 1 Power down SRIO0 pipe.

Table 8-7. HSSI_CR2 Bit Descriptions (Continued)

Name	Reset	Description	Settings
HSSI_STOP 17	0	**HSSI Stop** With the HSSI_IPG_STOP_ACK received from the HSSI, actively stops the clock to the HSSI complex and to its memory. If the memory is not put in self-refresh mode separately prior to ipg-stop activation, data will be lost,.	0 No action. 1 Stop clock.
CPRI_STOP 16	0	**CPRI Stop** With the CPRI_IPG_STOP_ACK received from the CPRI, actively stops the clock to the CPRI and to its memory. If the memory is not put in self-refresh mode separately prior to ipg-stop activation, data will be lost,.	0 No action. 1 Stop clock.
— 15–13	0	Reserved. Write to 0 for future compatibility.	
CPRI_REC_CLK0 12–10	0	**Reconstructed Clock Select** Used to choose the reconstructed clock to output to GPIO from the SerDes complex.	
— 9–2	0	Reserved. Write to 0 for future compatibility.	
EMSG_ECC_D 1	0	**RapidIO Enhanced Messaging Unit ECC Disable** Enables/disables the EMSG ECC.	0 ECC enabled. 1 ECC disabled.
— 0	0	Reserved. Write to 0 for future compatibility.	

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 MSC8157E 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 RGMII selected. 1 SGMII 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	1	1	1	1	1	1	1	1

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	1	1	1	1	1	1	1	1

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	1	1	1	1	1	1	1	1

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	1	1	1	1	1	1	1	1

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	0xFFFFFFFF	**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	1	0	1
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	0x830D	**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 MSC8157E Data Sheet includes recommended default values for this register to use with a standard RGMII PHY device. **AN4134** *RGMII Ethernet Timing in StarCore Based MSC8157 DSPs* (available under NDA) provides guidelines for adjusting GCR4 values for specific applications, if required.

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.
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.

Table 8-12. GCR4 Bit Descriptions (Continued)

Name	Reset	Description	Settings
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
	—	QE_STOP_MODE_B	SCOP_STOP	—				
Type	R/W							
Reset	0	1	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—	DROWSY_M3_V	—	DROWSY_M2_V	—	DROWSY_M3_EN	DROWSY_M2_EN	DROWSY_OVERRIDE_EN
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	0	Reserved. Write to zero for future compatibility.	
QE_STOP_MODE_B 30	0	**Stop QUICC Engine Block** Stops the QUICC Engine block operation.	0 Normal operation. 1 Stopped.
SCOP_STOP 29	0	**Stop SEC Block** Stops the SEC block operation.	0 Normal operation. 1 Stopped.
— 28–23	0	Reserved. Write to zero for future compatibility.	
DROWSY_M3_V 22	0	**Drowsy M3 Voltage Select** Selects the drowsy M3 memory voltage.	0 200 mV. 1 300 mV.
— 21	0	Reserved. Write to zero for future compatibility.	
DROWSY_M2_V 20	0	**Drowsy M2 Voltage Select** Selects the drowsy M2 memory voltage.	0 200 mV. 1 300 mV.

Table 8-13. GCR5 Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 19	0	Reserved. Write to zero for future compatibility.	
DROWSY_M3_EN 18	0	**Drowsy M3 Enable** Enables/disables the drowsy memory feature.	0 Disabled. 1 Enabled.
DROWSY_M2_EN 17	0	**Drowsy M2 Enable** Enables/disables the drowsy memory feature.	0 Disabled. 1 Enabled.
DROWSY_OVERRIDE_EN 16	0	**Drowsy Memory Override Enable** Enables/disables the drowsy memory override function. When disabled, override control is done via the RCW settings.	0 Disabled. 1 Enabled.
— 15–13	0	Reserved. Write to zero for future compatibility.	
PEX_IRQ_OUT 12	0	**PCI Express Message Signaled Interrupt Enable** Enables/disables triggering of a PCI Express message-signaled interrupt.	0 Disabled. 1 Enabled.
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.
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
	—		DDR_IDLE_MEM	DDR_YMMC_STOP_ACK	—			
Type	R							
Reset	0	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
	—							STOP_BS
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—		SCOP_IDLE	SCOP_STOP_ACK	OCNDMA1_IDLE	OCNDMA1_STOP_ACK	OCNDMA0_IDL:E	OCNDMA0_STOP_ACK
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
— 31–30	0	Reserved. Write to zero for future compatibility.	
DDR_IDLE_MEM 29	0	**DDR Controller Idle** Reflects the current status of DDR Controller.	0 Memory controller active. 1 Memory controller idle.
DDR_YMMC_STOP_ACK 30	0	**DDR Controller Refresh Mode** Reflects the current status of DDR Controller refresh mode.	0 Memory controller not in self-refresh mode. 1 Memory controller in self-refresh mode.
— 27–22	0	Reserved. Write to zero for future compatibility.	
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.

Table 8-14. GSR2 Bit Descriptions (Continued)

Name	Reset	Description	Settings
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.
— 15–9	0	Reserved. Write to zero for future compatibility.	
STOP_BS 8	0	**STOP_BS Input Status** Reflects the current status of the STOP_BS input signal.	0 STOP_BS not asserted. 1 STOP_BS asserted.
— 7–6	0	Reserved. Write to zero for future compatibility.	
SCOP_ IDLE 5	0	**SCOP Idle** Reflects the current status of the SEC block.	0 Active 1 Idle
SCOP_ STOP_ACK 2	0	**SCOP Stop Acknowledge** Indicates that the SEC block acknowledged a Stop request.	0 No stop ACK issued. 1 Stop ACK issued.
OCNDMA1_ IDLE 3	0	**OCNDMA1 Idle** Reflects the current status of the OCNDMA1 block.	0 Active 1 Idle
OCNDMA1_ STOP_ACK 2	0	**OCNDMA1 Stop Acknowledge** Indicates that the OCNDMA1 block acknowledged a Stop request.	0 No stop ACK issued. 1 Stop ACK issued.
OCNDMA0_ IDLE 1	0	**OCNDMA0 Idle** Reflects the current status of the OCNDMA0 block.	0 Active 1 Idle
OCNDMA0_ STOP_ACK 0	0	**OCNDMA0 Stop Acknowledge** Indicates that the OCNDMA0 block acknowledged a Stop request.	0 No stop ACK issued. 1 Stop ACK issued.

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 General Status Register 3 (GSR3)

GSR3 General Status Register 3 Offset 0x50

Bit	31	30	29	28	27	26	25	24
	CP_LOS6	CP_LOS5	CP_LOS4	CP_LOS3	CP_LOS2	CP_LOS1	—	PD_SRIO_1_TO_ SD_STATUS
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PD_SRIO_0_TO_SD_STATUS	SD2_RST_ERR	SD1_RST_ERR	—				
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
	—			CRPI_PU	MAPLE_ETVPE_PD	MAPLE_CRPE_PU	MAPLE_EQPE_PD	ERASE_DONE
Type	R							
Reset	0	0	0	0	0	0	0	0

GSR3 includes the status of several events. Those bits are not sticky and only sample the events. The GSR3 is reset by a hard reset event. All bits are cleared on reset.

Table 8-16. GSR3 Bit Descriptions

Name	Reset	Description	Settings
CP_LOS6 31	0	**CPRI6 LOS Reported** Indicates whether the smallform-factor pluggable (SFP) optical transceiver connected to CPRI6 indicates a loss-of-signal (LOS) status, meaning that the received optical power is below the worst-case receiver sensitivity.	0 Normal operation. 1 SFP device reported LOS.
CP_LOS5 30	0	**CPRI5 LOS Reported** Indicates whether the smallform-factor pluggable (SFP) optical transceiver connected to CPRI5 indicates a loss-of-signal (LOS) status, meaning that the received optical power is below the worst-case receiver sensitivity.	0 Normal operation. 1 SFP device reported LOS.
CP_LOS4 29	0	**CPRI4 LOS Reported** Indicates whether the smallform-factor pluggable (SFP) optical transceiver connected to CPRI4 indicates a loss-of-signal (LOS) status, meaning that the received optical power is below the worst-case receiver sensitivity.	0 Normal operation. 1 SFP device reported LOS.

Table 8-16. GSR3 Bit Descriptions (Continued)

Name	Reset	Description	Settings
CP_LOS3 28	0	**CPRI3 LOS Reported** Indicates whether the smallform-factor pluggable (SFP) optical transceiver connected to CPRI3 indicates a loss-of-signal (LOS) status, meaning that the received optical power is below the worst-case receiver sensitivity.	0 Normal operation. 1 SFP device reported LOS.
CP_LOS2 27	0	**CPRI2 LOS Reported** Indicates whether the smallform-factor pluggable (SFP) optical transceiver connected to CPRI2 indicates a loss-of-signal (LOS) status, meaning that the received optical power is below the worst-case receiver sensitivity.	0 Normal operation. 1 SFP device reported LOS.
CP_LOS1 26	0	**CPRI1 LOS Reported** Indicates whether the smallform-factor pluggable (SFP) optical transceiver connected to CPRI1 indicates a loss-of-signal (LOS) status, meaning that the received optical power is below the worst-case receiver sensitivity.	0 Normal operation. 1 SFP device reported LOS.
— 25	0	Reserved. Write to zero for future compatibility.	
PD_SRIO_1_ TO_ SD_ STATUS 24	x	**SRIO1 to SerDes Power Down Status** Reflects the status of the SRIO1 to SerDes port power. The default value depended on the value of the RCW[SP] field.	0 Power up. 1 Power down.
PD_SRIO_0_ TO_ SD_ STATUS 23	x	**SRIO0 to SerDes Power Down Status** Reflects the status of the SRIO0 to SerDes port power. The default value depended on the value of the RCW[SP] field.	0 Power up. 1 Power down.
SD2_RST_ERR 22	0	**SerDes PLL2 Reset Error** Reflects whether a SerDes PLL2 reset error occurred.	0 No error. 1 Error occurred.
SD1_RST_ERR 21	0	**SerDes PLL1 Reset Error** Reflects whether a SerDes PLL1 reset error occurred.	0 No error. 1 Error occurred.
— 21–5	0	Reserved. Write to zero for future compatibility.	
CPRI_PU 4	0	**CPRI Power Up** Reflects the status of the CPRI power (all lanes).	0 Power down. 1 Power up.
MAPLE_ETVPE_PD 3	0	**MAPLE-B ETVPE Power Down** Reflects the status of the MAPLE-B TVPE power.	0 Power up. 1 Power down.
MAPLE_CRPE_PU 2	0	**MAPLE-B CRPE Power Up** Reflects the status of the MAPLE-B CRPE power.	0 Power down. 1 Power up.
MAPLE_EQPE_PD 1	0	**MAPLE-B EQPE Power Down** Reflects the status of the MAPLE-B CRPE power.	0 Power up. 1 Power down.
ERASE_DONE 0	0	**Erase Done** Reflects the status of the erase operation.	0 Erase not done. 1 Erase done.

8.2.17 General Control Register 6 (GCR6)

GCR6 General Control Register 6 Offset 0x64

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
	—					RGMII2_SEL	—	
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GCR6 selects whether the RGMII2 pins support the CP_LOSx inputs or the RGMII2 signals. All bits are cleared on reset.

Table 8-17. GCR6 Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.	
RGMII2_SEL 29–24	0	**RGMII2 Select** The six CP_LOSx signals are multiplexed with six of the RGMII2 signals as follows: • CP_LOS1/GE2_RD2 • CP_LOS2/GE2_TD2 • CP_LOS3/GE2_RD3 • CP_LOS4/GE2_GTX_CLK • CP_LOS5/GE2_RD0 • CP_LOS6/GE2_TD3 This bit selects the signal set reflected at the connecting pins. The default (0) selects the six CP_LOSx signals. If the bit is set (1), it selects the RGMII2 signals. The remaining RGMII signals have dedicated connections that are not multiplexed.	0 CP_LOSx signals selected 1 RGMII2 signals selected.
— 1–0	0	Reserved. Write to zero for future compatibility.	

8.2.18 General Control Register 7 (GCR7)

GCR7 General Control Register 7 Offset 0x68

Bit	31	30	29	28	27	26	25	24
	—		TIMER_32B_0_MUX3_SEL					
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	—		TIMER_32B_0_MUX2_SEL					
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	—		TIMER_32B_0_MUX1_SEL					
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—		TIMER_32B_0_MUX0_SEL					
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GCR7 selects the inputs for the four timers in 32-bit Timer0. All bits are cleared on reset.

Table 8-18. GCR7 Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
TIMER_32B_0_MUX3_SEL 29–24	0	**Input Select for 32-bit Timer 3** Selects the input (out of 64) for the timer.	See **Table 8-19** for multiplexing options.
— 23–22	0	Reserved. Write to zero for future compatibility.	
TIMER_32B_0_MUX2_SEL 21–16	0	**Input Select for 32-bit Timer 2** Selects the input (out of 64) for the timer.	See **Table 8-19** for multiplexing options.
— 15–14	0	Reserved. Write to zero for future compatibility.	
TIMER_32B_0_MUX1_SEL 13–8	0	**Input Select for 32-bit Timer 1** Selects the input (out of 64) for the timer.	See **Table 8-19** for multiplexing options.
— 7–6	0	Reserved. Write to zero for future compatibility.	
TIMER_32B_0_MUX0_SEL 5–0	0	**Input Select for 32-bit Timer 0** Selects the input (out of 64) for the timer.	See **Table 8-19** for multiplexing options.

Table 8-19. Timer Input Multiplexing Options

Value (mode)	Multiplexer Inputs	Comments
0x0 (0)	TMR0	External
0x1 (1)	TMR1	External
0x2 (2)	TMR2	External
0x3 (3)	TMR3	External
0x4 (4)	TMR4	External
0x5 (5)	TMR5	External
0x6 (6)	TMR6	External
0x7 (7)	reserved	—
0x8 (8)	CPRI1 Rx HFP Out	66.67 µs
0x9 (9)	CPRI2 Rx HFP Out	66.67 µs
0xA (10)	CPRI3 Rx HFP Out	66.67 µs
0xB (11)	CPRI4 Rx HFP Out	66.67 µs
0xC (12)	CPRI5 Rx HFP Out	66.67 µs
0xD (13)	CPRI6 Rx HFP Out	66.67 µs
0xE (14)	reserved	—
0xF (15)	reserved	—
0x10 (16)	CPRI1 Tx HFP Out	66.67 µs
0x11 (17)	CPRI2 Tx HFP Out	66.67 µs
0x12 (18)	CPRI3 Tx HFP Out	66.67 µs
0x13 (19)	CPRI4 Tx HFP Out	66.67 µs
0x14 (20)	CPRI5 Tx HFP Out	66.67 µs
0x15 (21)	CPRI6 Tx HFP Out	66.67 µs
0x16 (22)	reserved	—
0x17 (23)	reserved	—
0x18 (24)	CPRI1 Rx BFP Out	260 ns
0x19 (25)	CPRI2 Rx BFP Out	260 ns
0x1A (26)	CPRI3 Rx BFP Out	260 ns
0x1B (27)	CPRI4 Rx BFP Out	260 ns
0x1C (28)	CPRI5 Rx BFP Out	260 ns
0x1D (29)	CPRI6 Rx BFP Out	260 ns
0x1E (30)	reserved	—
0x1F (31)	reserved	—
0x20 (32)	CPRI1 Tx BFP Out	260 ns
0x21 (33)	CPRI2 Tx BFP Out	260 ns
0x22 (34)	CPRI3 Tx BFP Out	260 ns
0x23 (35)	CPRI4 Tx BFP Out	260 ns

Table 8-19. Timer Input Multiplexing Options (Continued)

Value (mode)	Multiplexer Inputs	Comments
0x24 (36)	CPRI5 Tx BFP Out	260 ns
0x25 (37)	CPRI6 Tx BFP Out	260 ns
0x26 (38)	reserved	—
0x27 (39)	reserved	—
0x28 (40)	CPRI1 Rx NBFP Out	10 ms
0x29 (41)	CPRI2 Rx NBFP Out	10 ms
0x2A (42)	CPRI3 Rx NBFP Out	10 ms
0x2B (43)	CPRI4 Rx NBFP Out	10 ms
0x2C (44)	CPRI5 Rx NBFP Out	10 ms
0x2D (45)	CPRI6 Rx NBFP Out	10 ms
0x2E (46)	reserved	—
0x2F (47)	reserved	—
0x30 (48)	CPRI1 Tx NBFP Out	10 ms
0x31 (49)	CPRI2 Tx NBFP Out	10 ms
0x32 (50)	CPRI3 Tx NBFP Out	10 ms
0x33 (51)	CPRI4 Tx NBFP Out	10 ms
0x34 (52)	CPRI5 Tx NBFP Out	10 ms
0x35 (53)	CPRI6 Tx NBFP Out	10 ms
0x36 (54)	reserved	—
0x37 (55)	reserved	—
0x38 (56)	reserved	—
0x39 (57)	reserved	—
0x3A (58)	reserved	—
0x3B (59)	reserved	—
0x3C (60)	reserved	—
0x3D (61)	reserved	—
0x3E (62)	reserved	—
0x3F (63)	reserved	—

8.2.19 General Control Register 8 (GCR8)

GCR8 General Control Register 8 Offset 0x6C

Bit	31	30	29	28	27	26	25	24
	TIMERS_32B_CLKMUX_SEL	TIMERS_CLKMUX_SEL	TIMER_32B_1_MUX3_SEL					
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—		TIMER_32B_1_MUX2_SEL					
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—		TIMER_32B_1_MUX1_SEL					
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—		TIMER_32B_1_MUX0_SEL					
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GCR8 selects the inputs for the four timers in 32-bit Timer1. All bits are cleared on reset.

Table 8-20. GCR8 Bit Descriptions

Name	Reset	Description	Settings
TIMERS_32B_CLKMUX_SEL 31	0	**Timers_32b Clock Multiplex Select** Used to select the input to the clock multiplexer.	0 250 MHz (peripherals) 1 122.88 MHz (CPRI)
TIMERS_CLKMUX_SEL 30	0	**Timers Clock Multiplex Select** Used to select the input to the clock multiplexer.	0 250 MHz (peripherals) 1 122.88 MHz (CPRI)
TIMER_32B_1_MUX3_SEL 29–24	0	**Input Select for 32-bit Timer 3** Selects the input (out of 64) for the timer.	See **Table 8-19** for multiplexing options.
— 23–22	0	Reserved. Write to zero for future compatibility.	
TIMER_32B_1_MUX2_SEL 21–16	0	**Input Select for 32-bit Timer 2** Selects the input (out of 64) for the timer.	See **Table 8-19** for multiplexing options.
— 15–14	0	Reserved. Write to zero for future compatibility.	

Table 8-20. GCR8 Bit Descriptions (Continued)

Name	Reset	Description	Settings
TIMER_32B_1_MUX1_SEL 13–8	0	**Input Select for 32-bit Timer 1** Selects the input (out of 64) for the timer.	See **Table 8-19** for multiplexing options.
— 7–6	0	Reserved. Write to zero for future compatibility.	
TIMER_32B_1_MUX0_SEL 5–0	0	**Input Select for 32-bit Timer 0** Selects the input (out of 64) for the timer.	See **Table 8-19** for multiplexing options.

8.2.20 General Control Register 10 (GCR10)

GCR10 General Control Register 10 Offset 0x74

Bit	31	30	29	28	27	26	25	24
	GPIO_SLEW_RATE_CONTROL				—			
Type	R/W							
Reset	1	1	1	1	1	1	1	1
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	—							
Type	RW							
Reset	1	1	1	1	1	1	1	1
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	—							
Type	RW							
Reset	1	1	1	1	1	1	1	1
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—		OE_DDR_CLK_MCK_3	OE_DDR_CLK_MCK_0	OE_DDR_CLK_MCK_1	OE_DDR_CLK_MCK_2	—	
Type	RW							
Reset	1	1	1	1	1	1	1	1

GCR10 includes bits to control several functions. The GCR10 is reset by a hard reset event. All bits are cleared on reset.

Table 8-21. GCR10 Bit Descriptions

Name	Reset	Description	Settings
GPIO_SLEW_RATE_CONTROL 31	1	**GPIO Slew Rate Control** Used with GPIO signal to set the slew rate for the signals.	0 Medium. 1 High.
— 30–6	0x1FFFFFF	Reserved. Write to ones for future compatibility.	

Table 8-21. GCR10 Bit Descriptions (Continued)

Name	Reset	Description	Settings
OE_DDR_CLK_MCK_1 5	1	**DDR MCK3 Clock Output Enable** Enables/disables the DDR MCK3 clock output.	0 Output clock disabled. 1 Output clock enabled.
OE_DDR_CLK_MCK_0 4	1	**DDR MCK0 Clock Output Enable** Enables/disables the DDR MCK0 clock output.	0 Output clock disabled. 1 Output clock enabled.
OE_DDR_CLK_MCK_1 3	1	**DDR MCK1 Clock Output Enable** Enables/disables the DDR MCK1 clock output.	0 Output clock disabled. 1 Output clock enabled.
OE_DDR_CLK_MCK_2 2	1	**DDR MCK2 Clock Output Enable** Enables/disables the DDR MCK2 clock output.	0 Output clock disabled. 1 Output clock enabled.
— 1–0	11	Reserved. Write to ones for future compatibility.	

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							
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	—	
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
	—							
Type	R							
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 8-22. GIR1 Bit Descriptions

Name	Reset	Description	Settings
SWT7 31	0	**Software Watchdog Timer 7** Reflects the status of the watchdog timer interrupt.	0 Interrupt not asserted 1 Interrupt asserted
SWT6 30	0	**Software Watchdog Timer 6** Reflects the status of the watchdog timer interrupt.	0 Interrupt not asserted 1 Interrupt asserted
SWT5 29	0	**Software Watchdog Timer 5** Reflects the status of the watchdog timer interrupt.	0 Interrupt not asserted 1 Interrupt asserted
SWT4 28	0	**Software Watchdog Timer 4** Reflects the status of the watchdog timer interrupt.	0 Interrupt not asserted 1 Interrupt asserted
SWT3 27	0	**Software Watchdog Timer 3** Reflects the status of the watchdog timer interrupt.	0 Interrupt not asserted 1 Interrupt asserted
SWT2 26	0	**Software Watchdog Timer 2** Reflects the status of the watchdog timer interrupt.	0 Interrupt not asserted 1 Interrupt asserted
SWT1 25	0	**Software Watchdog Timer 1** Reflects the status of the watchdog timer interrupt.	0 Interrupt not asserted 1 Interrupt asserted
SWT0 24	0	**Software Watchdog Timer 0** Reflects the status of the watchdog timer interrupt.	0 Interrupt not asserted 1 Interrupt asserted
O2M1_ERR 23	0	**O2M 1 Error** Reflects the status of the O2M1 error interrupt.	0 Interrupt not asserted 1 Interrupt asserted
O2M0_ERR 22	0	**O2M 0 Error** Reflects the status of the O2M0 error interrupt.	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.	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	0	Reserved. Write to zero for future compatibility.	

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	SWT6_EN	SWT5_EN	SWT4_EN	SWT3_EN	SWT2_EN	SWT1_EN	SWT0_EN
Type	R/W							
Reset	1	1	1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	O2M1_ERR_EN	O2M0_ERR_EN	—	DMA_ERR_EN	CE_IECC_EN	CE_DECC_EN	—	
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
	—							
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 8-23. GIER1_[0–5] Bit Descriptions

Name	Reset	Description	Settings
SWT7_EN 31	1	**SWT 7 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT6_EN 30	1	**SWT 6 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT5_EN 29	1	**SWT 5 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT4_EN 28	1	**SWT 4 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT3_EN 27	1	**SWT 3 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT2_EN 26	1	**SWT 2 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT1_EN 25	1	**SWT 1 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled

Table 8-23. GIER1_[0–5] Bit Descriptions

Name	Reset	Description	Settings
SWT0_EN 24	1	**SWT 0 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
O2M1_ERR_EN 23	0	**O2M1 Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
O2M0_ERR_EN 22	0	**O2M0 Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
— 21	0	Reserved. Write to zero for future compatibility.	
DMA_ERR_EN 20	0	**DMA Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CE_IECC_EN 19	0	**ECC Error Interrupt of the QUICC Engine IMEM Enable**	0 Interrupt disabled 1 Interrupt enabled
CE_DECC_EN 18	0	**ECC Error Interrupt of the QUICC Engine DRAM Enable**	0 Interrupt disabled 1 Interrupt enabled
— 17–0	0	Reserved. Write to zero for future compatibility.	

8.2.23 General Interrupt Register 3 (GIR3)

GIR3 General Interrupt Register 3 Offset 0xA4

Bit	31	30	29	28	27	26	25	24
	—					PEX2O_HW_ASSERT	CLS1_ERR	CLS1_WP
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CLS1_OV	—	DDR_ERR	—			MAPLE_ECC_ERR	—
Type	R							
Reset	0	0	0	0	0	0	0	0

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

GIR3 includes interrupt status of some debug/profiling events within MSC8157E. 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 8-24. GIR3 Bit Descriptions

Name	Reset	Description	Settings
— 31–27	0	Reserved. Write to zero for future compatibility.	
PEX2O_HW_ASSERT 26	0	**PCI Express to OCN Hardware Error** Reflects PCI Express to OCN hardware error Interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS1_ERR 25	0	**CLASS1 Error Interrupt** Reflects CLASS1 Error Interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS1_WP 24	0	**CLASS1 Watchpoint Interrupt** Reflects Class1 watchpoint interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS1_OV 23	0	**CLASS1 Overrun Interrupt** Reflects CLASS1 overrun interrupt	0 Interrupt not asserted 1 Interrupt asserted
— 22	0	Reserved. Write to zero for future compatibility.	
DDR_ERR 21	0	**DDR Error Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
— 20–18	0	Reserved. Write to zero for future compatibility.	
MAPLE_ECC_ERR 17	0	**MAPLE 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.	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
	—					PEX2O_HW_ASSERT_EN	CLS1_ERR_EN	CLS1_WP_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CLS1_OV_EN	—	DDR_ERR_EN	—			MAPLE_ECC_ERR_EN	—
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	—			PM_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—			CLS0_ERR_EN	—		CLS0_WP_EN	CLS0_OV_EN
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 MSC8157E. 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 8-25. GIER3_[0–5] Bit Descriptions

Name	Reset	Description	Settings
— 31–26	0	Reserved. Write to zero for future compatibility.	
PEX2O_HW_ASSERT_EN 26	0	**PCI Express to OCN Hardware Error Enable**	0 Interrupt not asserted 1 Interrupt asserted
CLS1_ERR_EN 25	0	**CLASS1 Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS1_WP_EN 24	0	**CLASS1 Watchpoint Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS1_OV_EN 23	0	**CLASS1 Overrun Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled

Table 8-25. GIER3_[0–5] Bit Descriptions

Name	Reset	Description	Settings
— 22	0	Reserved. Write to zero for future compatibility.	
DDR_ERR_EN 21	0	**DDR Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
— 20–18	0	Reserved. Write to zero for future compatibility.	
MAPLE_ECC_ERR_EN 17	0	**MAPLE ECC Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
— 13–16	0	Reserved. Write to zero for future compatibility.	
MAPLE_GEN_ERR_EN 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 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 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 1	0	**CLASS0 Watchpoint Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS0_OV_EN 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
	—							GPEX_ECC_ERR
Type	W1C							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PEX2O_ECC_ERR	EMSG_ECC_ERR	DMA2OCN1_ECC_ERR	DMA2OCN0_ECC_ERR	SRIO1_ECC_ERR	SRIO0_ECC_ERR	—	
Type	W1C							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—						VSG_GIC_VIRQ31_STICKY	VSG_GIC_VIRQ30_STICKY
Type	W1C							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VSG_GIC_VIRQ29_STICKY	VSG_GIC_VIRQ28_STICKY	VSG_GIC_VIRQ276_STICKY	VSG_GIC_VIRQ26_STICKY	—	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 MSC8157E. 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 8-26. GIR5 Bit Descriptions

Name	Reset	Description	Settings
— 31–23	0	Reserved. Write to zero for future compatibility.	
GPEX_ECC_ERR 24	0	**PCI Express ECC Error** Reflects a multi-bit.ECC error in the PCI Express complex.	0 No error. 1 Error detected.
PEX2O_ECC_ERR 23	0	**PCI Express to OCN ECC Error** Reflects a multi-bit.ECC error in the PCI Express to OCN bridge.	0 No error. 1 Error detected.
EMSG_ECC_ERR 22	0	**EMSG ECC Error** Reflects a multi-bit.ECC error in the EMSG.	0 No error. 1 Error detected.
DMA2OCN1_ECC_ERR 21	0	**DMA2OCN1 ECC Error** Reflects a multi-bit.ECC error in the DMA-to-OCN bridge 1.	0 No error. 1 Error detected.
DMA2OCN0_ECC_ERR 20	0	**DMA2OCN0 ECC Error** Reflects a multi-bit.ECC error in the DMA-to-OCN bridge 0.	0 No error. 1 Error detected.

Table 8-26. GIR5 Bit Descriptions

Name	Reset	Description	Settings
SRIO1_ ECC_ERR 19	0	**Serial RapidIO Port 1 ECC Error** Reflects a multi-bit.ECC error in the serial RapIdIO Port 1.	0 No error. 1 Error detected.
SRIO0_ ECC_ERR 18	0	**Serial RapidIO Port 0 ECC Error** Reflects a multi-bit.ECC error in the serial RapIdIO Port 0.	0 No error. 1 Error detected.
— 17–10	0	Reserved. Write to zero for future compatibility.	
VSG_GIC_ VIRQ31_ STICKY 9	0	**Virtual Interrupt 31** Reflects the status of virtual interrupt 31. This interrupt is generated by RISC4 in the MAPLE-B 2L system.	0 Interrupt not asserted 1 Interrupt asserted
VSG_GIC_ VIRQ30_ STICKY 8	0	**Virtual Interrupt 30** Reflects the status of virtual interrupt 30. This interrupt is generated by RISC3 in the MAPLE-B 2L system.	0 Interrupt not asserted 1 Interrupt asserted
VSG_GIC_ VIRQ29_ STICKY 7	0	**Virtual Interrupt 29** Reflects the status of virtual interrupt 29. This interrupt is generated by RISC2 in the MAPLE-B 2L system.	0 Interrupt not asserted 1 Interrupt asserted
VSG_GIC_ VIRQ28_ STICKY 6	0	**Virtual Interrupt 28** Reflects the status of virtual interrupt 28. This interrupt is generated by RISC1 in the MAPLE-B 2L system.	0 Interrupt not asserted 1 Interrupt asserted
VSG_GIC_ VIRQ27_ STICKY 5	0	**Virtual Interrupt 27** Reflects the status of virtual interrupt 27. This interrupt is generated by RISC2 in the QUICC Engine subsystem.	0 Interrupt not asserted 1 Interrupt asserted
VSG_GIC_ VIRQ26_ STICKY 4	0	**Virtual Interrupt 26** Reflects the status of virtual interrupt 26. This interrupt is generated by RISC1 in the QUICC Engine subsystem.	0 Interrupt not asserted 1 Interrupt asserted
— 3	0	Reserved. Write to zero for future compatibility.v	
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.	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.	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.	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
	—							GPEX_ECC_ERR_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PEX2O_ECC_ERR_EN	EMSG_ECC_ERR_EN	DMA2OCN1_ECC_ERR_EN	DMA2OCN0_ECC_ERR_EN	SRIO1_ECC_ERR_EN	SRIO0_ECC_ERR_EN	—	
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—						VSG_GIC_VIRQ31_STICKY_EN	VSG_GIC_VIRQ30_STICKY_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VSG_GIC_VIRQ29_STICKY_EN	VSG_GIC_VIRQ28_STICKY_EN	VSG_GIC_VIRQ276_STICKY_EN	VSG_GIC_VIRQ26_STICKY_EN	—	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 8-27. GIER5_[0–5] Bit Descriptions

Name	Reset	Description	Settings
— 31–25	0	Reserved. Write to zero for future compatibility.	
GPEX_ECC_ERR_EN 24	0	**PCI Express ECC Error Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
PEX2O_ECC_ERR_EN 23	0	**PCI Express to OCN ECC Error Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_ECC_ERR_EN 22	0	**EMSG ECC Error Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled

Table 8-27. GIER5_[0–5] Bit Descriptions

Name	Reset	Description	Settings
DMA2OCN1_ECC_ERR_EN 21	0	**DMA2OCN1 ECC Error Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
DMA2OCN0_ECC_ERR_EN 20	0	**DMA2OCN0 ECC Error Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
SRIO1_ECC_ERR_EN 19	0	**Serial RapidIO Port 1 ECC Error Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
SRIO0_ECC_ERR_EN 18	0	**Serial RapidIO Port 0 ECC Error Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
— 17–10	0	Reserved. Write to zero for future compatibility.	
VSG_GIC_ VIRQ31_ STICKY_EN 9	0	**Virtual Interrupt 31 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
VSG_GIC_ VIRQ30_ STICKY_EN 8	0	**Virtual Interrupt 30 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
VSG_GIC_ VIRQ29_ STICKY_EN 7	0	**Virtual Interrupt 29 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
VSG_GIC_ VIRQ28_ STICKY_EN 6	0	**Virtual Interrupt 28 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
VSG_GIC_ VIRQ27_ STICKY_EN 5	0	**Virtual Interrupt 27 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
VSG_GIC_ VIRQ26_ STICKY_EN 4	0	**Virtual Interrupt 26** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
— 3	0	Reserved. Write to zero for future compatibility.v	
T4_T5_AE_EN 2	0	**Core 4 or Core 5 L2/M2 Access Error Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
T2_T3_AE_EN 1	0	**Core 2 or Core 3 L2/M2 Access Error Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
T0_T1_AE_EN 0	0	**Core 0 or Core 1 L2/M2 Access Error Interrupt Enable** Enables/disables 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
	—							MAGMA_SYS_FASTCCI_LATE_ARB
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**
	MAGMA_SYS_FAST_INIT1_WEIGHT				MAGMA_SYS_FAST_INIT0_WEIGHT			
Type	R/W							
Reset	1	1	1	1	1	1	1	1

GCR13 controls the MBus system. It controls fast arbitration.

Table 8-28. GCR11 Bit Descriptions

Name	Reset	Description	Settings
— 31–25	0	Reserved. Write to zero for future compatibility.	
MAGMA_SYS_FAST_CCI_LATE_ARB 24	0	**MBus System Fast Late Arbitration** Enables fast late arbitration.	0 Fast late arbitration not enabled. 1 Fast late arbitration enabled.
— 23–8	0	Reserved. Write to zero for future compatibility.	
MAGMA_SYS_FAST_INIT1_WEIGHT 7–4	1111	**MBus System Fast Arbitration Initial Weight 1** Selects the initial fast arbitration weight for QUICC Engine accesses.	Ranges from 0000 (lowest) to 1111 (highest weight—default)
MAGMA_SYS_FAST_INIT0_WEIGHT 7–4	1111	**MBus System Fast Arbitration Initial Weight 0** Selects the initial fast arbitration weight for internal test and SEC (if present) accesses.	Ranges from 0000 (lowest) to 1111 (highest weight—default)

8.2.28 General Control Register 13 (GCR13)

GCR13 General Control Register 13 Offset 0x118

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

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

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

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

GCR13 controls the MAPLE-B input bus. It allows a master to write to registers available to the MAPLE RISC engines. All bits are cleared on reset.

Table 8-29. GCR13 Bit Descriptions

Name	Reset	Description	Settings
MAPLE_PIO_IN 31–8	0	**MAPLE Input** Allows a master to write to registers available to the MAPLE RISC engines.	
— 7–0	0	Reserved. Write to zero for future compatibility.	

8.2.29 General Status Register 8 (GSR8)

GSR8 General Status Register 8 Offset 0x11C

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

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

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

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

GSR8 receives values from the MAPLE-B output bus. All bits are cleared on reset.

Table 8-30. GSR8 Bit Descriptions

Name	Reset	Description	Settings
MAPLE_PIO_OUT 31–8	0	**MAPLE Output** Allows a master to read from registers available to the MAPLE RISC engines.	
— 7–0	0	Reserved. Write to zero for future compatibility.	

8.2.30 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 8-31. 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 8-31. 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 8-31. 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 8-31. 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.

8.2.31 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 8-32. 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 8-32. 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 8-32. 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 8-32. 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.

8.2.32 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 8-33. 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 8-33. 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 8-33. 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.

8.2.33 DDR Controller General Configuration Register (DDRC_GCR)

DDRC_GCR DDR Controller General Configuration Register Offset 0x12C

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

Bit	23	22	21	20	19	18	17	16
	MBUS_PLUS_REF_SRC_ID				MBUS_PLUS_LATE_ARB_CTL	MBUS_PLUS_INIT1_WTD_ARB_CTL		
Type	R/W							
Reset	0	0	0	0	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	MBUS_PLUS_INIT1_WTD_ARB_CTL				MBUS_PLUS_INIT0_WTD_ARB_CTL			
Type	R/W							
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	MBUS_PLUS_INIT0_WTD_ARB_CTL			MBUS_PLUS	—	DDRC_STOP	DDRC_DOZE	DDRC_PWR_DOWN
Type	R/W							
Reset	1	1	1	1	0	0	0	0

DDRC_GCR controls part of the DDR controller operation. All bits are cleared on a hard reset event.

Table 8-34. DDRC_GCR Bit Descriptions

Name	Reset	Description	Settings
— 31–26	0	Reserved. Write to zero for future compatibility.	
MBUS_PLUS_REF_RWB 25	0	**MBus Plus Read Write Bit** Used to enter a read/write bit for profiling.	
MBUS_PLUS_REF_SRC_ID 24–20	00000	**MBus Plus Source ID** Used to identify the source ID for profiling.	
MBUS_PLUS_LATE_ARB_CTL 19	1	**MBus Plus Late Arbitration Control** Enables/disables late arbitration.	0 Do not enable late arbitration. 1 Enable late arbitration.

Table 8-34. DDRC_GCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
MBUS_PLUS_INIT1_WTD_ARB_CTL 18–12	1111111	**MBus Plus Initialor 1 Weighted Arbitration Control** Selects the weighted value for initiator 1.	
MBUS_PLUS_INIT0_WTD_ARB_CTL 11–5	1111111	**MBus Plus Initialor 0 Weighted Arbitration Control** Selects the weighted value for initiator 0.	
MBUS_PLUS 4	1	**MBus Plus** Used to select MBus plus mode.	0 MBus Plus not selected. 1 MBus Plus selected.
— 3	0	Reserved. Write to zero for future compatibility.	
DDRC_STOP 2	0	**DDRC 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 DDRC Stop not requested. 1 DDRC Stop requested.
DDRC_DOZE 1	0	**DDRC 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 is stopped. **Note:** To conserve power, when DDR 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.
DDRC_POWER_DOWN 0	0	**DDRC Power Down** When set, powers down all DDR areas with gated clocks.	0 DDRC power up. 1 DDRC 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 8-35. 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 8-35. 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.

8.2.35 QUICC Engine Input General Control Register (QE_PIO_IN_GCR)

QE_PIO_IN_GCR QUICC Engine Input General Control Register Offset 0x13C

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

QE_PIO_IN_GCR controls the QUICC Engine input bus. It allows a master to write to registers available to the QUICC Engine RISC engines. All bits are cleared on reset.

Table 8-36. QE_PIO_IN_GCR Bit Descriptions

Name	Reset	Description	Settings
QE_PIO_IN 31–8	0	**QUICC Engine Input** Allows a master to write to registers available to the QUICC Engine RISC engines.	
— 7–0	0	Reserved. Write to zero for future compatibility.	

8.2.36 QUICC Engine Output General Status Register (QE_PIO_OUT_GSR)

QE_PIO_OUT_GSR QUICC Engine Output General Status Register Offset 0x140

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

QE_PIO_OUT_GSR receives values from the QUICC Engine subsystem output bus. All bits are cleared on reset.

Table 8-37. QE_PIO_OUT_GSR Bit Descriptions

Name	Reset	Description	Settings
QE_PIO_OUT 31–8	0	**QUICC Engine Output** Allows a master to read from registers available to the QUICC Engine RISC engines.	
— 7–0	0	Reserved. Write to zero for future compatibility.	

8.2.37 L2Q Arbitration Control for Core Subsystems 0 and 1 (MEX_T2_0_1_ARB)

MEX_T2_0_1_ARB L2Q Arbitration Control for Core Subsystems 0 and 1 Offset 0x150

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**
	—							T_0_1_UPGRADE_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	T_0_1_UPGRADE_VALUE							
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	T_0_1_UPGRADE_VALUE							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

MEX_T2_0_1_ARB enables and sets the priority auto-upgrade value for Core Subsystems 0 and 1. All bits are cleared on reset.

Table 8-38. MEX_T2_0_1_ARB Bit Descriptions

Name	Reset	Description	Settings
— 31–17	0	Reserved. Write to zero for future compatibility.	
T_0_1_ UPGRADE_EN 16	0	**L2Q Arbitration Control Enable** Controls whether priority auto-upgrade is enabled.	0 Disabled 1 Enabled
T_0_1_ UPGRADE_ VALUE 15–0	0	**L2Q Arbitration Value** Determines the value to use for priority auto upgrade when enabled.	

8.2.38 L2Q Arbitration Control for Core Subsystems 2 and 3 (MEX_T2_2_3_ARB)

MEX_T2_2_3_ARB L2Q Arbitration Control for Core Subsystems 2 and 3 Offset 0x154

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**
	—							T_2_3_ UPGRADE_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	T_2_3_UPGRADE_VALUE							
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	T_2_3_UPGRADE_VALUE							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

MEX_T2_2_3_ARB enables and sets the priority auto-upgrade value for Core Subsystems 2 and 3. All bits are cleared on reset.

Table 8-39. MEX_T2_2_3_ARB Bit Descriptions

Name	Reset	Description	Settings
— 31–17	0	Reserved. Write to zero for future compatibility.	
T_2_3_ UPGRADE_EN 16	0	**L2Q Arbitration Control Enable** Controls whether priority auto-upgrade is enabled.	0 Disabled 1 Enabled
T_2_3_ UPGRADE_ VALUE 15–0	0	**L2Q Arbitration Value** Determines the value to use for priority auto upgrade when enabled.	

8.2.39 L2Q Arbitration Control for Core Subsystems 4 and 5 (MEX_T2_4_5_ARB)

MEX_T2_4_5_ARB L2Q Arbitration Control for Core Subsystems 4 and 5 Offset 0x158

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**
	—							T_4_5_ UPGRADE_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	T_4_5_UPGRADE_VALUE							
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	T_4_5_UPGRADE_VALUE							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

MEX_T2_4_5_ARB enables and sets the priority auto-upgrade value for Core Subsystems 4 and 5. All bits are cleared on reset.

Table 8-40. MEX_T2_4_5_ARB Bit Descriptions

Name	Reset	Description	Settings
— 31–17	0	Reserved. Write to zero for future compatibility.	
T_4_5_ UPGRADE_EN 16	0	**L2Q Arbitration Control Enable** Controls whether priority auto-upgrade is enabled.	0 Disabled 1 Enabled
T_4_5_ UPGRADE_ VALUE 15–0	0	**L2Q Arbitration Value** Determines the value to use for priority auto upgrade when enabled.	

8.2.40 General Interrupt Register 6 (GIR6)

GIR6 General Interrupt Register 6 Offset 0x170

Bit	31	30	29	28	27	26	25	24
	—		CPRI_TX_CTRL6	CPRI_TX_CTRL5	CPRI_TX_CTRL4	CPRI_TX_CTRL3	CPRI_TX_CTRL2	CPRI_TX_CTRL1
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—		CPRI_RX_CTRL6	CPRI_RX_CTRL5	CPRI_RX_CTRL4	CPRI_RX_CTRL3	CPRI_RX_CTRL2	CPRI_RX_CTRL
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—		CPRI_TX_FRAME_TIMNG6	CPRI_TX_FRAME_TIMNG5	CPRI_TX_FRAME_TIMNG4	CPRI_TX_FRAME_TIMNG3	CPRI_TX_FRAME_TIMNG2	CPRI_TX_FRAME_TIMNG1
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—		CPRI_RX_FRAME_TIMNG6	CPRI_RX_FRAME_TIMNG5	CPRI_RX_FRAME_TIMNG4	CPRI_RX_FRAME_TIMNG3	CPRI_RX_FRAME_TIMNG2	CPRI_RX_FRAME_TIMNG1
Type	R							
Reset	0	0	0	0	0	0	0	0

GIR6 includes interrupt status of CPRI events. Those bits are not sticky, but only sample events. GIR6 is reset by a hard reset event. All bits are cleared on reset.

Table 8-41. GIR6 Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.v	
CPRI_TX_CTRL6 29	0	**CPRI6 Transmit Control Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_TX_CTRL5 28	0	**CPRI5 Transmit Control Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_TX_CTRL4 27	0	**CPRI4 Transmit Control Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_TX_CTRL3 26	0	**CPRI3 Transmit Control Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_TX_CTRL2 25	0	**CPRI2 Transmit Control Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted

Table 8-41. GIR6 Bit Descriptions

Name	Reset	Description	Settings
CPRI_TX_CTRL1 24	0	**CPRI1 Transmit Control Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
— 23–22	0	Reserved. Write to zero for future compatibility.v	
CPRI_RX_CTRL6 21	0	**CPRI6 Receive Control Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_RX_CTRL5 20	0	**CPRI5 Receive Control Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_RX_CTRL4 19	0	**CPRI4 Receive Control Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_RX_CTRL3 18	0	**CPRI3 Receive Control Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_RX_CTRL2 17	0	**CPRI2 Receive Control Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_RX_CTRL1 16	0	**CPRI1 Receive Control Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
— 15–14	0	Reserved. Write to zero for future compatibility.v	
CPRI_TX_FRAME_TIMING6 13	0	**CPRI6 Transmit Frame Timing Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_TX_FRAME_TIMING5 12	0	**CPRI5 Transmit Frame Timing Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_TX_FRAME_TIMING4 11	0	**CPRI4 Transmit Frame Timing Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_TX_FRAME_TIMING3 10	0	**CPRI3 Transmit Frame Timing Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_TX_FRAME_TIMING2 9	0	**CPRI2 Transmit Frame Timing Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_TX_FRAME_TIMING1 8	0	**CPRI1 Transmit Frame Timing Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
— 7–6	0	Reserved. Write to zero for future compatibility.v	

Table 8-41. GIR6 Bit Descriptions

Name	Reset	Description	Settings
CPRI_RX_FRAME_TIMING6 5	0	**CPRI6 Receive Frame Timing Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_RX_FRAME_TIMING5 12	0	**CPRI5 Receive Frame Timing Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_RX_FRAME_TIMING4 11	0	**CPRI4 Receive Frame Timing Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_RX_FRAME_TIMING3 10	0	**CPRI3 Receive Frame Timing Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_RX_FRAME_TIMING2 9	0	**CPRI2 Receive Frame Timing Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_RX_FRAME_TIMING1 8	0	**CPRI1 Receive Frame Timing Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted

8.2.41 General Interrupt Enable Register 6 (GIER6_x)

GIER6_0 General Interrupt Enable Register 6 for Cores 0–5 Offset 0x174
GIER6_1 Offset 0x178
GIER6_2 Offset 0x17C
GIER6_3 Offset 0x180
GIER6_4 Offset 0x184
GIER6_5 Offset 0x188

Bit	31	30	29	28	27	26	25	24
	—		CPRI_TX_CTRL6_EN	CPRI_TX_CTRL5_EN	CPRI_TX_CTRL4_EN	CPRI_TX_CTRL3_EN	CPRI_TX_CTRL2_EN	CPRI_TX_CTRL1_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—		CPRI_RX_CTRL6_EN	CPRI_RX_CTRL5_EN	CPRI_RX_CTRL4_EN	CPRI_RX_CTRL3_EN	CPRI_RX_CTRL2_EN	CPRI_RX_CTRL_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—		CPRI_TX_FRAME_TIMNG6_EN	CPRI_TX_FRAME_TIMNG5_EN	CPRI_TX_FRAME_TIMNG4_EN	CPRI_TX_FRAME_TIMNG3_EN	CPRI_TX_FRAME_TIMNG2_EN	CPRI_TX_FRAME_TIMNG1_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—		CPRI_RX_FRAME_TIMNG6_EN	CPRI_RX_FRAME_TIMNG5_EN	CPRI_RX_FRAME_TIMNG4_EN	CPRI_RX_FRAME_TIMNG3_EN	CPRI_RX_FRAME_TIMNG2_EN	CPRI_RX_FRAME_TIMNG1_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIER6_[0–5] include interrupt enable bits for cores 0–5 for interrupts defined by GIR6. GIER6_[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 8-42. GIR6_[0–5] Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.v	
CPRI_TX_CTRL6 29	0	**CPRI6 Transmit Control Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_TX_CTRL5 28	0	**CPRI5 Transmit Control Interrupt Enable** Enables/disables the interrupt.	0 Interrupt not asserted 1 Interrupt asserted

Table 8-42. GIR6_[0–5] Bit Descriptions

Name	Reset	Description	Settings
CPRI_TX_CTRL4 27	0	**CPRI4 Transmit Control Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_TX_CTRL3 26	0	**CPRI3 Transmit Control Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_TX_CTRL2 25	0	**CPRI2 Transmit Control Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_TX_CTRL1 24	0	**CPRI1 Transmit Control Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
— 23–22	0	Reserved. Write to zero for future compatibility.v	
CPRI_RX_CTRL6 21	0	**CPRI6 Receive Control Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_RX_CTRL5 20	0	**CPRI5 Receive Control Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_RX_CTRL4 19	0	**CPRI4 Receive Control Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_RX_CTRL3 18	0	**CPRI3 Receive Control Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_RX_CTRL2 17	0	**CPRI2 Receive Control Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_RX_CTRL1 16	0	**CPRI1 Receive Control Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
— 15–14	0	Reserved. Write to zero for future compatibility.v	
CPRI_TX_FRAME_TIMING6 13	0	**CPRI6 Transmit Frame Timing Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_TX_FRAME_TIMING5 12	0	**CPRI5 Transmit Frame Timing Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_TX_FRAME_TIMING4 11	0	**CPRI4 Transmit Frame Timing Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_TX_FRAME_TIMING3 10	0	**CPRI3 Transmit Frame Timing Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled

Table 8-42. GIR6_[0–5] Bit Descriptions

Name	Reset	Description	Settings
CPRI_TX_FRAME_TIMING2 9	0	**CPRI2 Transmit Frame Timing Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_TX_FRAME_TIMING1 8	0	**CPRI1 Transmit Frame Timing Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
— 7–6	0	Reserved. Write to zero for future compatibility.v	
CPRI_RX_FRAME_TIMING6 5	0	**CPRI6 Receive Frame Timing Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_RX_FRAME_TIMING5 4	0	**CPRI5 Receive Frame Timing Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_RX_FRAME_TIMING4 3	0	**CPRI4 Receive Frame Timing Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_RX_FRAME_TIMING3 2	0	**CPRI3 Receive Frame Timing Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_RX_FRAME_TIMING2 1	0	**CPRI2 Receive Frame Timing Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_RX_FRAME_TIMING1 0	0	**CPRI1 Receive Frame Timing Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled

8.2.42 General Interrupt Register 7 (GIR7)

GIR7 General Interrupt Register 7 Offset 0x194

Bit	31	30	29	28	27	26	25	24
	EMSG_QM_TX7	EMSG_QM_TX6	EMSG_QM_TX5	EMSG_QM_TX4	EMSG_QM_TX3	EMSG_QM_TX2	EMSG_QM_TX1	EMSG_QM_TX0
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EMSG_QM_RX7	EMSG_QM_RX6	EMSG_QM_RX5	EMSG_QM_RX4	EMSG_QM_RX3	EMSG_QM_RX2	EMSG_QM_RX1	EMSG_QM_RX0
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EMSG_BM7	EMSG_BM6	EMSG_BM5	EMSG_BM4	EMSG_BM3	EMSG_BM2	EMSG_BM1	EMSG_BM0
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—			EMSG_BM_GEN_ERR	EMSG_QM_GEN_ERR	—	EMSG_GEN_ERR	CPRI_GEN_ERR
Type	R							
Reset	0	0	0	0	0	0	0	0

GIR7 includes interrupt status of some internal events within MSC8157E. Those bits are not sticky, but only sample events. GIR7 is reset by a hard reset event. All bits are cleared on reset.

Table 8-43. GIR7 Bit Descriptions

Name	Reset	Description	Settings
EMSG_QM_TX7 31	0	**EMSG QM TX7 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_TX6 30	0	**EMSG QM TX6 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_TX5 29	0	**EMSG QM TX5 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_TX4 28	0	**EMSG QM TX4 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_TX3 27	0	**EMSG QM TX3 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_TX2 26	0	**EMSG QM TX2 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_TX1 25	0	**EMSG QM TX1 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted

Table 8-43. GIR7 Bit Descriptions

Name	Reset	Description	Settings
EMSG_QM_TX0 24	0	**EMSG QM TX0 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_RX7 23	0	**EMSG QM RX7 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_RX6 22	0	**EMSG QM RX6 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_RX5 21	0	**EMSG QM RX5 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_RX4 20	0	**EMSG QM RX4 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_RX3 19	0	**EMSG QM RX3 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_RX2 18	0	**EMSG QM RX2 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_RX1 17	0	**EMSG QM RX1 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_RX0 16	0	**EMSG QM RX0 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_BM7 15	0	**EMSG BM7 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_BM6 14	0	**EMSG BM6 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_BM5 13	0	**EMSG BM5 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_BM4 12	0	**EMSG BM4 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_BM3 11	0	**EMSG BM3 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_BM2 10	0	**EMSG BM2 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_BM1 9	0	**EMSG BM1 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_BM0 8	0	**EMSG BM0 Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
— 7–5	0	Reserved. Write to zero for future compatibility.v	
EMSG_BM_GEN_ERR 4	0	**EMSG Buffer Manager General Error Interrupt** Reflects the status of the EMSG buffer manager general error interrupt.	0 Interrupt not asserted 1 Interrupt asserted

Table 8-43. GIR7 Bit Descriptions

Name	Reset	Description	Settings
EMSG_QM_GEN_ERR 3	0	**EMSG Queue Manager General Error Interrupt** Reflects the status of the EMSG queue manager general error interrupt.	0 Interrupt not asserted 1 Interrupt asserted
— 2	0	Reserved. Write to zero for future compatibility.v	
EMSG_GEN_ERR 1	0	**EMSG General Error Interrupt** Reflects the status of the EMSG general error interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CPRI_GEN_ERR 0	0	**CPRI General Error Interrupt** Reflects the status of the CPRI general error interrupt.	0 Interrupt not asserted 1 Interrupt asserted

8.2.43 General Interrupt Enable Register 7 (GIER7_x)

GIER7_0 General Interrupt Enable Register 7 for Cores 0–5 Offset 0x198
GIER7_1 Offset 0x19C
GIER7_2 Offset 0x1A0
GIER7_3 Offset 0x1A4
GIER7_4 Offset 0x1A8
GIER7_5 Offset 0x1AC

Bit	31	30	29	28	27	26	25	24
	EMSG_QM_TX7_EN	EMSG_QM_TX6_EN	EMSG_QM_TX5_EN	EMSG_QM_TX4_EN	EMSG_QM_TX3_EN	EMSG_QM_TX2_EN	EMSG_QM_TX1_EN	EMSG_QM_TX0_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EMSG_QM_RX7_EN	EMSG_QM_RX6_EN	EMSG_QM_RX5_EN	EMSG_QM_RX4_EN	EMSG_QM_RX3_EN	EMSG_QM_RX2_EN	EMSG_QM_RX1_EN	EMSG_QM_RX0_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EMSG_BM7_EN	EMSG_BM6_EN	EMSG_BM5_EN	EMSG_BM4_EN	EMSG_BM3_EN	EMSG_BM2_EN	EMSG_BM1_EN	EMSG_BM0_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—			EMSG_BM_GEN_ERR_EN	EMSG_QM_GEN_ERR_EN	—	EMSG_GEN_ERR_EN	—
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIER7_[0–5] include interrupt enable bits for cores 0–5 for interrupts defined by GIR7. GIER7_[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 8-44. GIR7_[0–5] Bit Descriptions

Name	Reset	Description	Settings
EMSG_QM_TX7_EN 31	0	**EMSG QM TX7 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_TX6_EN 30	0	**EMSG QM TX6 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_TX5_EN 29	0	**EMSG QM TX5 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_TX4_EN 28	0	**EMSG QM TX4 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_TX3_EN 27	0	**EMSG QM TX3 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_TX2_EN 26	0	**EMSG QM TX2 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_TX1_EN 25	0	**EMSG QM TX1 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_TX0_EN 24	0	**EMSG QM TX0 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX7_EN 23	0	**EMSG QM RX7 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX6_EN 22	0	**EMSG QM RX6 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX5_EN 21	0	**EMSG QM RX5 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX4_EN 20	0	**EMSG QM RX4 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX3_EN 19	0	**EMSG QM RX3 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX2_EN 18	0	**EMSG QM RX2 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX1_EN 17	0	**EMSG QM RX1 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX0_EN 16	0	**EMSG QM RX0 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled

Table 8-44. GIR7_[0–5] Bit Descriptions

Name	Reset	Description	Settings
EMSG_BM7_EN 15	0	**EMSG BM7 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_BM6_EN 14	0	**EMSG BM6 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_BM5_EN 13	0	**EMSG BM5 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_BM4_EN 12	0	**EMSG BM4 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_BM3_EN 11	0	**EMSG BM3 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_BM2_EN 10	0	**EMSG BM2 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_BM1_EN 9	0	**EMSG BM1 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_BM0_EN 8	0	**EMSG BM0 Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
— 7–5	0	Reserved. Write to zero for future compatibility.v	
EMSG_BM_GEN_ERR_EN 4	0	**EMSG BM General Error Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_GEN_ERR_EN 3	0	**EMSG QM General Error Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
— 2	0	Reserved. Write to zero for future compatibility.v	
EMSG_GEN_ERR_EN 1	0	**EMSG General Error Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI_GEN_ERR_EN 0	0	**CPRI General Error Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled

8.2.44 DDR View Through L2 Memory Core Subsystems 0–3 (L2MAP_0_3)

L2MAP_0_3 DDR View Through L2 Memory Core Subsystems 0–3 Offset 0x1B8

Bit	31	30	29	28	27	26	25	24
	—		L2MAP_3					
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	—		L2MAP_2					
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	—		L2MAP_1					
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—		L2MAP_0					
Type	R/W							
Reset	0	0	0	0	0	0	0	0

L2MAP_0_3 controls the windows used by the L2 cache for each of the core subsystems 0–3. The window seen for a specific core subsystem is defined by the following formula:.

$$\text{Core Subsystem Window n} = 40000000 + (\text{L2MAP\_n} \times 32\ \text{MB}) - (41\text{FFFFFF} + (\text{L2MAP\_n} \times 32\ \text{MB}))$$

n is valid for 0 to 15, n is reserved for 16 to 63.

All bits are cleared on reset.

Table 8-45. L2MAP_0_3 Bit Descriptions

Name	Reset	Description
— 31–30	0	Reserved. Write to zero for future compatibility.
L2MAP_3 29–24	0	**L2 Map for Core Subsystem 3** Defines the core subsystem window seen by the DDR memory through the L2 cache.
— 23–22	0	Reserved. Write to zero for future compatibility.
L2MAP_2 21–16	0	**L2 Map for Core Subsystem 2** Defines the core subsystem window seen by the DDR memory through the L2 cache.
— 15–14	0	Reserved. Write to zero for future compatibility.
L2MAP_1 13–8	0	**L2 Map for Core Subsystem 1** Defines the core subsystem window seen by the DDR memory through the L2 cache.
— 7–6	0	Reserved. Write to zero for future compatibility.
L2MAP_0 5–0	0	**L2 Map for Core Subsystem 0** Defines the core subsystem window seen by the DDR memory through the L2 cache.

8.2.45 DDR View Through L2 Memory Core Subsystems 4–5 (L2MAP_4_5)

L2MAP_4_5 DDR View Through L2 Memory Core Subsystems 4–5 Offset 0x1BC

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
	—		L2MAP_5					
Type	R.W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—		L2MAP_4					
Type	R/W							
Reset	0	0	0	0	0	0	0	0

L2MAP_4_5 controls the windows used by the L2 cache for each of the core subsystems 4–5. The window seen for a specific core subsystem is defined by the following formula:.

$$\text{Core Subsystem Window n} = 40000000 + (\text{L2MAP\_n} \times 32\ \text{MB}) - (41\text{FFFFFF} + (\text{L2MAP\_n} \times 32\ \text{MB}))$$

n is valid for 0 to 15, n is reserved for 16 to 63.

All bits are cleared on reset.

Table 8-46. L2MAP_4_5 Bit Descriptions

Name	Reset	Description
— 31–14	0	Reserved. Write to zero for future compatibility.
L2MAP_5 13–8	0	**L2 Map for Core Subsystem 1** Defines the core subsystem window seen by the DDR memory through the L2 cache.
— 7–6	0	Reserved. Write to zero for future compatibility.
L2MAP_4 5–0	0	**L2 Map for Core Subsystem 0** Defines the core subsystem window seen by the DDR memory through the L2 cache.

8.2.46 eMSG to QUICC Engine External Request Enable (CPCEER)

CPCEER eMSG to QUICC Engine External Request Enable Offset 0x1DC

Bit	31	30	29	28	27	26	25	24
	EMSG_QM_TX7_QE_EN_1	EMSG_QM_TX6_QE_EN_1	EMSG_QM_TX5_QE_EN_1	EMSG_QM_TX4_QE_EN_1	EMSG_QM_TX3_QE_EN_1	EMSG_QM_TX2_QE_EN_1	EMSG_QM_TX1_QE_EN_1	EMSG_QM_TX0_QE_EN_1
Type	R/W							
Reset	1	1	1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	EMSG_QM_TX7_QE_EN_0	EMSG_QM_TX6_QE_EN_0	EMSG_QM_TX5_QE_EN_0	EMSG_QM_TX4_QE_EN_0	EMSG_QM_TX3_QE_EN_0	EMSG_QM_TX2_QE_EN_0	EMSG_QM_TX1_QE_EN_0	EMSG_QM_TX0_QE_EN_0
Type	R/W							
Reset	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	EMSG_QM_RX7_QE_EN_1	EMSG_QM_RX6_QE_EN_1	EMSG_QM_RX5_QE_EN_1	EMSG_QM_RX4_QE_EN_1	EMSG_QM_RX3_QE_EN_1	EMSG_QM_RX2_QE_EN_1	EMSG_QM_RX1_QE_EN_1	EMSG_QM_RX0_QE_EN_1
Type	R/W							
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	EMSG_QM_RX7_QE_EN_0	EMSG_QM_RX6_QE_EN_0	EMSG_QM_RX5_QE_EN_0	EMSG_QM_RX4_QE_EN_0	EMSG_QM_RX3_QE_EN_0	EMSG_QM_RX2_QE_EN_0	EMSG_QM_RX1_QE_EN_0	EMSG_QM_RX0_QE_EN_0
Type	R/W							
Reset	1	1	1	1	1	1	1	1

CPCEER includes interrupt enable bits for the eMSG interrupts to the QUICC Engine external request inputs 0 and 1. All bits are set on reset.

Table 8-47. CPCEER Bit Descriptions

Name	Reset	Description	Settings
EMSG_QM_TX7_QE_ EN_1 31	1	**EMSG QM TX7 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_TX6_QE_ EN_1 30	1	**EMSG QM TX6 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_TX5_QE_ EN_1 29	1	**EMSG QM TX5 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_TX4_QE_ EN_1 28	1	**EMSG QM TX4 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled

Table 8-47. CPCEER Bit Descriptions

Name	Reset	Description	Settings
EMSG_QM_ TX3_QE_ EN_1 27	1	**EMSG QM TX3 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ TX2_QE_ EN_1 26	1	**EMSG QM TX2 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ TX1_QE_ EN_1 25	1	**EMSG QM TX1 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ TX0_QE_ EN_1 24	1	**EMSG QM TX0 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ RX7_QE_ EN_1 23	1	**EMSG QM RX7 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ RX6_QE_ EN_1 22	1	**EMSG QM RX6 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ RX5_QE_ EN_1 21	1	**EMSG QM RX5 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ RX4_QE_ EN_1 20	1	**EMSG QM RX4 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ RX3_QE_ EN_1 19	1	**EMSG QM RX3 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ RX2_QE_ EN_1 18	1	**EMSG QM RX2 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ RX1_QE_ EN_1 17	1	**EMSG QM RX1 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ RX0_QE_ EN_1 16	1	**EMSG QM RX0 Interrupt to QUICC Engine External Request 1 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ TX7_QE_ EN_0 15	1	**EMSG QM TX7 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ TX6_QE_ EN_0 14	1	**EMSG QM TX6 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ TX5_QE_ EN_0 13	1	**EMSG QM TX5 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
EMSG_QM_ TX4_QE_ EN_0 12	1	**EMSG QM TX4 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_ TX3_QE_ EN_0 11	1	**EMSG QM TX3 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled

Table 8-47. CPCEER Bit Descriptions

Name	Reset	Description	Settings
EMSG_QM_TX2_QE_ EN_0 10	1	**EMSG QM TX2 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_TX1_QE_ EN_0 9	1	**EMSG QM TX1 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_TX0_QE_ EN_0 8	1	**EMSG QM TX0 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX7_QE_ EN_0 7	1	**EMSG QM RX7 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX6_QE_ EN_0 6	1	**EMSG QM RX6 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX5_QE_ EN_0 5	1	**EMSG QM RX5 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX4_QE_ EN_0 4	1	**EMSG QM RX4 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX3_QE_ EN_0 3	1	**EMSG QM RX3 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX2_QE_ EN_0 2	1	**EMSG QM RX2 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX1_QE_ EN_0 1	1	**EMSG QM RX1 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
EMSG_QM_RX0_QE_ EN_0 0	1	**EMSG QM RX0 Interrupt to QUICC Engine External Request 0 Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled

8.2.47 RGMII1 High Resolution Delay Register (UCC1_DELAY_HR)

UCC1_DELAY_HR RGMII1 High Resolution Delay Register Offset 0x1E0

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
	—				UCC1RCLK1_BYPASSSHRT_2	UCC1RCLK1_BYPASS190_1	UCC1RCLK1_BYPASSMED_2	UCC1RCLK1_BYPASSMED_1
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	UCC1TCLK1_BYPASSSHRT_2	UCC1TCLK1_BYPASS190_1	UCC1TCLK1_BYPASSMED_2	UCC1TCLK1_BYPASSMED_1	UCC1CLKOD_BYPASSSHRT_2	UCC1CLKOD_BYPASS190_1	UCC1CLKOD_BYPASSMED_2	UCC1CLKOD_BYPASSMED_1
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	UCC1RXDD_BYPASSSHRT_2	UCC1RXDD_BYPASS190_1	UCC1RXDD_BYPASSMED_2	UCC1RXDD_BYPASSMED_1	UCC1TXDD_BYPASSSHRT_2	UCC1TXDD_BYPASS190_1	UCC1TXDD_BYPASSMED_2	UCC1TXDD_BYPASSMED_1
Type	R/W							
Reset	0	0	0	0	0	0	0	0

UCC1_DELAY_HR allows fine-tuning of the clock and data signal delays for RGMII1. These adjustments are added to the values supplied in GCR4. All bits are cleared on reset.

Table 8-48. UCC1_DELAY_HR Bit Descriptions

Name	Reset	Description	Settings
— 31–20	0	Reserved. Write to zero for future compatibility.	
UCC1RCLK1_BYPASSSHRT_2 19	0	**UCC1 RX Clock In Delay—Bypass Short 2** Allows you to include/exclude a short tap delay to the RGMII1 RX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1RCLK1_BYPASSSHRT_1 18	0	**UCC1 RX Clock In Delay—Bypass Short 1** Allows you to include/exclude a short tap delay to the RGMII1 RX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1RCLK1_BYPASSMED_2 17	0	**UCC1 RX Clock In Delay—Bypass Medium 2** Allows you to include/exclude a medium tap delay to the RGMII1 RX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1RCLK1_BYPASSMED_1 16	0	**UCC1 RX Clock In Delay—Bypass Medium 1** Allows you to include/exclude a medium tap delay to the RGMII1 RX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1TCLK1_BYPASSSHRT_2 15	0	**UCC1 TX Clock In Delay—Bypass Short 2** Allows you to include/exclude a short tap delay to the RGMII1 TX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.

Table 8-48. UCC1_DELAY_HR Bit Descriptions (Continued)

Name	Reset	Description	Settings
UCC1TCLK1_ BYPASSSHRT_1 14	0	**UCC1 TX Clock In Delay—Bypass Short 1** Allows you to include/exclude a short tap delay to the RGMII1 TX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1TCLK1_ BYPASSMED_2 13	0	**UCC1 TX Clock In Delay—Bypass Medium 2** Allows you to include/exclude a medium tap delay to the RGMII1 TX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1TCLK1_ BYPASSMED_1 12	0	**UCC1 TX Clock In Delay—Bypass Medium 1** Allows you to include/exclude a medium tap delay to the RGMII1 TX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1CLKOD_ BYPASSSHRT_2 11	0	**UCC1 Clock Out Delay—Bypass Short 2** Allows you to include/exclude a short tap delay to the RGMII1 clock-out delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1CLKOD_ BYPASSSHRT_1 10	0	**UCC1 Clock Out Delay—Bypass Short 1** Allows you to include/exclude a short tap delay to the RGMII1 clock-out delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1CLKOD_ BYPASSMED_2 9	0	**UCC1 Clock Out Delay—Bypass Medium 2** Allows you to include/exclude a medium tap delay to the RGMII1 clock-out delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1CLKOD_ BYPASSMED_1 8	0	**UCC1 Clock Out Delay—Bypass Medium 1** Allows you to include/exclude a medium tap delay to the RGMII1 clock-out delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1RXDD_ BYPASSSHRT_2 7	0	**UCC1 RX Data Delay—Bypass Short 2** Allows you to include/exclude a short tap delay to the RX data delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1RXDD_ BYPASSSHRT_1 6	0	**UCC1 RX Data Delay—Bypass Short 1** Allows you to include/exclude a short tap delay to the RX data delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1RXDD_ BYPASSMED_2 5	0	**UCC1 RX Data Delay—Bypass Medium 2** Allows you to include/exclude a medium tap delay to the RX data delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1RXDD_ BYPASSMED_1 4	0	**UCC1 RX Data Delay—Bypass Medium 1** Allows you to include/exclude a medium tap delay to the RX data delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1TXDD_ BYPASSSHRT_2 3	0	**UCC1 TX Data Delay—Bypass Short 2** Allows you to include/exclude a short tap delay to the TX data delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1TXDD_ BYPASSSHRT_1 2	0	**UCC1 TX Data Delay—Bypass Short 1** Allows you to include/exclude a short tap delay to the TX data delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1TXDD_ BYPASS250_2 1	0	**UCC1 TX Data Delay—Bypass Medium 2** Allows you to include/exclude a medium tap delay to the TX data delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC1TXDD_ BYPASSMED_1 0	0	**UCC1 TX Data Delay—Bypass Medium 1** Allows you to include/exclude a medium tap delay to the TX data delay.	0 Enables the tap delay. 1 Disables the tap delay.

Note: There are four high resolution tap delays that can be added for each signal line: two short delays and two medium delays. The sum of the stow short delays plus the two medium delays is equal to one long delay (see **Table 8-12** for details on how the delays are combined).

8.2.48 RGMII2 High Resolution Delay Register (UCC3_DELAY_HR)

UCC3_DELAY_HR RGMII2 High Resolution Delay Register Offset 0x1E4

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
	—				UCC3RCLK1_BYPASSSHRT_2	UCC3RCLK1_BYPASSSHRT_1	UCC3RCLK1_BYPASSMED_2	UCC3RCLK1_BYPASSMED_1
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	UCC3TCLK1_BYPASSSHRT_2	UCC3TCLK1_BYPASSSHRT_1	UCC3TCLK1_BYPASSMED_2	UCC3TCLK1_BYPASSMED_1	UCC3CLKOD_BYPASSSHRT_2	UCC3CLKOD_BYPASSSHRT_1	UCC3CLKOD_BYPASSMED_2	UCC3CLKOD_BYPASSMED_1
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	UCC3RXDD_BYPASSSHRT_2	UCC3RXDD_BYPASSSHRT_1	UCC3RXDD_BYPASSMED_2	UCC3RXDD_BYPASSMED_1	UCC3TXDD_BYPASSSHRT_2	UCC3TXDD_BYPASSSHRT_1	UCC3TXDD_BYPASSMED_2	UCC3TXDD_BYPASSMED_1
Type	R/W							
Reset	0	0	0	0	0	0	0	0

UCC3_DELAY_HR allows fine-tuning of the clock and signal delays for RGMII1. These adjustments are added to the values supplied in GCR4. All bits are cleared on reset.

Table 8-49. UCC3_DELAY_HR Bit Descriptions

Name	Reset	Description	Settings
— 31–20	0	Reserved. Write to zero for future compatibility.	
UCC3RCLK1_BYPASSSHRT_2 19	0	**UCC3 RX Clock In Delay—Bypass Short 2** Allows you to include/exclude a short tap delay to the RGMII2 RX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3RCLK1_BYPASSSHRT_1 18	0	**UCC3 RX Clock In Delay—Bypass Short 1** Allows you to include/exclude a short tap delay to the RGMII2 RX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3RCLK1_BYPASSMED_2 17	0	**UCC3 RX Clock In Delay—Bypass Medium 2** Allows you to include/exclude a medium tap delay to the RGMII2 RX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3RCLK1_BYPASSMED_1 16	0	**UCC3 RX Clock In Delay—Bypass Medium 1** Allows you to include/exclude a medium tap delay to the RGMII2 RX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3TCLK1_BYPASSSHRT_2 15	0	**UCC3 TX Clock In Delay—Bypass Short 2** Allows you to include/exclude a short tap delay to the RGMII2 TX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.

Table 8-49. UCC3_DELAY_HR Bit Descriptions (Continued)

Name	Reset	Description	Settings
UCC3TCLK1_ BYPASSSHRT_1 14	0	**UCC3 TX Clock In Delay—Bypass Short 1** Allows you to include/exclude a short tap delay to the RGMII2 TX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3TCLK1_ BYPASSMED_2 13	0	**UCC3 TX Clock In Delay—Bypass Medium 2** Allows you to include/exclude a medium tap delay to the RGMII2 TX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3TCLK1_ BYPASSMED_1 12	0	**UCC3 TX Clock In Delay—Bypass Medium 1** Allows you to include/exclude a medium tap delay to the RGMII2 TX clock-in delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3CLKOD_ BYPASSSHRT_2 11	0	**UCC3 Clock Out Delay—Bypass Short 2** Allows you to include/exclude a short tap delay to the RGMII2 clock-out delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3CLKOD_ BYPASSSHRT_1 10	0	**UCC3 Clock Out Delay—Bypass Short 1** Allows you to include/exclude a short tap delay to the RGMII2 clock-out delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3CLKOD_ BYPASSMED_2 9	0	**UCC3 Clock Out Delay—Bypass Medium 2** Allows you to include/exclude a medium tap delay to the RGMII2 clock-out delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3CLKOD_ BYPASSMED_1 8	0	**UCC3 Clock Out Delay—Bypass Medium 1** Allows you to include/exclude a medium tap delay to the RGMII2 clock-out delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3RXDD_ BYPASSSHRT_2 7	0	**UCC3 RX Data Delay—Bypass Short 2** Allows you to include/exclude a short tap delay to the RX data delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3RXDD_ BYPASSSHRT_1 6	0	**UCC3 RX Data Delay—Bypass Short 1** Allows you to include/exclude a short tap delay to the RX data delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3RXDD_ BYPASSMED_2 5	0	**UCC3 RX Data Delay—Bypass Medium 2** Allows you to include/exclude a medium tap delay to the RX data delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3RXDD_ BYPASSMED_1 4	0	**UCC3 RX Data Delay—Bypass Medium 1** Allows you to include/exclude a medium tap delay to the RX data delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3TXDD_ BYPASSSHRT_2 3	0	**UCC3 TX Data Delay—Bypass Short 2** Allows you to include/exclude a short tap delay to the TX data delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3TXDD_ BYPASSSHRT_1 2	0	**UCC3 TX Data Delay—Bypass Short 1** Allows you to include/exclude a short tap delay to the TX data delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3TXDD_ BYPASSMED_2 1	0	**UCC3 TX Data Delay—Bypass Medium 2** Allows you to include/exclude a medium tap delay to the TX data delay.	0 Enables the tap delay. 1 Disables the tap delay.
UCC3TXDD_ BYPASSMED_1 0	0	**UCC3 TX Data Delay—Bypass Medium 1** Allows you to include/exclude a medium tap delay to the TX data delay.	0 Enables the tap delay. 1 Disables the tap delay.

Note: There are four high resolution tap delays that can be added for each signal line: two short delays and two medium delays. The sum of the stow short delays plus the two medium delays is equal to one long delay (see **Table 8-12** for details on how the delays are combined).

8.2.49 General Interrupt Register 8 (GIR8)

GIR8 General Interrupt Register 8 Offset 0x1E8

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
	—					CPRI3	CPRI2	CPRI1
Type	R							
Reset	0	0	0	0	0	0	0	0

GIR8 includes the CPRI to MAPLE interrupt status for the three CPRI interrupts. Those bits are not sticky, but only sample events. GIR8 is reset by a hard reset event. All bits are cleared on reset.

Table 8-50. GIR8 Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.v	
CPRI3 2	0	**CPRI to MAPLE Interrupt 3** Reflects the status of the interrupt. CPRI3 indicates a chunk (16 chips) detected.	0 Interrupt not asserted 1 Interrupt asserted
CPRI2 1	0	**CPRI to MAPLE Interrupt 2** Reflects the status of the interrupt. CPRI2 indicates a frame boundary detected.	0 Interrupt not asserted 1 Interrupt asserted
CPRI1 0	0	**CPRI to MAPLE Interrupt 1** Reflects the status of the interrupt. CPRI1 indicates as sub-slot (256 chips) detected.	0 Interrupt not asserted 1 Interrupt asserted

8.2.50 CPRI to MAPLE External Request Enable (MAPLE_EXT_REQ_EN_1)

MAPLE_EXT_REQ_EN_1 CPRI to MAPLE External Request Enable Offset 0x1F0

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
	—					CPRI3_MPL_EN	CPRI2_MPL_EN	CPRI1_MPL_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

MAPLE_EXT_REQ_EN_1 includes interrupt enable bits for the CPRI to the MAPLE interrupt external request input. All bits are cleared on reset.

Table 8-51. MAPLE_EXT_REQ_EN_1 Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.v	
CPRI3_MPL_ EN 2	0	**CPRI to MAPLE Interrupt 3 External Request Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI2_MPL_ EN 1	0	**CPRI to MAPLE Interrupt 2 External Request Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled
CPRI1_MPL_ EN 0	0	**CPRI to MAPLE Interrupt 1 External Request Enable** Enables/disables the interrupt.	0 Interrupt disabled 1 Interrupt enabled

9 Memory Map

This section describes the memory map of MSC8157E.

The MSC8157E incorporates four address spaces:

- Shared memory (M3, DDR, QUICC Engine subsystem, boot ROM, and MAPLE-B2) address space.
- Shared SC3850 DSP core subsystem M2/L2 memories
- SC3850 DSP core subsystem internal address space, 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 is identical to the MSC8156 DSP family, except for the MAPLE-B2, which uses 4 MB instead of 1 MB; M3 memory which was enlarged from 1 MB to 3 MB; and the second DDR controller, which was eliminated. The memory space includes the M3 memory, DDR memory, MAPLE-B2, 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
0x40000000–0xBFFFFFFF	DDR Memory (default value)	2 G
0xC0000000–0xC02FFFFF	M3 Memory	3 M
0xC0300000–0xCFFFFFFF	Reserved	253 M
0xD0000000–0xD09FFFF	Reserved (used for M3 memory in the MSC8144 DSP)	10 M
0xD0A00000–0xDFFFFFFF	Reserved	246 M
0xE0000000–0xE03FFFFF	MAPLE-B2	4 M
0xE0400000–0xFEDFFFFF	Reserved	490 M
0xFEE00000–0xFEE3FFFF	QUICC Engine Subsystem	256 K
0xFEE40000–0xFEEFFFFF	Reserved (QUICC Engine Subsystem)	768 K
0xFEF00000–0xFEF17FFF	Boot ROM	96 K

Table 9-1. Shared Memory Address Space (Continued)

Address	Purpose	Size in Bytes
0xFEF18000–0xFEFFFFFF	Reserved	928 K
0xFF000000–0xFFF0FFFF	SC3850 subsystem internal address space	15 M + 64 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 all other cores. Other cores can access another core *n* in accordance with the following policy:

- Core subsystem 0 can access the M2/L2 memory of core subsystems 0, 2, 3, 4, and 5.
- Core subsystem 1 can access the M2/L2 memory of core subsystems 1, 2, 3, 4, and 5.
- Core subsystem 2 can access the M2/L2 memory of core subsystems 2, 4, and 5.
- Core subsystem 3 can access the M2/L2 memory of core subsystems 3, 4, and 5.
- Core subsystem 4 can only access the M2/L2 memory of core subsystem 4 (itself).
- Core subsystem 5 can only access the M2/L2 memory of core subsystem 5 (itself).

Although the M2 memories have different physical addresses, all cores use a virtual address of 0x0 to access their own M2 memory and uses MMU translation to access the correct memory space.

When partitioned as L2 cache memory, the L2 cache can only be accessed by the core to which it belongs and by other masters in the system, but not by all other cores. Other cores can access a core *n* in accordance with the following policy:

- Core subsystem 0 can access the L2 memory of core subsystems 0, 2, 3, 4, and 5.
- Core subsystem 1 can access the L2 memory of core subsystems 1, 2, 3, 4, and 5.
- Core subsystem 2 can access the L2 memory of core subsystems 2, 4, and 5.
- Core subsystem 3 can access the L2 memory of core subsystems 3, 4, and 5.
- Core subsystem 4 can only access the L2 memory of core subsystem 4 (itself).
- Core subsystem 5 can only access the L2 memory of core subsystem 5 (itself).

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 is shown in **Table 9-2** which indicates the total DDR access range.

Each core can access a 32 MB window in the DDR memory. The memory space between 0x40000000 and 0xBFFFFFFF is partitioned into 64 non-overlapping ranges of 32 MB each. The window is mapped to each core subsystem using the L2MAP_0_3 and L2MAP_4_5 general configuration registers. See **Section 8.2.44**, *DDR View Through L2 Memory Core Subsystems 0–3 (L2MAP_0_3)*, on page 8-79 and **Section 8.2.45**, *DDR View Through L2 Memory Core Subsystems 4–5 (L2MAP_4_5)*, on page 8-80 for details.

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 DDR which uses the address range (0x40000000 + Y × 32 MB) to (0x41FFFFFF + Y × 32 MB), where Y is a 6-bit integer value configured by a register in the general configuration register space. Thus, the memory space between 0x40000000 and 0xBFFFFFFF is partitioned into 64 non-overlapping ranges, each 32 MB is size. The mapping is different for each DSP subsystems. The default setting is Y = 0 for all cores, that is, this address range is always mapped to the first 32 MB of DDR memory, which uses the address range 0x40000000–0x41FFFFFF. 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	Comments
0x20000000–0x23FFFFFF	Reserved	64 M	—
0x24000000–0x25FFFFFF	DDR through Core 5 L2	32 M	Seen by the L2 Cache as (40000000 + Y × 32 MB) to (41FFFFFF + Y × 32 MB). Y is the Core5 slave port 6-bit mapping from the GCR. Y = 0 to 63 and mapped to the DDR.

Table 9-2. Shared M2/L2 Memories Address Space (Continued)

Address	Purpose	Size in Bytes	Comments
0x26000000–0x27FFFFFF	DDR through Core 4 L2	32 M	Seen by the L2 Cache as (40000000 + Y × 32 MB) to (41FFFFFF + Y × 32 MB). Y is the Core4 slave port 6-bit mapping from the GCR. Y = 0 to 63 and mapped to the DDR.
0x28000000–0x29FFFFFF	DDR through Core 3 L2	32 M	Seen by the L2 Cache as (40000000 + Y × 32 MB) to (41FFFFFF + Y × 32 MB). Y is the Core3 slave port 6-bit mapping from the GCR. Y = 0 to 63 and mapped to the DDR.
0x2A000000–0x2BFFFFFF	DDR through Core 2 L2	32 M	Seen by the L2 Cache as (40000000 + Y × 32 MB) to (41FFFFFF + Y × 32 MB). Y is the Core2 slave port 6-bit mapping from the GCR. Y = 0 to 63 and mapped to the DDR.
0x2C000000–0x2DFFFFFF	DDR through Core 1 L2	32 M	Seen by the L2 Cache as (40000000 + Y × 32 MB) to (41FFFFFF + Y × 32 MB). Y is the Core1 slave port 6-bit mapping from the GCR. Y = 0 to 63 and mapped to the DDR.
0x2E000000–0x2FFFFFFF	DDR through Core 0 L2	32 M	Seen by the L2 Cache as (40000000 + Y × 32 MB) to (41FFFFFF + Y × 32 MB). Y is the Core0 slave port 6-bit mapping from the GCR. Y = 0 to 63 and mapped to the DDR.
0x30000000–0x3007FFFF	Core0 M2 memory	512 K max.	Not all of the 512 KB are always allocated as M2. The allocation and illegal access detection are controlled by the core subsystem.
0x30080000–0x30FFFFFF	Reserved	16 M – 512 K	
0x31000000–0x3107FFFF	Core 1 M2 memory	512 K max.	Not all of the 512 KB are always allocated as M2. The allocation and illegal access detection are controlled by the core subsystem.
0x31080000–0x31FFFFFF	Reserved	16 M – 512 K	
0x32000000–0x3207FFFF	Core 2 M2 memory	512 K max.	Not all of the 512 KB are always allocated as M2. The allocation and illegal access detection are controlled by the core subsystem.
0x32080000–0x32FFFFFF	Reserved	16 M – 512 K	
0x33000000–0x3307FFFF	Core 3 M2 memory	512 K max.	Not all of the 512 KB are always allocated as M2. The allocation and illegal access detection are controlled by the core subsystem.
0x33080000–0x33FFFFFF	Reserved	16 M – 512 K	
0x34000000–0x3407FFFF	Core 4 M2 memory	512 K max.	Not all of the 512 KB are always allocated as M2. The allocation and illegal access detection are controlled by the core subsystem.

Table 9-2. Shared M2/L2 Memories Address Space (Continued)

Address	Purpose	Size in Bytes	Comments
0x34080000–0x34FFFFFF	Reserved	16 M – 512 K	
0x35000000–0x3507FFFF	Core 5 M2 memory	512 K max.	Not all of the 512 KB are always allocated as M2. The allocation and illegal access detection are controlled by the core subsystem.
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
0xFFF0A400–0xFFF0EFFF	Reserved	19 K	
0xFFF0F0000–0xFFF0F3FF	L2 cache controller	1 K	Supervisor
0xFFF0F400–0xFFF0FFFF	Reserved	3 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 *SC3850 DSP Subsystem Reference Manual* for details.

9.4 CCSR Address Space

The MSC8157E CCSR is mapped within a contiguous block of memory starting as 0xFFF10000. The size of the CCSR in MSC8157E 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	CLASS0	4 K
0xFFF19000–0xFFF1BFFF	Reserved	12 K

Table 9-4. CCSR Address Space (Continued)

Address	Purpose	Size (Bytes)
0xFFF1C000–0xFFF1FFFF	MAPLE-B2	16 K
0xFFF20000–0xFFF21FFF	DDR Controller	8 K
0xFFF22000–0xFFF23FFF	Reserved.	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	Timer0	256
0xFFF26100–0xFFF261FF	Timer1	256
0xFFF26200–0xFFF262FF	Timer2	256
0xFFF26300–0xFFF263FF	Timer3	256
0xFFF26400–0xFFF265FF	Timer_32b_0	512
0xFFF26600–0FFF267FF	Timer_32b_1	512
0xFFF26800–0xFFF26BFF	Reserved	1K
0xFFF26C00–0xFFF26C3F	UART	64
0xFFF26C40–0xFFF26FFF	Reserved	1K – 64
0xFFF27000–0xFFF270FF	GIC	256
0xFFF27100–0xFFF271FF	Hardware Semaphores	256
0xFFF27200–0xFFF272FF	GPIO	256
0xFFF27300–0xFFF27FFF	Reserved	4 K – 768
0xFFF28000–0xFFF281FF	General Configuration Registers	512
0xFFF28200–0xFFF3FFFF	Reserved	96 K – 512
0xFFF40000–0xFFF5FFFF	CPRI Registers	128 K
0xFFF60000–0xFFF7FFFF	eMSG Registers	128 K
0xFFF80000–0xFFF9FFFF	RapidIO Registers	128 K
0xFFFA0000–0xFFFA0FFF	Reserved	4 K
0xFFFA1000–0xFFFA103F	OCN Crossbar Switch to MBus0	64
0xFFFA1040–0xFFFA107F	OCN Crossbar Switch to MBus1	64
0xFFFA1080–0xFFFA7FFF	Reserved	28 K – 128
0xFFFA8000–0xFFFA8FFF	Dedicated DMA Controller 0	4 K
0xFFFA9000–0xFFFA9FFF	DMA Controller 0 to OCN	4 K
0xFFFAA000–0xFFFAAFFF	Dedicated DMA Controller 1	4 K
0xFFFAB000–0xFFFABFFF	DMA Controller 1 to OCN	4 K
0xFFFAC000–0xFFFACFFF	Reserved	4 K
0xFFFAD000–0xFFFAD7FF	SerDes PHY 1	2 K
0xFFFAD800–0xFFFB6FFF	Reserved	38 K
0xFFFB7000–0xFFFB7FFF	PCI Express	4 K

Table 9-4. CCSR Address Space (Continued)

Address	Purpose	Size (Bytes)
0xFFFB8000–0xFFFB9FFF	Reserved	8 K
0xFFFBA000–0xFFFBAFFF	CLASS1	4K
0xFFFBB000–0xFFFBB7FF	Reserved	2 K
0xFFFBB800–0xFFFBB8FF	Performance Monitor Registers	256
0xFFFBB900–0xFFFCFFFF	Reserved	82 K – 256
0xFFFD0000–0xFFFDFFFF	Security Engine (SEC)	64 K
0xFFFE0000–0xFFFFEFFF	Reserved	124 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 and system peripherals 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)	Comment
0x00000000–0x1FFFFFFF		Reserved	768 M	
The allocation of the shared core subsystem L2 space (core subsystem 0–3) is as follows:				
0x20000000–0x2BFFFFFF		Shared L2 space for Cores 0 and 1		Only L2 spaces are available. Others are reserved.
0x20000000–0x27FFFFFF		Shared L2 space for Cores 2 and 3		
The reserved L2 space (core subsystem 0–5) is as follows:				
0x2C000000–0x2FFFFFFF		Reserved for Core 0		End address is computed as 0x30000000 + (core index × 0x1000000) – 1
0x2C000000–0x30FFFFFF		Reserved for Core 1		
0x28000000–0x31FFFFFF		Reserved for Core 2		
0x28000000–0x32FFFFFF		Reserved for Core 3		
0x20000000–0x33FFFFFF		Reserved for Core 4		
0x20000000–0x34FFFFFF		Reserved for Core 5		
The allocation of the private M2 memory space f is core subsystem dependent as follows:				
Core 0	0x30000000–0x30FFFFFF	Private M2 memory (maximum)	512 K maximum	Not all of the 512 KB is always allocated. Allocation and illegal access detection are done by the core subsystem.
Core 1	0x31000000–0x31FFFFFF	Private M2 memory (maximum)	512 K maximum	
Core 2	0x32000000–0x32FFFFFF	Private M2 memory (maximum)	512 K maximum	
Core 3	0x33000000–0x33FFFFFF	Private M2 memory (maximum)	512 K maximum	
Core 4	0x34000000–0x34FFFFFF	Private M2 memory (maximum)	512 K maximum	
Core 5	0x35000000–0x35FFFFFF	Private M2 memory (maximum)	512 K maximum	
The reserved M2 space (core subsystem 0–5) is as follows:				
0x31000000–0x31FFFFFF		Reserved for Core 0		For cores 0–3, the end address is computed as 0x30000000 + ((core index + 2) × 0x1000000) – 1. For cores 4–5, the end address is fixed.
0x32000000–0x32FFFFFF		Reserved for Core 1		
0x33000000–0x33FFFFFF		Reserved for Core 2		
0x34000000–0x34FFFFFF		Reserved for Core 3		
0x35000000–0x3FFFFFFF		Reserved for Core 4		
0x36000000–0x3FFFFFFF		Reserved for Core 5		

Table 9-5. SC3850 (Data) View of the System Address Space (Continued)

Address		Purpose	Size (Bytes)	Comment
The shared M2 space (core subsystem 0–3 only) is as follows:				
Core 0	0x32000000–0x3FFFFFFF	Shared M2 space		Only M2 spaces are available. Others are reserved.
Core 1	0x32000000–0x3FFFFFFF	Shared M2 space		
Core 2	0x34000000–0x3FFFFFFF	Shared M2 space		
Core 3	0x34000000–0x3FFFFFFF	Shared M2 space		
Other memory space is as follows:				
0x40000000–0xFEFFFFFF		Shared Memory Address Space	3 G – 16 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 MSC8157E peripherals (RapidIO and PCI Express controllers, JTAG, QUICC Engine subsystem, 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 – 16 M
0xFF000000–0xFFF0FFFF	Reserved	15M + 64K
0xFFF10000–0xFFFFEFFF	CCSR Address Space	956 K
0xFFFFF000–0xFFFFFFFF	Reserved	4 K

An external initiator (to the MSC8157E 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)
0x00000000–0x1FFFFFFF	Reserved	512 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 – 16 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	
• 0x40000000–0xBFFFFFFF	DDR memory	
• 0xC0000000–0xD09FFFFF	M3 Memory	
• 0xD0A00000–0xDFFFFFFF	reserved	
• 0xE0000000–0xE03FFFFF	MAPLE-B2 (see **Chapter 26**, *Multi Accelerator Platform Engine, Baseband 2 (MAPLE-B2)*)	
– 0xE0000000–0xE00004FF	Parameter RAM	
– 0xE0000000–0xE0000003	reserved	
– 0xE0000004–0xE0000007	MAPLE BD Rings Configuration Parameter 0	MBDRCP0
– 0xE0000008–0xE000000B	MAPLE BD Rings Configuration Parameter 1	MBDRCP1
– 0xE000000C–0xE000000F	reserved	
– 0xE0000010–0xE0000013	MAPLE UCode Version Number	MUCVP
– 0xE0000014–0xE0000017	reserved	
– 0xE0000018–0xE000001B	MAPLE Timer Period Parameter	MP_TPP
– 0xE000001C–0xE000001F	reserved	
– 0xE0000020–0xE0000023	MAPLE Mode Configurations 0 Parameter. (programmed by API)	MMC0P
– 0xE0000024–0xE0000027	MAPLE Mode Configurations 1 Parameter. (programmed by API)	MMC1P
– 0xE0000028–0xE000002B	MAPLE eTVPE Configuration parameter (programmed by API)	MTVCP
– 0xE000002C–0xE000002F	CRPE-DL Output Mode Configuration Parameter. (programmed by API)	CDOMCP

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE0000030–0xE0000033	CRPE-ULB Mode Configuration Parameter. (programmed by API)	CUBMCP
– 0xE0000034–0xE000003F	reserved	
– 0xE0000040–0xE0000043	MAPLE CRPE Reset Completion Indication Parameter	MCRRCIP
– 0xE0000044–0xE000007F	reserved	
– 0xE0000080–0xE0000083	MAPLE eTVPE BD Ring High Priority A 0 Parameter	MTVBRHPA0P
– 0xE0000084–0xE0000087	MAPLE eTVPE BD Ring High Priority B 0 Parameter	MTVBRHPB0P
– 0xE0000088–0xE000008B	MAPLE eTVPE BD Ring High Priority A 1 Parameter	MTVBRHPA1P
– 0xE000008C–0xE000008F	MAPLE eTVPE BD Ring High Priority B 1 Parameter	MTVBRHPB1P
– 0xE0000090–0xE0000093	MAPLE eTVPE BD Ring High Priority A 2 Parameter	MTVBRHPA2P
– 0xE0000094–0xE0000097	MAPLE eTVPE BD Ring High Priority B 2 Parameter	MTVBRHPB2P
– 0xE0000098–0xE000009B	MAPLE eTVPE BD Ring High Priority A 3 Parameter	MTVBRHPA3P
– 0xE000009C–0xE000009F	MAPLE eTVPE BD Ring High Priority B 3 Parameter	MTVBRHPB3P
– 0xE00000A0–0xE00000A3	MAPLE eTVPE BD Ring High Priority A 4 Parameter	MTVBRHPA4P
– 0xE00000A4–0xE00000A7	MAPLE eTVPE BD Ring High Priority B 4 Parameter	MTVBRHPB4P
– 0xE00000A8–0xE00000AB	MAPLE eTVPE BD Ring High Priority A 5 Parameter	MTVBRHPA5P
– 0xE00000AC–0xE00000AF	MAPLE eTVPE BD Ring High Priority B 5 Parameter	MTVBRHPB5P
– 0xE00000B0–0xE00000B3	MAPLE eTVPE BD Ring High Priority A 6 Parameter	MTVBRHPA6P
– 0xE00000B4–0xE00000B7	MAPLE eTVPE BD Ring High Priority B 6 Parameter	MTVBRHPB6P
– 0xE00000B8–0xE00000BB	MAPLE eTVPE BD Ring High Priority A 7 Parameter	MTVBRHPA7P
– 0xE00000BC–0xE00000BF	MAPLE eTVPE BD Ring High Priority B 7 Parameter	MTVBRHPB7P
– 0xE00000C0–0xE00000C3	MAPLE eTVPE BD Ring Low Priority A 0 Parameter	MTVBRLPA0P
– 0xE00000C4–0xE00000C7	MAPLE eTVPE BD Ring Low Priority B 0 Parameter	MTVBRLPB0P
– 0xE00000C8–0xE00000CB	MAPLE eTVPE BD Ring Low Priority A 1 Parameter	MTVBRLPA1P
– 0xE00000CC–0xE00000CF	MAPLE eTVPE BD Ring Low Priority B 1 Parameter	MTVBRLPB1P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE00000D0–0xE00000D3	MAPLE eTVPE BD Ring Low Priority A 2 Parameter	MTVBRLPA2P
– 0xE00000D4–0xE00000D7	MAPLE eTVPE BD Ring Low Priority B 2 Parameter	MTVBRLPB2P
– 0xE00000D8–0xE00000DB	MAPLE eTVPE BD Ring Low Priority A 3 Parameter	MTVBRLPA3P
– 0xE00000DC–0xE00000DF	MAPLE eTVPE BD Ring Low Priority B 3 Parameter	MTVBRLPB3P
– 0xE00000E0–0xE00000E3	MAPLE eTVPE BD Ring Low Priority A 4 Parameter	MTVBRLPA4P
– 0xE00000E4–0xE00000E7	MAPLE eTVPE BD Ring Low Priority B 4 Parameter	MTVBRLPB4P
– 0xE00000E8–0xE00000EB	MAPLE eTVPE BD Ring Low Priority A 5 Parameter	MTVBRLPA5P
– 0xE00000EC–0xE00000EF	MAPLE eTVPE BD Ring Low Priority B 5 Parameter	MTVBRLPB5P
– 0xE00000F0–0xE00000F3	MAPLE eTVPE BD Ring Low Priority A 6 Parameter	MTVBRLPA6P
– 0xE00000F4–0xE00000F7	MAPLE eTVPE BD Ring Low Priority B 6 Parameter	MTVBRLPB6P
– 0xE00000F8–0xE00000FB	MAPLE eTVPE BD Ring Low Priority A 7 Parameter	MTVBRLPA7P
– 0xE00000FC–0xE00000FF	MAPLE eTVPE BD Ring Low Priority B 7 Parameter	MTVBRLPB7P
– 0xE0000100–0xE0000103	MAPLE eFTPE_0 BD Ring High Priority A 0 Parameter	MF0BRHPA0P
– 0xE0000104–0xE0000107	MAPLE eFTPE_0 BD Ring High Priority B 0 Parameter	MF0BRHPB0P
– 0xE0000108–0xE000010B	MAPLE eFTPE_0 BD Ring High Priority A 1 Parameter	MF0BRHPA1P
– 0xE000010C–0xE000010F	MAPLE eFTPE_0 BD Ring High Priority B 1 Parameter	MF0BRHPB1P
– 0xE0000110–0xE0000113	MAPLE eFTPE_0 BD Ring High Priority A 2 Parameter	MF0BRHPA2P
– 0xE0000114–0xE0000117	MAPLE eFTPE_0 BD Ring High Priority B 2 Parameter	MF0BRHPB2P
– 0xE0000118–0xE000011B	MAPLE eFTPE_0 BD Ring High Priority A 3 Parameter	MF0BRHPA3P
– 0xE000011C–0xE000011F	MAPLE eFTPE_0 BD Ring High Priority B 3 Parameter	MF0BRHPB3P
– 0xE0000120–0xE0000123	MAPLE eFTPE_0 BD Ring High Priority A 4 Parameter	MF0BRHPA4P
– 0xE0000124–0xE0000127	MAPLE eFTPE_0 BD Ring High Priority B 4 Parameter	MF0BRHPB4P
– 0xE0000128–0xE000012B	MAPLE eFTPE_0 BD Ring High Priority A 5 Parameter	MF0BRHPA5P
– 0xE000012C–0xE000012F	MAPLE eFTPE_0 BD Ring High Priority B 5 Parameter	MF0BRHPB5P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE0000130–0xE0000133	MAPLE eFTPE_0 BD Ring High Priority A 6 Parameter	MF0BRHPA6P
– 0xE0000134–0xE0000137	MAPLE eFTPE_0 BD Ring High Priority B 6 Parameter	MF0BRHPB6P
– 0xE0000138–0xE000013B	MAPLE eFTPE_0 BD Ring High Priority A 7 Parameter	MF0BRHPA7P
– 0xE000013C–0xE000013F	MAPLE eFTPE_0 BD Ring High Priority B 7 Parameter	MF0BRHPB7P
– 0xE0000140–0xE0000143	MAPLE eFTPE_0 BD Ring Low Priority A 0 Parameter	MF0BRLPA0P
– 0xE0000144–0xE0000147	MAPLE eFTPE_0 BD Ring Low Priority B 0 Parameter	MF0BRLPB0P
– 0xE0000148–0xE000014B	MAPLE eFTPE_0 BD Ring Low Priority A 1 Parameter	MF0BRLPA1P
– 0xE000014C–0xE000014F	MAPLE eFTPE_0 BD Ring Low Priority B 1 Parameter	MF0BRLPB1P
– 0xE0000150–0xE0000153	MAPLE eFTPE_0 BD Ring Low Priority A 2 Parameter	MF0BRLPA2P
– 0xE0000154–0xE0000157	MAPLE eFTPE_0 BD Ring Low Priority B 2 Parameter	MF0BRLPB2P
– 0xE0000158–0xE000015B	MAPLE eFTPE_0 BD Ring Low Priority A 3 Parameter	MF0BRLPA3P
– 0xE000015C–0xE000015F	MAPLE eFTPE_0 BD Ring Low Priority B 3 Parameter	MF0BRLPB3P
– 0xE0000160–0xE0000163	MAPLE eFTPE_0 BD Ring Low Priority A 4 Parameter	MF0BRLPA4P
– 0xE0000164–0xE0000167	MAPLE eFTPE_0 BD Ring Low Priority B 4 Parameter	MF0BRLPB4P
– 0xE0000168–0xE000016B	MAPLE eFTPE_0 BD Ring Low Priority A 5 Parameter	MF0BRLPA5P
– 0xE000016C–0xE000016F	MAPLE eFTPE_0 BD Ring Low Priority B 5 Parameter	MF0BRLPB5P
– 0xE0000170–0xE0000173	MAPLE eFTPE_0 BD Ring Low Priority A 6 Parameter	MF0BRLPA6P
– 0xE0000174–0xE0000177	MAPLE eFTPE_0 BD Ring Low Priority B 6 Parameter	MF0BRLPB6P
– 0xE0000178–0xE000017B	MAPLE eFTPE_0 BD Ring Low Priority A 7 Parameter	MF0BRLPA7P
– 0xE000017C–0xE000017F	MAPLE eFTPE_0 BD Ring Low Priority B 7 Parameter	MF0BRLPB7P
– 0xE0000180–0xE0000183	MAPLE eFTPE_1 BD Ring High Priority A 0 Parameter	MF1BRHPA0P
– 0xE0000184–0xE0000187	MAPLE eFTPE_1 BD Ring High Priority B 0 Parameter	MF1BRHPB0P
– 0xE0000188–0xE000018B	MAPLE eFTPE_1 BD Ring High Priority A 1 Parameter	MF1BRHPA1P
– 0xE000018C–0xE000018F	MAPLE eFTPE_1 BD Ring High Priority B 1 Parameter	MF1BRHPB1P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE0000190–0xE0000183	MAPLE eFTPE_1 BD Ring High Priority A 2 Parameter	MF1BRHPA2P
– 0xE0000194–0xE0000197	MAPLE eFTPE_1 BD Ring High Priority B 2 Parameter	MF1BRHPB2P
– 0xE0000198–0xE000019B	MAPLE eFTPE_1 BD Ring High Priority A 3 Parameter	MF1BRHPA3P
– 0xE000019C–0xE000019F	MAPLE eFTPE_1 BD Ring High Priority B 3 Parameter	MF1BRHPB3P
– 0xE00001A0–0xE00001A3	MAPLE eFTPE_1 BD Ring High Priority A 4 Parameter	MF1BRHPA4P
– 0xE00001A4–0xE00001A7	MAPLE eFTPE_1 BD Ring High Priority B 4 Parameter	MF1BRHPB4P
– 0xE00001A8–0xE00001AB	MAPLE eFTPE_1 BD Ring High Priority A 5 Parameter	MF1BRHPA5P
– 0xE00001AC–0xE00001AF	MAPLE eFTPE_1 BD Ring High Priority B 5 Parameter	MF1BRHPB5P
– 0xE00001B0–0xE00001B3	MAPLE eFTPE_1 BD Ring High Priority A 6 Parameter	MF1BRHPA6P
– 0xE00001B4–0xE00001B7	MAPLE eFTPE_1 BD Ring High Priority B 6 Parameter	MF1BRHPB6P
– 0xE00001B8–0xE00001BB	MAPLE eFTPE_1 BD Ring High Priority A 7 Parameter	MF1BRHPA7P
– 0xE00001BC–0xE00001BF	MAPLE eFTPE_1 BD Ring High Priority B 7 Parameter	MF1BRHPB7P
– 0xE00001C0–0xE00001C3	MAPLE eFTPE_1 BD Ring Low Priority A 0 Parameter	MF1BRLPA0P
– 0xE00001C4–0xE00001C7	MAPLE eFTPE_1 BD Ring Low Priority B 0 Parameter	MF1BRLPB0P
– 0xE00001C8–0xE00001CB	MAPLE eFTPE_1 BD Ring Low Priority A 1 Parameter	MF1BRLPA1P
– 0xE00001CC–0xE00001CF	MAPLE eFTPE_1 BD Ring Low Priority B 1 Parameter	MF1BRLPB1P
– 0xE00001D0–0xE00001D3	MAPLE eFTPE_1 BD Ring Low Priority A 2 Parameter	MF1BRLPA2P
– 0xE00001D4–0xE00001D7	MAPLE eFTPE_1 BD Ring Low Priority B 2 Parameter	MF1BRLPB2P
– 0xE00001D8–0xE00001DB	MAPLE eFTPE_1 BD Ring Low Priority A 3 Parameter	MF1BRLPA3P
– 0xE00001DC–0xE00001DF	MAPLE eFTPE_1 BD Ring Low Priority B 3 Parameter	MF1BRLPB3P
– 0xE00001E0–0xE00001E3	MAPLE eFTPE_1 BD Ring Low Priority A 4 Parameter	MF1BRLPA4P
– 0xE00001E4–0xE00001E7	MAPLE eFTPE_1 BD Ring Low Priority B 4 Parameter	MF1BRLPB4P
– 0xE00001E8–0xE00001EB	MAPLE eFTPE_1 BD Ring Low Priority A 5 Parameter	MF1BRLPA5P
– 0xE00001EC–0xE00001EF	MAPLE eFTPE_1 BD Ring Low Priority B 5 Parameter	MF1BRLPB5P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE00001F0–0xE00001F3	MAPLE eFTPE_1 BD Ring Low Priority A 6 Parameter	MF1BRLPA6P
– 0xE00001F4–0xE00001F7	MAPLE eFTPE_1 BD Ring Low Priority B 6 Parameter	MF1BRLPB6P
– 0xE00001F8–0xE00001FB	MAPLE eFTPE_1 BD Ring Low Priority A 7 Parameter	MF1BRLPA7P
– 0xE00001FC–0xE00001FF	MAPLE eFTPE_1 BD Ring Low Priority B 7 Parameter	MF1BRLPB7P
– 0xE0000200–0xE0000203	MAPLE eFTPE_2 BD Ring High Priority A 0 Parameter	MF2BRHPA0P
– 0xE0000204–0xE0000207	MAPLE eFTPE_2 BD Ring High Priority B 0 Parameter	MF2BRHPB0P
– 0xE0000208–0xE000020B	MAPLE eFTPE_2 BD Ring High Priority A 1 Parameter	MF2BRHPA1P
– 0xE000020C–0xE000020F	MAPLE eFTPE_2 BD Ring High Priority B 1 Parameter	MF2BRHPB1P
– 0xE0000210–0xE0000213	MAPLE eFTPE_2 BD Ring High Priority A 2 Parameter	MF2BRHPA2P
– 0xE0000214–0xE0000217	MAPLE eFTPE_2 BD Ring High Priority B 2 Parameter	MF2BRHPB2P
– 0xE0000218–0xE000021B	MAPLE eFTPE_2 BD Ring High Priority A 3 Parameter	MF2BRHPA3P
– 0xE000021C–0xE000021F	MAPLE eFTPE_2 BD Ring High Priority B 3 Parameter	MF2BRHPB3P
– 0xE0000220–0xE0000223	MAPLE eFTPE_2 BD Ring High Priority A 4 Parameter	MF2BRHPA4P
– 0xE0000224–0xE0000227	MAPLE eFTPE_2 BD Ring High Priority B 4 Parameter	MF2BRHPB4P
– 0xE0000228–0xE000022B	MAPLE eFTPE_2 BD Ring High Priority A 5 Parameter	MF2BRHPA5P
– 0xE000022C–0xE000022F	MAPLE eFTPE_2 BD Ring High Priority B 5 Parameter	MF2BRHPB5P
– 0xE0000230–0xE0000233	MAPLE eFTPE_2 BD Ring High Priority A 6 Parameter	MF2BRHPA6P
– 0xE0000234–0xE0000237	MAPLE eFTPE_2 BD Ring High Priority B 6 Parameter	MF2BRHPB6P
– 0xE0000238–0xE000023B	MAPLE eFTPE_2 BD Ring High Priority A 7 Parameter	MF2BRHPA7P
– 0xE000023C–0xE000023F	MAPLE eFTPE_2 BD Ring High Priority B 7 Parameter	MF2BRHPB7P
– 0xE0000240–0xE0000243	MAPLE eFTPE_2 BD Ring Low Priority A 0 Parameter	MF2BRLPA0P
– 0xE0000244–0xE0000247	MAPLE eFTPE_2 BD Ring Low Priority B 0 Parameter	MF2BRLPB0P
– 0xE0000248–0xE000024B	MAPLE eFTPE_2 BD Ring Low Priority A 1 Parameter	MF2BRLPA1P
– 0xE000024C–0xE000024F	MAPLE eFTPE_2 BD Ring Low Priority B 1 Parameter	MF2BRLPB1P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE0000250–0xE0000253	MAPLE eFTPE_2 BD Ring Low Priority A 2 Parameter	MF2BRLPA2P
– 0xE0000254–0xE0000257	MAPLE eFTPE_2 BD Ring Low Priority B 2 Parameter	MF2BRLPB2P
– 0xE0000258–0xE000025B	MAPLE eFTPE_2 BD Ring Low Priority A 3 Parameter	MF2BRLPA3P
– 0xE000025C–0xE000025F	MAPLE eFTPE_2 BD Ring Low Priority B 3 Parameter	MF2BRLPB3P
– 0xE0000260–0xE0000263	MAPLE eFTPE_2 BD Ring Low Priority A 4 Parameter	MF2BRLPA4P
– 0xE0000264–0xE0000267	MAPLE eFTPE_2 BD Ring Low Priority B 4 Parameter	MF2BRLPB4P
– 0xE0000268–0xE000026B	MAPLE eFTPE_2 BD Ring Low Priority A 5 Parameter	MF2BRLPA5P
– 0xE000026C–0xE000026F	MAPLE eFTPE_2 BD Ring Low Priority B 5 Parameter	MF2BRLPB5P
– 0xE0000270–0xE0000273	MAPLE eFTPE_2 BD Ring Low Priority A 6 Parameter	MF2BRLPA6P
– 0xE0000274–0xE0000277	MAPLE eFTPE_2 BD Ring Low Priority B 6 Parameter	MF2BRLPB6P
– 0xE0000278–0xE000027B	MAPLE eFTPE_2 BD Ring Low Priority A 7 Parameter	MF2BRLPA7P
– 0xE000027C–0xE000027F	MAPLE eFTPE_2 BD Ring Low Priority B 7 Parameter	MF2BRLPB7P
– 0xE0000280–0xE0000283	MAPLE DEPE BD Ring High Priority A 0 Parameter	MDEBRHPA0P
– 0xE0000284–0xE0000287	MAPLE DEPE BD Ring High Priority B 0 Parameter	MDEBRHPB0P
– 0xE0000288–0xE000028B	MAPLE DEPE BD Ring High Priority A 1 Parameter	MDEBRHPA1P
– 0xE000028C–0xE000028F	MAPLE DEPE BD Ring High Priority B 1 Parameter	MDEBRHPB1P
– 0xE0000290–0xE0000293	MAPLE DEPE BD Ring High Priority A 2 Parameter	MDEBRHPA2P
– 0xE0000294–0xE0000297	MAPLE DEPE BD Ring High Priority B 2 Parameter	MDEBRHPB2P
– 0xE0000298–0xE000029B	MAPLE DEPE BD Ring High Priority A 3 Parameter	MDEBRHPA3P
– 0xE000029C–0xE000029F	MAPLE DEPE BD Ring High Priority B 3 Parameter	MDEBRHPB3P
– 0xE00002A0–0xE00002A3	MAPLE DEPE BD Ring High Priority A 4 Parameter	MDEBRHPA4P
– 0xE00002A4–0xE00002A7	MAPLE DEPE BD Ring High Priority B 4 Parameter	MDEBRHPB4P
– 0xE00002A8–0xE00002AB	MAPLE DEPE BD Ring High Priority A 5 Parameter	MDEBRHPA5P
– 0xE00002AC–0xE00002AF	MAPLE DEPE BD Ring High Priority B 5 Parameter	MDEBRHPB5P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE00002B0–0xE00002B3	MAPLE DEPE BD Ring High Priority A 6 Parameter	MDEBRHPA6P
– 0xE00002B4–0xE00002B7	MAPLE DEPE BD Ring High Priority B 6 Parameter	MDEBRHPB6P
– 0xE00002B8–0xE00002BB	MAPLE DEPE BD Ring High Priority A 7 Parameter	MDEBRHPA7P
– 0xE00002BC–0xE00002BF	MAPLE DEPE BD Ring High Priority B 7 Parameter	MDEBRHPB7P
– 0xE00002C0–0xE00002C3	MAPLE DEPE BD Ring Low Priority A 0 Parameter	MDEBRLPA0P
– 0xE00002C4–0xE00002C7	MAPLE DEPE BD Ring Low Priority B 0 Parameter	MDEBRLPB0P
– 0xE00002C8–0xE00002CB	MAPLE DEPE BD Ring Low Priority A 1 Parameter	MDEBRLPA1P
– 0xE00002CC–0xE00002CF	MAPLE DEPE BD Ring Low Priority B 1 Parameter	MDEBRLPB1P
– 0xE00002D0–0xE00002D3	MAPLE DEPE BD Ring Low Priority A 2 Parameter	MDEBRLPA2P
– 0xE00002D4–0xE00002D7	MAPLE DEPE BD Ring Low Priority B 2 Parameter	MDEBRLPB2P
– 0xE00002D8–0xE00002DB	MAPLE DEPE BD Ring Low Priority A 3 Parameter	MDEBRLPA3P
– 0xE00002DC–0xE00002DF	MAPLE DEPE BD Ring Low Priority B 3 Parameter	MDEBRLPB3P
– 0xE00002E0–0xE00002E3	MAPLE DEPE BD Ring Low Priority A 4 Parameter	MDEBRLPA4P
– 0xE00002E4–0xE00002E7	MAPLE DEPE BD Ring Low Priority B 4 Parameter	MDEBRLPB4P
– 0xE00002E8–0xE00002EB	MAPLE DEPE BD Ring Low Priority A 5 Parameter	MDEBRLPA5P
– 0xE00002EC–0xE00002EF	MAPLE DEPE BD Ring Low Priority B 5 Parameter	MDEBRLPB5P
– 0xE00002F0–0xE00002F3	MAPLE DEPE BD Ring Low Priority A 6 Parameter	MDEBRLPA6P
– 0xE00002F4–0xE00002F7	MAPLE DEPE BD Ring Low Priority B 6 Parameter	MDEBRLPB6P
– 0xE00002F8–0xE00002FB	MAPLE DEPE BD Ring Low Priority A 7 Parameter	MDEBRLPA7P
– 0xE00002FC–0xE00002FF	MAPLE DEPE BD Ring Low Priority B 7 Parameter	MDEBRLPB7P
– 0xE0000300–0xE0000303	MAPLE CRCPE BD Ring High Priority A 0 Parameter	MCRCBRHPA0P
– 0xE0000304–0xE0000307	MAPLE CRCPE BD Ring High Priority B 0 Parameter	MCRCBRHPB0P
– 0xE0000308–0xE000030B	MAPLE CRCPE BD Ring High Priority A 1 Parameter	MCRCBRHPA1P
– 0xE000030C–0xE000030F	MAPLE CRCPE BD Ring High Priority B 1 Parameter	MCRCBRHPB1P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE0000310–0xE0000313	MAPLE CRCPE BD Ring High Priority A 2 Parameter	MCRCBRHPA2P
– 0xE0000314–0xE0000317	MAPLE CRCPE BD Ring High Priority B 2 Parameter	MCRCBRHPB2P
– 0xE0000318–0xE000031B	MAPLE CRCPE BD Ring High Priority A 3 Parameter	MCRCBRHPA3P
– 0xE000031C–0xE000031F	MAPLE CRCPE BD Ring High Priority B 3 Parameter	MCRCBRHPB3P
– 0xE0000320–0xE0000323	MAPLE CRCPE BD Ring High Priority A 4 Parameter	MCRCBRHPA4P
– 0xE0000324–0xE0000327	MAPLE CRCPE BD Ring High Priority B 4 Parameter	MCRCBRHPB4P
– 0xE0000328–0xE000032B	MAPLE CRCPE BD Ring High Priority A 5 Parameter	MCRCBRHPA5P
– 0xE000032C–0xE000032F	MAPLE CRCPE BD Ring High Priority B 5 Parameter	MCRCBRHPB5P
– 0xE0000330–0xE0000333	MAPLE CRCPE BD Ring High Priority A 6 Parameter	MCRCBRHPA6P
– 0xE0000334–0xE0000337	MAPLE CRCPE BD Ring High Priority B 6 Parameter	MCRCBRHPB6P
– 0xE0000338–0xE000033B	MAPLE CRCPE BD Ring High Priority A 7 Parameter	MCRCBRHPA7P
– 0xE000033C–0xE000033F	MAPLE CRCPE BD Ring High Priority B 7 Parameter	MCRCBRHPB7P
– 0xE0000340–0xE0000343	MAPLE CRCPE BD Ring Low Priority A 0 Parameter	MCRCBRLPA0P
– 0xE0000344–0xE0000347	MAPLE CRCPE BD Ring Low Priority B 0 Parameter	MCRCBRLPB0P
– 0xE0000348–0xE000034B	MAPLE CRCPE BD Ring Low Priority A 1 Parameter	MCRCBRLPA1P
– 0xE000034C–0xE000034F	MAPLE CRCPE BD Ring Low Priority B 1 Parameter	MCRCBRLPB1P
– 0xE0000350–0xE0000353	MAPLE CRCPE BD Ring Low Priority A 2 Parameter	MCRCBRLPA2P
– 0xE0000354–0xE0000357	MAPLE CRCPE BD Ring Low Priority B 2 Parameter	MCRCBRLPB2P
– 0xE0000358–0xE000035B	MAPLE CRCPE BD Ring Low Priority A 3 Parameter	MCRCBRLPA3P
– 0xE000035C–0xE000035F	MAPLE CRCPE BD Ring Low Priority B 3 Parameter	MCRCBRLPB3P
– 0xE0000360–0xE0000363	MAPLE CRCPE BD Ring Low Priority A 4 Parameter	MCRCBRLPA4P
– 0xE0000364–0xE0000367	MAPLE CRCPE BD Ring Low Priority B 4 Parameter	MCRCBRLPB4P
– 0xE0000368–0xE000036B	MAPLE CRCPE BD Ring Low Priority A 5 Parameter	MCRCBRLPA5P
– 0xE000036C–0xE000036F	MAPLE CRCPE BD Ring Low Priority B 5 Parameter	MCRCBRLPB5P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE0000370–0xE0000373	MAPLE CRCPE BD Ring Low Priority A 6 Parameter	MCRCBRLPA6P
– 0xE0000374–0xE0000377	MAPLE CRCPE BD Ring Low Priority B 6 Parameter	MCRCBRLPB6P
– 0xE0000378–0xE000037B	MAPLE CRCPE BD Ring Low Priority A 7 Parameter	MCRCBRLPA7P
– 0xE000037C–0xE000037F	MAPLE CRCPE BD Ring Low Priority B 7 Parameter	MCRCBRLPB7P
– 0xE0000380–0xE0000383	MAPLE EQPE BD Ring High Priority A 0 Parameter	MEQBRHPA0P
– 0xE0000384–0xE0000387	MAPLE EQPE BD Ring High Priority B 0 Parameter	MEQBRHPB0P
– 0xE0000388–0xE000038B	MAPLE EQPE BD Ring High Priority A 1 Parameter	MEQBRHPA1P
– 0xE000038C–0xE000038F	MAPLE EQPE BD Ring High Priority B 1 Parameter	MEQBRHPB1P
– 0xE0000390–0xE0000393	MAPLE EQPE BD Ring High Priority A 2 Parameter	MEQBRHPA2P
– 0xE0000394–0xE0000397	MAPLE EQPE BD Ring High Priority B 2 Parameter	MEQBRHPB2P
– 0xE0000398–0xE000039B	MAPLE EQPE BD Ring High Priority A 3 Parameter	MEQBRHPA3P
– 0xE000039C–0xE000039F	MAPLE EQPE BD Ring High Priority B 3 Parameter	MEQBRHPB3P
– 0xE00003A0–0xE00003A3	MAPLE EQPE BD Ring High Priority A 4 Parameter	MEQBRHPA4P
– 0xE00003A4–0xE00003A7	MAPLE EQPE BD Ring High Priority B 4 Parameter	MEQBRHPB4P
– 0xE00003A8–0xE00003AB	MAPLE EQPE BD Ring High Priority A 5 Parameter	MEQBRHPA5P
– 0xE00003AC–0xE00003AF	MAPLE EQPE BD Ring High Priority B 5 Parameter	MEQBRHPB5P
– 0xE00003B0–0xE00003B3	MAPLE EQPE BD Ring High Priority A 6 Parameter	MEQBRHPA6P
– 0xE00003B4–0xE00003B7	MAPLE EQPE BD Ring High Priority B 6 Parameter	MEQBRHPB6P
– 0xE00003B8–0xE00003BB	MAPLE EQPE BD Ring High Priority A 7 Parameter	MEQBRHPA7P
– 0xE00003BC–0xE00003BF	MAPLE EQPE BD Ring High Priority B 7 Parameter	MEQBRHPB7P
– 0xE00003C0–0xE00003C3	MAPLE EQPE BD Ring Low Priority A 0 Parameter	MEQBRLPA0P
– 0xE00003C4–0xE00003C7	MAPLE EQPE BD Ring Low Priority B 0 Parameter	MEQBRLPB0P
– 0xE00003C8–0xE00003CB	MAPLE EQPE BD Ring Low Priority A 1 Parameter	MEQBRLPA1P
– 0xE00003CC–0xE00003CF	MAPLE EQPE BD Ring Low Priority B 1 Parameter	MEQBRLPB1P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE00003D0–0xE00003D3	MAPLE EQPE BD Ring Low Priority A 2 Parameter	MEQBRLPA2P
– 0xE00003D4–0xE00003D7	MAPLE EQPE BD Ring Low Priority B 2 Parameter	MEQBRLPB2P
– 0xE00003D8–0xE00003DB	MAPLE EQPE BD Ring Low Priority A 3 Parameter	MEQBRLPA3P
– 0xE00003DC–0xE00003DF	MAPLE EQPE BD Ring Low Priority B 3 Parameter	MEQBRLPB3P
– 0xE00003E0–0xE00003E3	MAPLE EQPE BD Ring Low Priority A 4 Parameter	MEQBRLPA4P
– 0xE00003E4–0xE00003E7	MAPLE EQPE BD Ring Low Priority B 4 Parameter	MEQBRLPB4P
– 0xE00003E8–0xE00003EB	MAPLE EQPE BD Ring Low Priority A 5 Parameter	MEQBRLPA5P
– 0xE00003EC–0xE00003EF	MAPLE EQPE BD Ring Low Priority B 5 Parameter	MEQBRLPB5P
– 0xE00003F0–0xE00003F3	MAPLE EQPE BD Ring Low Priority A 6 Parameter	MEQBRLPA6P
– 0xE00003F4–0xE00003F7	MAPLE EQPE BD Ring Low Priority B 6 Parameter	MEQBRLPB6P
– 0xE00003F8–0xE00003FB	MAPLE EQPE BD Ring Low Priority A 7 Parameter	MEQBRLPA7P
– 0xE00003FC–0xE00003FF	MAPLE EQPE BD Ring Low Priority B 7 Parameter	MEQBRLPB7P
– 0xE0000400–0xE0000403	MAPLE CONVPE BD Ring High Priority A 0 Parameter	MCONVBRHPA0P
– 0xE0000404–0xE0000407	MAPLE CONVPE BD Ring High Priority B 0 Parameter	MCONVBRHPB0P
– 0xE0000408–0xE000040B	MAPLE CONVPE BD Ring High Priority A 1 Parameter	MCONVBRHPA1P
– 0xE000040C–0xE000040F	MAPLE CONVPE BD Ring High Priority B 1 Parameter	MCONVBRHPB1P
– 0xE0000410–0xE0000413	MAPLE CONVPE BD Ring High Priority A 2 Parameter	MCONVBRHPA2P
– 0xE0000414–0xE0000417	MAPLE CONVPE BD Ring High Priority B 2 Parameter	MCONVBRHPB2P
– 0xE0000418–0xE000041B	MAPLE CONVPE BD Ring High Priority A 3 Parameter	MCONVBRHPA3P
– 0xE000041C–0xE000041F	MAPLE CONVPE BD Ring High Priority B 3 Parameter	MCONVBRHPB3P
– 0xE0000420–0xE0000423	MAPLE CONVPE BD Ring High Priority A 4 Parameter	MCONVBRHPA4P
– 0xE0000424–0xE0000417	MAPLE CONVPE BD Ring High Priority B 4 Parameter	MCONVBRHPB4P
– 0xE0000428–0xE000042B	MAPLE CONVPE BD Ring High Priority A 5 Parameter	MCONVBRHPA5P
– 0xE000042C–0xE000042F	MAPLE CONVPE BD Ring High Priority B 5 Parameter	MCONVBRHPB5P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE0000430–0xE0000433	MAPLE CONVPE BD Ring High Priority A 6 Parameter	MCONVBRHPA6P
– 0xE0000434–0xE0000437	MAPLE CONVPE BD Ring High Priority B 6 Parameter	MCONVBRHPB6P
– 0xE0000438–0xE000043B	MAPLE CONVPE BD Ring High Priority A 7 Parameter	MCONVBRHPA7P
– 0xE000043C–0xE000043F	MAPLE CONVPE BD Ring High Priority B 7 Parameter	MCONVBRHPB7P
– 0xE0000440–0xE0000443	MAPLE CONVPE BD Ring Low Priority A 0 Parameter	MCONVBRLPA0P
– 0xE0000444–0xE0000447	MAPLE CONVPE BD Ring Low Priority B 0 Parameter	MCONVBRLPB0P
– 0xE0000448–0xE000044B	MAPLE CONVPE BD Ring Low Priority A 1 Parameter	MCONVBRLPA1P
– 0xE000044C–0xE000044F	MAPLE CONVPE BD Ring Low Priority B 1 Parameter	MCONVBRLPB1P
– 0xE0000450–0xE0000453	MAPLE CONVPE BD Ring Low Priority A 2 Parameter	MCONVBRLPA2P
– 0xE0000454–0xE0000457	MAPLE CONVPE BD Ring Low Priority B 2 Parameter	MCONVBRLPB2P
– 0xE0000458–0xE000045B	MAPLE CONVPE BD Ring Low Priority A 3 Parameter	MCONVBRLPA3P
– 0xE000045C–0xE000045F	MAPLE CONVPE BD Ring Low Priority B 3 Parameter	MCONVBRLPB3P
– 0xE0000460–0xE0000463	MAPLE CONVPE BD Ring Low Priority A 4 Parameter	MCONVBRLPA4P
– 0xE0000464–0xE0000467	MAPLE CONVPE BD Ring Low Priority B 4 Parameter	MCONVBRLPB4P
– 0xE0000468–0xE000046B	MAPLE CONVPE BD Ring Low Priority A 5 Parameter	MCONVBRLPA5P
– 0xE000046C–0xE000046F	MAPLE CONVPE BD Ring Low Priority B 5 Parameter	MCONVBRLPB5P
– 0xE0000470–0xE0000473	MAPLE CONVPE BD Ring Low Priority A 6 Parameter	MCONVBRLPA6P
– 0xE0000474–0xE0000477	MAPLE CONVPE BD Ring Low Priority B 6 Parameter	MCONVBRLPB6P
– 0xE0000478–0xE000047B	MAPLE CONVPE BD Ring Low Priority A 7 Parameter	MCONVBRLPA7P
– 0xE000047C–0xE000047F	MAPLE CONVPE BD Ring Low Priority B 7 Parameter	MCONVBRLPB7P
– 0xE0000480–0xE00004FF	reserved	
– 0xE0000500–0xE00005FF	eTVPE parameters	
– 0x00000500–0x00000503	MAPLE Turbo Viterbi Puncturing Vector 0 High Configuration Parameter	MTVPV0HCP
– 0x00000504–0x00000507	MAPLE Turbo Viterbi Puncturing Vector 0 Low Configuration Parameter	MTVPV0LCP

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0x00000508–0x0000050B	MAPLE Turbo Viterbi Puncturing Vector 1 High Configuration Parameter	MTVPV1HCP
– 0x0000050C–0x0000050F	MAPLE Turbo Viterbi Puncturing Vector 1 Low Configuration Parameter	MTVPV1LCP
– 0x00000510–0x00000513	MAPLE Turbo Viterbi Puncturing Vector 2 High Configuration Parameter	MTVPV2HCP
– 0x00000514–0x00000517	MAPLE Turbo Viterbi Puncturing Vector 2 Low Configuration Parameter	MTVPV2LCP
– 0x00000518–0x0000051B	MAPLE Turbo Viterbi Puncturing Vector 3 High Configuration Parameter	MTVPV3HCP
– 0x0000051C–0x0000051F	MAPLE Turbo Viterbi Puncturing Vector 3 Low Configuration Parameter	MTVPV3LCP
– 0x00000520–0x00000523	MAPLE Turbo Viterbi Puncturing Vector 4 High Configuration Parameter	MTVPV4HCP
– 0x00000524–0x00000527	MAPLE Turbo Viterbi Puncturing Vector 4 Low Configuration Parameter	MTVPV4LCP
– 0x00000528–0x0000052B	MAPLE Turbo Viterbi Puncturing Vector 5 High Configuration Parameter	MTVPV5HCP
– 0x0000052C–0x0000052F	MAPLE Turbo Viterbi Puncturing Vector 5 Low Configuration Parameter	MTVPV5LCP
– 0x00000530–0x00000533	MAPLE Turbo Viterbi Puncturing Vector 6 High Configuration Parameter	MTVPV6HCP
– 0x00000534–0x00000537	MAPLE Turbo Viterbi Puncturing Vector 6 Low Configuration Parameter	MTVPV6LCP
– 0x00000538–0x0000053B	MAPLE Turbo Viterbi Puncturing Vector 7 High Configuration Parameter	MTVPV7HCP
– 0x0000053C–0x0000053F	MAPLE Turbo Viterbi Puncturing Vector 7 Low Configuration Parameter	MTVPV7LCP
– 0x00000540–0x00000543	MAPLE Turbo Viterbi Puncturing Vector 8 High Configuration Parameter	MTVPV8HCP
– 0x00000544–0x00000547	MAPLE Turbo Viterbi Puncturing Vector 8 Low Configuration Parameter	MTVPV8LCP
– 0x00000548–0x0000054B	MAPLE Turbo Viterbi Puncturing Vector 9 High Configuration Parameter	MTVPV9HCP
– 0x0000054C–0x0000054F	MAPLE Turbo Viterbi Puncturing Vector 9 Low Configuration Parameter	MTVPV9LCP
– 0x00000550–0x00000553	MAPLE Turbo Viterbi Puncturing Period Configuration 0 Parameter	MTVPPC0P
– 0x00000554–0x00000557	MAPLE Turbo Viterbi Puncturing Period Configuration 1 Parameter	MTVPPC1P
– 0x00000558–0x0000055B	MAPLE Turbo Viterbi Puncturing Period Configuration 2 Parameter	MTVPPC2P
– 0xE000055C–0xE000055F	reserved	
– 0xE0000560–0xE0000563	MAPLE Turbo Viterbi Polynomial Vector Set 1 Configuration 0 parameter	MTVPVS1C0P
– 0xE0000564–0xE0000567	MAPLE Turbo Viterbi Polynomial Vector Set 1 Configuration 1 parameter	MTVPVS1C1P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE0000568–0xE000056B	MAPLE Turbo Viterbi Polynomial Vector Set 2 Configuration 0 parameter	MTVPVS2C0P
– 0xE000056C–0xE000056F	MAPLE Turbo Viterbi Polynomial Vector Set 2 Configuration 1 parameter	MTVPVS2C1P
– 0xE0000570–0xE0000573	MAPLE Turbo Viterbi Polynomial Vector Set 3 Configuration 0 parameter	MTVPVS3C0P
– 0xE0000574–0xE0000577	MAPLE Turbo Viterbi Polynomial Vector Set 3 Configuration 1 parameter	MTVPVS3C1P
– 0xE0000578–0xE00005FF	reserved	
– 0xE0000600–0xE000067F	eFTPE parameters	
– 0xE0000600–0xE0000603	eFTPE Data Size Set 1 Parameter 0	FTPEDSS1P0
– 0xE0000604–0xE0000607	eFTPE Data Size Set 1 Parameter 1	FTPEDSS1P1
– 0xE0000608–0xE000060B	eFTPE Data Size Set 1 Parameter 2	FTPEDSS1P2
– 0xE000060C–0xE000060F	eFTPE Data Size Set 2 Parameter 0	FTPEDSS2P0
– 0xE0000610–0xE0000613	eFTPE Data Size Set 2 Parameter 1	FTPEDSS2P1
– 0xE0000614–0xE0000617	eFTPE Data Size Set 2 Parameter 2	FTPEDSS2P2
– 0xE0000618–0xE000061B	eFTPE Data Size Set 3 Parameter 0	FTPEDSS3P0
– 0xE000061C–0xE000061F	eFTPE Data Size Set 3 Parameter 1	FTPEDSS3P1
– 0xE0000620–0xE0000623	eFTPE Data Size Set 3 Parameter 2	FTPEDSS3P2
– 0xE0000624–0xE0000627	eFTPE Data Size Set 4 Parameter 0	FTPEDSS4P0
– 0xE0000628–0xE000062B	eFTPE Data Size Set 4 Parameter 1	FTPEDSS4P1
– 0xE000062C–0xE000062F	eFTPE Data Size Set 4 Parameter 2	FTPEDSS4P2
– 0xE0000630–0xE0000633	eFTPE Data Size Set 5 Parameter 0	FTPEDSS5P0
– 0xE0000634–0xE0000637	eFTPE Data Size Set 5 Parameter 1	FTPEDSS5P1
– 0xE0000638–0xE000063B	eFTPE Data Size Set 5 Parameter 2	FTPEDSS5P2
– 0xE000063C–0xE000063F	eFTPE Data Size Set 6 Parameter 0	FTPEDSS6P0
– 0xE0000640–0xE0000643	eFTPE Data Size Set 6 Parameter 1	FTPEDSS6P1
– 0xE0000644–0xE0000647	eFTPE Data Size Set 6 Parameter 2	FTPEDSS6P2

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE0000648–0xE000064B	eFTPE Data Size Set 7 Parameter 0	FTPEDSS7P0
– 0xE000064C–0xE000064F	eFTPE Data Size Set 7 Parameter 1	FTPEDSS7P1
– 0xE0000650–0xE0000653	eFTPE Data Size Set 7 Parameter 2	FTPEDSS7P2
– 0xE0000654–0xE000067F	Reserved	
– 0xE0000680–0xE00006FF	EQPE parameters	
– 0xE0000680–0xE0000683	MAPLE EQPE Y Buffer Mode Parameter	MEQYBMP
– 0xE0000684–0xE00007FF	Reserved	
– 0xE0000800–0xE0002FFF	CRPE_ULB Parameters	
– 0xE0000800–0xE00021FF	CRPE ULB Physical Channel parameters. These parameters are in groups of four. The parameters for channel 0 are in sequence, followed by those for channel 1, and so forth.	
– 0xE0000800–0xE0000E7F	MAPLE CRPE ULB Physical Channel 0–415 Base Address Parameter	MCUBPCH[0–415} BAP
– 0xE0000804–0xE0000E83	MAPLE CRPE ULB Physical Channel 0–415 Size Parameter	MCUBPCH[0–415] SZP
– 0xE0000808–0xE0000E87	MAPLE CRPE ULB Physical Channel 0–415 Write Pointer Parameter	MCUBPCH[0–415} WPP
– 0xE000080C–0xE0000E8B	MAPLE CRPE ULB Physical Channel 0–415 Output Buffer Interrupt Config Parameter	MCUBPCH[0–415} OBICP
– 0xE0000E8C–0xE00021FF	reserved	
– 0xE0002200–0xE000237F	CRPE ULB Group Configuration parameters. These parameters are in sets of four. The parameters for group 0 are in sequence, followed by those for group 1, and so forth.	
– 0xE0002200–0xE000225F	MAPLE CRPE ULB Group 0–23 Configuration 1 Parameter	MCUBG[0–23]C1P
– 0xE0002204–0xE0002263	MAPLE CRPE ULB Group0–23 Configuration 2 Parameter	MCUBG[0–23]C2P
– 0xE0002208–0xE0002267	MAPLE CRPE ULB Group 0–23 Configuration 3 Parameter	MCUBG[0–23]C3P
– 0xE000220C–0xE000226B	MAPLE CRPE ULB Group 0–23 Configuration 4 Parameter	MCUBG[0–23]C4P
– 0xE000226C–0xE000237F	reserved	
– 0xE0002380–0xE000243F	MAPLE CRPE ULB Antenna Descriptor 0–47 Parameter	MCUBANTD[0–47]P
– 0xE0002440–0xE0002443	MAPLE CRPE ULB Configuration Parameter	MCUBCP
– 0xE0002444–0xE0002447	MAPLE CRPE ULB Output Buffer Interrupt Configuration Parameter.	MCUBOBICP
– 0xE0002448–0xE00024FF	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE0002500–0xE00028FF	MAPLE CRPE ULB Finished Channels Buffer 0–511 Parameter	MCUBFCB[0–511]P
– 0xE0002900–0xE0002CFF	Group Core Descriptors Memory	
– 0xE0002D00–0xE0002FFF	reserved	
– 0xE0003000–0xE00033FF	CRPE_ULF parameters	
– 0xE0003000–0xE000301F	MAPLE CRPE-ULF Commands Input Buffer Address 0–7 Parameter	MCUFCIBA[0–7]P
– 0xE0003020–0xE000303F	MAPLE CRPE-ULF Commands Input Buffer Write Pointer 0–7 Parameter	MCUFCIBWP[0–7]P
– 0xE0003040–0xE000305F	MAPLE CRPE-ULF Commands Input Buffer Read Pointer 0–7 Parameter	MCUFCIBRP[0–7]P
– 0xE0003060–0xE000306F	Reserved	
– 0xE0003070–0xE0003073	MAPLE CRPE-ULF Interpolation Bypass General Configuration Parameter.	MCUFIBGCP
– 0xE0003074–0xE000307F	Reserved	
– 0x00003080–0x0000309F	MAPLE CRPE-ULF Interpolation Bypass Group Attributes 0–7 parameter	MCUFIBGA[0–7]P
– 0x000030A0–0x000030FF	MAPLE CRPE-ULF Interpolation Bypass Ant Address 0–23 Parameter	MCUFIBAA[0–23]P
– 0x00003100–0x000033FF	reserved	
– 0xE0003400–0xE00087FF	CPRE_DL parameters	
– 0xE0003400–0xE00073FF	CRPE Downlink Slot Channel parameters. These parameters are in groups of four. The parameters for channel 0 are in sequence, followed by those for channel 1, and so forth.	
– 0xE0003400–0xE0003FF3	MAPLE CRPE DownLink Slot Channel 0–1023 Parameter 0	MCDLSC[0–1023]P0
– 0xE0003404–0xE0003FF7	MAPLE CRPE DownLink Slot Channel 0–1023 Parameter 1	MCDLSC[0–1023]P1
– 0xE0003408–0xE0003FFB	MAPLE CRPE DownLink Slot Channel 0–1023 Parameter 2	MCDLSC[0–1023]P2
– 0xE000340C–0xE0003FFF	MAPLE CRPE DownLink Slot Channel 0–1023 Read Pointer Parameter	MCDLSC[0–1023] RPP
– 0xE0004000–0xE00073FF	reserved	
– 0xE0007400–0xE000807F	CRPE Downlink Fast Channel parameters. These parameters are in groups of four. The parameters for channel 0 are in sequence, followed by those for channel 1, and so forth.	
– 0xE0007400–0xE0008073	MAPLE CRPE DownLink Fast Channel 0–199 Parameter 0	MCDLFC[0–199]P0
– 0xE0007404–0xE0008077	MAPLE CRPE DownLink Fast Channel 0–199 Parameter 1	MCDLFC[0–199]P1
– 0xE0007408–0xE000807B	MAPLE CRPE DownLink Fast Channel 0–199 Parameter 2	MCDLFC[0–199]P2

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE000740C–0xE000807F	MAPLE CRPE DownLink Fast Channel 0–199 Read Pointer Parameter	MCDLFC[0–199]RPP
– 0xE0008080–0xE000817F	CRPE Downlink Output Buffer parameters. These parameters are in groups of four including a reserved block per parameter set). The parameters for channel 0 are in sequence, followed by those for channel 1, and so forth.	
– 0xE0008080–0xE0008173	MAPLE CRPE-DL Output Buffer 0–15 Base Address Parameter	MCDLOB[0–15]BAP
– 0xE0008084–0xE0008177	MAPLE CRPE-DL Output Buffer 0–15 Size Parameter	MCDLOB[0–15]SP
– 0xE0008088–0xE000817B	MAPLE CRPE-DL Output Buffer 0–15 Write Pointer Parameter	MCDOB[0–15]WPP
– 0xE000808C–0xE000817F	reserved	
– 0xE0008180–0xE0008183	MAPLE CRPE-DL Number of Channels Limit Parameter	MCDLNOCLP
– 0xE0008184–0xE00087FF	reserved	
– 0xE0008800–0xE0009FFF	reserved	
– 0xE000A000–0xE0011FFF	Buffer Descriptor (BD) Rings	
– 0xE0012000–0xE003FFFF	reserved	
– 0xE0040000–0xE007FFFF	eTVPE (register details begin on **page 26-513**)	
– 0xE0040000–0xE007DFFF	reserved	
– 0xE007E000	eTVPE Configuration 0 Register	TVPEC0R
– 0xE007E004–0xE007E03B	reserved	
– 0xE007E03C	eTVPE Aposteriori Quality Configuration Register	TVAQCR
– 0xE007E040–0xE007FFFF	reserved	
– 0xE0080000–0xE00BFFFF	eFTPE_0 (register details begin on **page 26-515**)	
– 0xE0080000–0xE00BE13F	reserved	
– 0xE00BE140–0xE00BE143	eFTPE_0 Data Size Register 0	EFTPE0DSR0
– 0xE00BE144–0xE00BE147	eFTPE_0 Data Size Register 1	EFTPE0DSR1
– 0xE00BE148–0xE00BE14B	eFTPE_0 Data Size Register 2	EFTPE0DSR2
– 0xE00BE14C–0xE00BE19F	reserved	
– 0xE00BE1A0	eFTPE_0 Configuration Register	EFTPE0CR
– 0xE00BE1A4–0xE00BE247	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE00BE248	eFTPE_0 ECC Interrupt Status Register	EFTPE0ECCISR
– 0xE00BE24C–0xE00BFFFF	reserved	
– 0xE00C0000–0xE00FFFFF	eFTPE_1 (register details begin on **page 26-515**)	
– 0xE0080000–0xE00FE13F	reserved	
– 0xE00FE140–0xE00FE143	eFTPE_1 Data Size Register 0	EFTPE1DSR0
– 0xE00FE144–0xE00FE147	eFTPE_1 Data Size Register 1	EFTPE1DSR1
– 0xE00FE148–0xE00FE14B	eFTPE_1 Data Size Register 2	EFTPE1DSR2
– 0xE00FE14C–0xE00FE19F	reserved	
– 0xE00FE1A0	eFTPE_1 Configuration Register	EFTPE1CR
– 0xE00FE1A4–0xE00FE247	reserved	
– 0xE00FE248	eFTPE_1 ECC Interrupt Status Register	EFTPE1ECCISR
– 0xE00BE24C–0xE00FFFFF	reserved	
– 0xE0100000–0xE013FFFF	eFTPE_2 (register details begin on **page 26-515**)	
– 0xE0100000–0xE013E13F	reserved	
– 0xE013E140–0xE013E143	eFTPE_2 Data Size Register 0	EFTPE2DSR0
– 0xE013E144–0xE013E147	eFTPE_2 Data Size Register 1	EFTPE2DSR1
– 0xE013E148–0xE013E14B	eFTPE_2 Data Size Register 2	EFTPE2DSR2
– 0xE013E14C–0xE013E19F	reserved	
– 0xE013E1A0	eFTPE_2 Configuration Register	EFTPE2CR
– 0xE013E1A4–0xE013E247	reserved	
– 0xE013E248	eFTPE_2 ECC Interrupt Status Register	EFTPE2ECCISR
– 0xE013E24C–0xE013FFFF	reserved	
– 0xE0140000–0xE017FFFF	reserved for DEPE	
– 0xE0180000–0xE01FFFFF	EQPE (register details begin on **page 26-521**)	
– 0xE0180000–0xE01CFA07	reserved	
– 0xE01CFA08	EQPE Threshold Register	EQ_THRESH

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE01CFA0C–0xE01CFB07	reserved	
– 0xE01CFB08	EQPE Event Register	EQ_ECCEVENT
– 0xE01CFB0C–0xE01FFFFF	reserved	
– 0xE0200000–0xE027FFFF	CRPE (register details begin on **page 26-525**)	
– 0xE0200000–0xE02035FF	CPRE input buffer	
– 0xE0203600–0xE021FFFF	reserved	
– 0xE0220000–0xE020807F	CRPE Downlink Virtual Antenna Gains Control Registers. These 9-byte registers are in groups of two. The registers for antenna 0 are in sequence, followed by those for antenna 1, and so forth.	
– 0xE0220000–0xE022C7AF	CRPE Downlink Virtual Antenna Gains Control Register 0_[0–511]	CDVAGLR0_[0–511]
– 0xE0220008–0xE022C7B7	CRPE Downlink Virtual Antenna Gains Control Register 1_[0–511]	CDVAGLR1_[0–511]
– 0xE022C7B8–0xE023DFFF	reserved	
– 0xE023E000	UL FAST General Configuration Register	ULFGCR
– 0xE023E004–0xE023E07F	reserved	
– 0xE023E080	UL FAST Interpolation Configuration Register 0	ULFICR0
– 0xE023E084	UL FAST Interpolation Configuration Register 1	ULFICR1
– 0xE023E088	UL FAST Interpolation Configuration Register 2	ULFICR2
– 0xE023E08C	UL FAST Interpolation Configuration Register 3	ULFICR3
– 0xE023E090	UL FAST Interpolation Configuration Register 4	ULFICR4
– 0xE023E094	UL FAST Interpolation Configuration Register 5	ULFICR5
– 0xE023E098	UL FAST Interpolation Configuration Register 6	ULFICR6
– 0xE023E09C	UL FAST Interpolation Configuration Register 7	ULFICR7
– 0xE023E0A0	UL FAST Interpolation Configuration Register 8	ULFICR8
– 0xE023E0A4	UL FAST Interpolation Configuration Register 9	ULFICR9
– 0xE023E0A8	UL FAST Interpolation Configuration Register 10	ULFICR10
– 0xE023E0AC	UL FAST Interpolation Configuration Register 11	ULFICR11
– 0xE023E0B0	UL FAST Interpolation Configuration Register 12	ULFICR12
– 0xE023E0B4	UL FAST Interpolation Configuration Register 13	ULFICR13
– 0xE023E0B8	UL FAST Interpolation Configuration Register 14	ULFICR14
– 0xE023E0BC	UL FAST Interpolation Configuration Register 15	ULFICR15
– 0xE023E0C0–0xE023E0FF	reserved	
– 0xE023E100	UL FAST Output Buffer 0 Base Configuration Register	ULFOB0BCR
– 0xE023E104	UL FAST Output Buffer 0 Attributes Configuration Register	ULFOB0ACR
– 0xE023E108	UL FAST Output Buffer 1 Base Configuration Register	ULFOB1BCR
– 0xE023E10C	UL FAST Output Buffer 1 Attributes Configuration Register	ULFOB1ACR
– 0xE023E110	UL FAST Output Buffer 2 Base Configuration Register	ULFOB2BCR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE023E114	UL FAST Output Buffer 2 Attributes Configuration Register	ULFOB2ACR
– 0xE023E118	UL FAST Output Buffer 3 Base Configuration Register	ULFOB3BCR
– 0xE023E11C	UL FAST Output Buffer 3 Attributes Configuration Register	ULFOB3ACR
– 0xE023E120	UL FAST Output Buffer 4 Base Configuration Register	ULFOB4BCR
– 0xE023E124	UL FAST Output Buffer 4 Attributes Configuration Register	ULFOB4ACR
– 0xE023E128	UL FAST Output Buffer 5 Base Configuration Register	ULFOB5BCR
– 0xE023E12C	UL FAST Output Buffer 5 Attributes Configuration Register	ULFOB05ACR
– 0xE023E130	UL FAST Output Buffer 6 Base Configuration Register	ULFOB6BCR
– 0xE023E134	UL FAST Output Buffer 6 Attributes Configuration Register	ULFOB6ACR
– 0xE023E138	UL FAST Output Buffer 7 Base Configuration Register	ULFOB7BCR
– 0xE023E13C	UL FAST Output Buffer 7 Attributes Configuration Register	ULFOB7ACR
– 0xE023E140	UL FAST Output Buffer 8 Base Configuration Register	ULFOB8BCR
– 0xE023E144	UL FAST Output Buffer 8 Attributes Configuration Register	ULFOB8ACR
– 0xE023E148	UL FAST Output Buffer 9 Base Configuration Register	ULFOB9BCR
– 0xE023E14C	UL FAST Output Buffer 9 Attributes Configuration Register	ULFOB9ACR
– 0xE023E150	UL FAST Output Buffer 10 Base Configuration Register	ULFOB10BCR
– 0xE023E154	UL FAST Output Buffer 10 Attributes Configuration Register	ULFOB10ACR
– 0xE023E158	UL FAST Output Buffer 11 Base Configuration Register	ULFOB11BCR
– 0xE023E15C	UL FAST Output Buffer 11 Attributes Configuration Register	ULFOB11ACR
– 0xE023E160	UL FAST Output Buffer 12 Base Configuration Register	ULFOB12BCR
– 0xE023E164	UL FAST Output Buffer 12 Attributes Configuration Register	ULFOB12ACR
– 0xE023E168	UL FAST Output Buffer 13 Base Configuration Register	ULFOB13BCR
– 0xE023E16C	UL FAST Output Buffer 13 Attributes Configuration Register	ULFOB13ACR
– 0xE023E170	UL FAST Output Buffer 14 Base Configuration Register	ULFOB14BCR
– 0xE023E174	UL FAST Output Buffer 14 Attributes Configuration Register	ULFOB14ACR
– 0xE023E178	UL FAST Output Buffer 15 Base Configuration Register	ULFOB15BCR
– 0xE023E17C	UL FAST Output Buffer 15 Attributes Configuration Register	ULFOB15ACR
– 0xE023E180	UL FAST Output Buffer 16 Base Configuration Register	ULFOB16BCR
– 0xE023E184	UL FAST Output Buffer 16 Attributes Configuration Register	ULFOB16ACR
– 0xE023E188	UL FAST Output Buffer 17 Base Configuration Register	ULFOB17BCR
– 0xE023E18C	UL FAST Output Buffer 17 Attributes Configuration Register	ULFOB17ACR
– 0xE023E190–0xE023E27F	reserved	
– 0xE023E280	ULF Event Status Register	ULFESR
– 0xE023E284–0xE023E287	reserved	
– 0xE023E288	ULF Interpolation Status Register	ULFISR
– 0xE023E28C	ULF DD Status Register	ULFDDSR
– 0xE023E290	ULF Command FIFO Status Register	ULFCFSR
– 0xE023E294	ULF Input Buffer Status Register	ULFIBSR
– 0xE023E298	ULF Time Status Register	ULFTSR
– 0xE023E29C	ULF ECC Status Register	ULFECCSR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE023E2A0–0xE023EFFF	reserved	
– 0xE023F000	CRPE Downlink Beam Forming Coefficients Values Control Register 0	CDBFCVLR0
– 0xE023F010	CRPE Downlink Beam Forming Coefficients Values Control Register 1	CDBFCVLR1
– 0xE023F020	CRPE Downlink Beam Forming Coefficients Values Control Register 2	CDBFCVLR2
– 0xE023F030	CRPE Downlink Beam Forming Coefficients Values Control Register 3	CDBFCVLR3
– 0xE023F040	CRPE Downlink Beam Forming Coefficients Values Control Register 4	CDBFCVLR4
– 0xE023F050	CRPE Downlink Beam Forming Coefficients Values Control Register 5	CDBFCVLR5
– 0xE023F060	CRPE Downlink Beam Forming Coefficients Values Control Register 6	CDBFCVLR6
– 0xE023F070	CRPE Downlink Beam Forming Coefficients Values Control Register 7	CDBFCVLR7
– 0xE023F080	CRPE Downlink Start Control Register	CDSLR
– 0xE023F084	CRPE Downlink TPC Command Control Register	CDTCLR
– 0xE023F088–0xE023F08F	reserved	
– 0xE023F090	CRPE Downlink Virtual Antennas Gain Command Control Register	CDVAGCLR
– 0xE023F094–0xE023F097	received	
– 0xE023F098	CRPE Downlink General Command Control Register	CDGCLR
– 0x0023F0A0–0x0023F0DF	CRPE Downlink Idle Period Control Register 0–15	CDIPLR[0–15]
– 0x0023F0E0–0x0023F0EF	CRPE Downlink Beam Forming Coefficients Timing Command Control Register 0–3	CDBFCTCLR[0–3]
– 0x0023F0F0–0x0023F0FF	CRPE Downlink Combined Chips Shift Command Control Registers 0–3	CDCCSCLR[0–3]
– 0x0023F100	CRPE Downlink Event Status Register	CDESR
– 0x0023F104	CRPE Downlink Processing Stage Status Register	CDPSSR
– 0x0023F108	CRPE Downlink ECC Status Register	CDECCSR
– 0xE023F10C	CRPE Downlink Rate Control Register	CDRLR
– 0xE023F110–0xE023F147	reserved	
– 0xE023F148	CRPE Downlink Normalization Value Register	CDNVR
– 0xE025F00C–0xE025F00F	reserved	
– 0xE025F010	CRPR-ULB Interpolation Weights 1 Sample 0 Configuration Register	CRUBIW1S0CR
– 0xE025F014	CRPR-ULB Interpolation Weights 1 Sample 1 Configuration Register	CRUBIW1S1CR
– 0xE025F018	CRPR-ULB Interpolation Weights 1 Sample 2 Configuration Register	CRUBIW1S2CR
– 0xE025F01C	CRPR-ULB Interpolation Weights 1 Sample 3 Configuration Register	CRUBIW1S3CR
– 0xE025F020	CRPR-ULB Interpolation Weights 1 Sample 4 Configuration Register	CRUBIW1S4CR
– 0xE025F024	CRPR-ULB Interpolation Weights 1 Sample 5 Configuration Register	CRUBIW1S5CR
– 0xE025F028	CRPR-ULB Interpolation Weights 1 Sample 6 Configuration Register	CRUBIW1S6CR
– 0xE025F02C	CRPR-ULB Interpolation Weights 1 Sample 7 Configuration Register	CRUBIW1S7CR
– 0xE025F030	CRPR-ULB Interpolation Weights 2 Sample 0 Configuration Register	CRUBIW2S0CR
– 0xE025F034	CRPR-ULB Interpolation Weights 2 Sample 1 Configuration Register	CRUBIW2S1CR
– 0xE025F038	CRPR-ULB Interpolation Weights 2 Sample 2 Configuration Register	CRUBIW2S2CR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xE025F03C	CRPR-ULB Interpolation Weights 2 Sample 3 Configuration Register	CRUBIW2S3CR
– 0xE025F040	CRPR-ULB Interpolation Weights 2 Sample 4 Configuration Register	CRUBIW2S4CR
– 0xE025F044	CRPR-ULB Interpolation Weights 2 Sample 5 Configuration Register	CRUBIW2S5CR
– 0xE025F048	CRPR-ULB Interpolation Weights 2 Sample 6 Configuration Register	CRUBIW2S6CR
– 0xE025F04C	CRPR-ULB Interpolation Weights 2 Sample 7 Configuration Register	CRUBIW2S7CR
– 0xE025F050	CRPR-ULB Group First Antenna 0 Configuration Register	CRUBGFA0CR
– 0xE025F054	CRPR-ULB Group First Antenna 1 Configuration Register	CRUBGFA1CR
– 0xE025F058	CRPR-ULB Group First Antenna 2 Configuration Register	CRUBGFA2CR
– 0xE025F05C	CRPR-ULB Group First Antenna 3 Configuration Register	CRUBGFA3CR
– 0xE025F060	CRPR-ULB Group First Antenna 4 Configuration Register	CRUBGFA4CR
– 0xE025F064	CRPR-ULB Group First Antenna 5 Configuration Register	CRUBIGFA5CR
– 0xE025F068	CRPR-ULB Group Number of Antenna 0 Configuration Register	CRUBGNO0CR
– 0xE025F06C	CRPR-ULB Group Number of Antenna 1 Configuration Register	CRUBGNO1CR
– 0xE025F070	CRPR-ULB Group Number of Antenna 2 Configuration Register	CRUBGNO2CR
– 0xE025F074	CRPR-ULB Group Number of Antenna 3 Configuration Register	CRUBGNO3CR
– 0xE025F078	CRPR-ULB Group Number of Antenna 4 Configuration Register	CRUBGNO4CR
– 0xE025F07C	CRPR-ULB Group Number of Antenna 5 Configuration Register	CRUBIGNO5CR
– 0xE025F080–0xE025F7FF	reserved	
– 0xE025F800	CRPE Uplink Batch Event Status Register	CRUBESR
– 0xE025F804–0xE025F81F	reserved	
– 0xE025F820	CRPE-ULB Output Sat Counter Status Register	CRUBOSCSR
– 0xE025F824	CRPE-ULB Interpolation Sat Counter Status Register	CRUBGNOA0CR
– 0xE025F828–0xE027FFFF	reserved	
– 0xE0280000–0xE03FFFFF	reserved	
– 0xE0400000–0xFEDFFFFF	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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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	
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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	
– 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	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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 *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
– 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	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 0xFFF10378–0xFFF1037B	Reserved	
– 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)*)	
– 0xFFF18000–0xFFF187FF	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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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	CLASS Priority Mapping Register 12	C0PMR12
– 0xFFF18834	CLASS Priority Mapping Register 13	C0PMR13
– 0xFFF18838	CLASS Priority Mapping Register 14	C0PMR14
– 0xFFF1883C– 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
– 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	CLASS Priority Auto Upgrade Value Register 12	C0PAVR12
– 0xFFF18874	CLASS Priority Auto Upgrade Value Register 13	C0PAVR13
– 0xFFF18878	CLASS Priority Auto Upgrade Value Register 14	C0PAVR14
– 0xFFF1887C– 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	CLASS Priority Auto Upgrade Control Register 12	C0PACR12
– 0xFFF188B4	CLASS Priority Auto Upgrade Control Register 13	C0PACR13
– 0xFFF188B8	CLASS Priority Auto Upgrade Control Register 14	C0PACR14

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF188BC–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	CLASS Error Address Register 12	C0EAR12
– 0xFFF189B4	CLASS Error Address Register 13	C0EAR13
– 0xFFF189B8	CLASS Error Address Register 14	C0EAR14
– 0xFFF189BC–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
– 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	CLASS Error Extended Address Register 12	C0EEAR12
– 0xFFF189F4	CLASS Error Extended Address Register 13	C0EEAR13
– 0xFFF189F8	CLASS Error Extended Address Register 14	C0EEAR14
– 0xFFF189FC–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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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	CLASS Initiator Profiling Configuration Register 12	C0IPCR12
– 0xFFF18A34	CLASS Initiator Profiling Configuration Register 13	C0IPCR13
– 0xFFF18A38	CLASS Initiator Profiling Configuration Register 14	C0IPCR14
– 0xFFF18A3C– 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	CLASS Initiator Watch Point Control Register 12	C0IWPCR12
– 0xFFF18A74	CLASS Initiator Watch Point Control Register 13	C0IWPCR13
– 0xFFF18A78	CLASS Initiator Watch Point Control Register 14	C0IWPCR14
– 0xFFF18A7C– 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
– 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	CLASS Arbitration Weight Register 12	C0AWR12
– 0xFFF18AB4	CLASS Arbitration Weight Register 13	C0AWR13

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF18AB8	CLASS Arbitration Weight Register 14	C0AWR14
– 0xFFF18ABC–0xFFF18C03	reserved	
– 0xFFF18C04	CLASS Start Address Decoder 1	C0SAD1
– 0xFFF18C08	CLASS Start Address Decoder 2	C0SAD2
– 0xFFF18C0C–0xFFF18C1B	reserved	
– 0xFFF18C1C	CLASS Start Address Decoder 7	C0SAD7
– 0xFFF18C20	CLASS Start Address Decoder 8	C0SAD8
– 0xFFF18C24	CLASS Start Address Decoder 9	C0SAD9
– 0xFFF18C28–0xFFF18C43	reserved	
– 0xFFF18C44	CLASS End Address Decoder 1	C0EAD1
– 0xFFF18C48	CLASS End Address Decoder 2	C0EAD2
– 0xFFF18C4C–0xFFF18C4B	reserved	
– 0xFFF18C5C	CLASS End Address Decoder 7	C0EAD7
– 0xFFF18C60	CLASS End Address Decoder 8	C0EAD8
– 0xFFF18C64	CLASS End Address Decoder 9	C0EAD9
– 0xFFF18C68–0xFFF18C83	reserved	
– 0xFFF18C84	CLASS Attributes Decoder 1	C0ATD1
– 0xFFF18C88–0xFFF18C9B	reserved	
– 0xFFF18C9C	CLASS Attributes Decoder 7	C0ATD7
– 0xFFF18CA0	CLASS Attributes Decoder 8	C0ATD8
– 0xFFF18CA4	CLASS Attributes Decoder 9	C0ATD9
– 0xFFF18CA8–0xFFF18D7F	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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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	
– 0xFFF18FC0	CLASS Arbitration Control Register	C0ACR
– 0xFFF18FC4–0xFFF18FFF	Reserved	
• 0xFFF19000–0xFFF1CFFF	reserved	
• 0xFFF1C000–0xFFF1FFFF	MAPLE-B2 (see **Chapter 26**, *Multi Accelerator Platform Engine, Baseband 2 (MAPLE-B2)*)	
– 0xFFF1C000–0xFFF1CFFF	Reserved	
– 0xFFF1D000	PSIF Command Register	PCR
– 0xFFF1D004–0xFFF1D5FF	Reserved	
– 0xFFF1D600	PSIF PIC Event Register 0	PSPICER0
– 0xFFF1D604	PSIF PIC Event Register 1	PSPICER1
– 0xFFF1D608	PSIF PIC Event Register 2	PSPICER2
– 0xFFF1D60C	PSIF PIC Edge/Level Register	PSPICELR
– 0xFFF1D610	PSIF PIC Mask Register 0	PSPICMR0
– 0xFFF1D614	PSIF PIC Mask Register 1	PSPICMR1
– 0xFFF1D618	PSIF PIC Mask Register 2	PSPICMR2
– 0xFFF1D61C	PSIF PIC Interrupt/Assertion Clocks Registers	PSPICIACR
– 0xFFF1D620–0xFFF1FFFF	reserved	
• 0xFFF20000–0xFFF21FFF	DDR Controller (see **Chapter 12**, *DDR SDRAM Memory Controller*)	
– 0xFFF20000	Chip Select 0 Bounds	CS0_BNDS
– 0xFFF20004–0xFFF20007	reserved	
– 0xFFF20008	Chip Select 1 Bounds	CS1_BNDS
– 0xFFF2000C–0xFFF2007F	reserved	
– 0xFFF20080	Chip Select 0 Configuration	CS0_CONFIG
– 0xFFF20084	Chip Select 1 Configuration	CS1_CONFIG
– 0xFFF20088–0xFFF200BF	reserved	
– 0xFFF200C0	Chip Select 0 Configuration 2	CS0_CONFIG_2

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF200C4	Chip Select 1 Configuration 2	CS1_CONFIG_2
– 0xFFF200C8–0xFFF200FF	reserved	
– 0xFFF20100	DDR SDRAM Timing Configuration 3	TIMING_CFG_3
– 0xFFF20104	DDR SDRAM Timing Configuration 0	TIMING_CFG_0
– 0xFFF20108	DDR SDRAM Timing Configuration 1	TIMING_CFG_1
– 0xFFF2010C	DDR SCRAM Timing Configuration 2	TIMING_CFG_2
– 0xFFF20110	DDR SDRAM Control Configuration	DDR_SDRAM_CFG
– 0xFFF20114	DDR SDRAM Control Configuration 2	DDR_SDRAM_CFG_2
– 0xFFF20118	DDR SDRAM Mode Configuration	DDR_SDRAM_MODE
– 0xFFF2011C	DDR SDRAM Mode Configuration 2	DDR_SDRAM_MODE_2
– 0xFFF20120	DDR SDRAM Mode Control	DDR_SCRAM_MD_CNTL
– 0xFFF20124	DDR SDRAM Interval Configuration	DDR_SDRAM_INTERVAL
– 0xFFF20128	DDR SDRAM Data Initialization	DDR_DATA_INIT
– 0xFFF2012C–0xFFF2012F	reserved	
– 0xFFF20130	DDR SDRAM Clock Control Configuration Register	DDR_SDRAM_CLK_CNTL
– 0xFFF20134–0xFFF20147	reserved	
– 0xFFF20148	DDR SDRAM Initialization Address Register	DDR_INIT_ADDR
– 0xFFF2014C	DDR Initialization Enable Register	DDR_INIT_EN
– 0xFFF20150–0xFFF2015F	reserved	
– 0xFFF20160	DDR SDRAM Timing Configuration 4 Register	TIMING_CFG_4
– 0xFFF20164	DDR SDRAM Timing Configuration 5 Register	TIMING_CFG_5
– 0xFFF20168–0xFFF2016F	reserved	
– 0xFFF20170	DDR ZQ Calibration Control Register	DDR_ZQ_CNTL
– 0xFFF20174	DDR Write Leveling Control Register	DDR_WRLVL_CNTL
– 0xFFF22178	reserved	
– 0xFFF2017C	DDR Self Refresh Counter	DDR_SR_CNTR
– 0xFFF20180	DDR SDRAM Register Control Words 1 Register	DDR_SDRAM_RCW_1
– 0xFFF20184	DDR SDRAM Register Control Words 2 Register	DDR_SDRAM_RCW_2
– 0xFFF20188–0xFFF2018F	reserved	
– 0xFFF20190	DDR Write Leveling Control 2 Register	DDR_WRLVL_CNTL_2
– 0xFFF20194	DDR Write Leveling Control 3	DDR_WRLVL_CNTL_3
– 0xFFF20198–0xFFF2019F	reserved	
– 0xFFF20200	DDR SDRAM Mode 3 Configuration Register	DDR_SDRAM_MODE_3
– 0xFFF20204	DDR SDRAM Mode Configuration 4 Register	DDR_SDRAM_MODE_4
– 0xFFF20208–0xFFF20B1F	reserved	
– 0xFFF20B20	DDR Debug Status Register 1	DDRDSR_1

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF20B24	DDR Debug Status Register 2	DDRDSR_2
– 0xFFF20B28	DDR Control Driver Register 1	DDRCDR_1
– 0xFFF20B2C	DDR Control Driver Register 2	DDRCDR_2
– 0xFFF20B30–0xFFF20BF7	reserved	
– 0xFFF20BF8	DDR IP Block Revision 1	DDR_IP_REV1
– 0xFFF20BFC	DDR IP Block Revision 2	DDR_IP_REV2
– 0xFFF20C00–0xFFF20CFF	reserved	
– 0xFFF20D00	DDR Memory Test Control Register	DDR_MTCR
– 0xFFF20D04–0xFFF20D1F	reserved	
– 0xFFF20D20–0xFFF20D47	DDR Data Memory Test Pattern 0–9	DDR_MTP[0–9]
– 0xFFF20D48–0xFFF20DFF	reserved	
– 0xFFF20E00	Memory Data Path Error Injection Mask High	DDR_ERR_INJECT_HI
– 0xFFF20E04	Memory Data Path Error Injection Mask Low	DDR_ERR_INJECT_LO
– 0xFFF20E08	Memory Data Path Error Injection Mask ECC	DDR_ERR_INJECT
– 0xFFF20E0C–0xFFF20E1F	reserved	
– 0xFFF20E20	Memory Data Path Read Capture High	CAPTURE_DATA_HI
– 0xFFF20E24	Memory Data Path Read Capture Low	CAPTURE-DATA_LO
– 0xFFF20E28	Memory Data Path Read Capture ECC	CAPTURE_ECC
– 0xFFF20E2C–0xFFF20E3F	reserved	
– 0xFFF20E40	Memory Error Detect	ERR_DETECT
– 0xFFF20E44	Memory Error Disable	ERR_DISABLE
– 0xFFF20E48	Memory Error Interrupt Enable	ERR_INT_EN
– 0xFFF20E4C	Memory Error Attributes Capture	CAPTURE_ATTRIBUTES
– 0xFFF20E50	Memory Error Address Capture	CAPTURE_ADDRESS
– 0xFFF20E54–0xFFF20E57	reserved	
– 0xFFF20E58	Single-Bit ECC Memory Error Management	ERR_SBE
– 0xFFF20E5C–0xFFF21FFF	reserved	
• 0xFFF22000–0xFFF23FFF	Reserved	
• 0xFFF24000–0xFFF2407F	Clocks (see **Chapter 7**, *Clocks*)	
– 0xFFF24000	System Clock Control Register	SCCR
– 0xFFF24004–0xFFF2407F	reserved	
• 0xFFF24080–0xFFF247FF	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF24800–0xFFF248FF	Reset (see **Chapter 5**, *Reset*)	
– 0xFFF249800	Reset Configuration Word Low Register	RCWLR
– 0xFFF248904	Reset Configuration Word High Register	RCWHR
– 0xFFF248908–0xFFF2480F	reserved	
– 0xFFF249810	Reset Status Register	RSR
– 0xFFF249814–0xFFF24817	reserved	
– 0xFFF248918	Reset Protection Register	RPR
– 0xFFF2481C	Reset Control Register	RCR
– 0xFFF249820	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	
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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	
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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	
• 0xFFF26400–0xFFF265FF	Timer_32b 0 (see **Chapter 21**, *Timers*)	
– 0xFFF26400	Timer_32b 0 Channel 0 Compare 1 Register	TMR_32b_0_CMP10
– 0xFFF26404	Timer_32b 0 Channel 0 Compare 2 Register	TMR_32b_0_CMP20
– 0xFFF26408	Timer_32b 0 Channel 0 Capture Register	TMR_32b_0_CAP0
– 0xFFF2640C	Timer_32b 0 Channel 0 Load Register	TMR_32b_0_LOAD0
– 0xFFF26410	Timer_32b 0 Channel 0 Hold Register	TMR_32b_0_HOLD0
– 0xFFF26414	Timer_32b 0 Channel 0 Counter Register	TMR_32b_0_CNTR0
– 0xFFF26418	Timer_32b 0 Channel 0 Control Register	TMR_32b_0_CTL0
– 0xFFF2641C	Timer_32b 0 Channel 0 Status and Control Register	TMR_32b_0_SCTL0
– 0xFFF26420	Timer_32b 0 Channel 0 Load 1 Register	TMR_32b_0_CMPLD10
– 0xFFF26424	Timer_32b 0 Channel 0 Load 2 Register	TMR_32b_0_CMPLD20
– 0xFFF26428	Timer_32b 0 Channel 0 Comparator Status and Control Register	TMR_32b_0_COMSC0

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF2642C–0xFFF2643F	reserved	
– 0xFFF26440	Timer_32b 0 Channel 1 Compare 1 Register	TMR_32b_0_CMP11
– 0xFFF26444	Timer_32b 0 Channel 1 Compare 2 Register	TMR_32b_0_CMP21
– 0xFFF26448	Timer_32b 0 Channel 1 Capture Register	TMR_32b_0_CAP1
– 0xFFF2644C	Timer_32b 0 Channel 1 Load Register	TMR_32b_0_LOAD1
– 0xFFF26450	Timer_32b 0 Channel 1 Hold Register	TMR_32b_0_HOLD1
– 0xFFF26454	Timer_32b 0 Channel 1 Counter Register	TMR_32b_0_CNTR1
– 0xFFF26458	Timer_32b 0 Channel 1 Control Register	TMR_32b_0_CTL1
– 0xFFF2645C	Timer_32b 0 Channel 1 Status and Control Register	TMR_32b_0_SCTL1
– 0xFFF26460	Timer_32b 0 Channel 1 Load 1 Register	TMR_32b_0_CMPLD11
– 0xFFF26464	Timer_32b 0 Channel 1 Load 2 Register	TMR_32b_0_CMPLD21
– 0xFFF26468	Timer_32b 0 Channel 1 Comparator Status and Control Register	TMR_32b_0_COMSC1
– 0xFFF2646C–0xFFF2647F	reserved	
– 0xFFF26480	Timer_32b 0 Channel 2 Compare 1 Register	TMR_32b_0_CMP12
– 0xFFF26484	Timer_32b 0 Channel 2 Compare 2 Register	TMR_32b_0_CMP22
– 0xFFF26488	Timer_32b 0 Channel 2 Capture Register	TMR_32b_0_CAP2
– 0xFFF2648C	Timer_32b 0 Channel 2 Load Register	TMR_32b_0_LOAD2
– 0xFFF26490	Timer_32b 0 Channel 2 Hold Register	TMR_32b_0_HOLD2
– 0xFFF26494	Timer_32b 0 Channel 2 Counter Register	TMR_32b_0_CNTR2
– 0xFFF26498	Timer_32b 0 Channel 2 Control Register	TMR_32b_0_CTL2
– 0xFFF2649C	Timer_32b 0 Channel 2 Status and Control Register	TMR_32b_0_SCTL2
– 0xFFF264A0	Timer_32b 0 Channel 2 Load 1 Register	TMR_32b_0_CMPLD12
– 0xFFF264A4	Timer_32b 0 Channel 2 Load 2 Register	TMR_32b_0_CMPLD22
– 0xFFF264A8	Timer_32b 0 Channel 2 Comparator Status and Control Register	TMR_32b_0_COMSC2
– 0xFFF264AC–0xFFF264BF	reserved	
– 0xFFF264C0	Timer_32b 0 Channel 3 Compare 1 Register	TMR_32b_0_CMP13
– 0xFFF264C4	Timer_32b 0 Channel 3 Compare 2 Register	TMR_32b_0_CMP23
– 0xFFF264C8	Timer_32b 0 Channel 3 Capture Register	TMR_32b_0_CAP3
– 0xFFF264CC	Timer_32b 0 Channel 3 Load Register	TMR_32b_0_LOAD3
– 0xFFF264D0	Timer_32b 0 Channel 3 Hold Register	TMR_32b_0_HOLD3
– 0xFFF264D4	Timer_32b 0 Channel 3 Counter Register	TMR_32b_0_CNTR3
– 0xFFF264D8	Timer_32b 0 Channel 3 Control Register	TMR_32b_0_CTL3
– 0xFFF264DC	Timer_32b 0 Channel 3 Status and Control Register	TMR_32b_0_SCTL3
– 0xFFF264E0	Timer_32b 0 Channel 3 Load 1 Register	TMR_32b_0_CMPLD13
– 0xFFF264E4	Timer_32b 0 Channel 3 Load 2 Register	TMR_32b_0_CMPLD23
– 0xFFF264E8	Timer_32b 0 Channel 3 Comparator Status and Control Register	TMR_32b_0_COMSC3
– 0xFFF264EC–0xFFF264FF	reserved	
– 0xFFF26500	Timer_32b 0 Global System Timer Register	TMR_32b_0_GLB
– 0xFFF26504	Timer_32b 0 Global System Timer Control Register	TMR_32b_0_GLBCTL

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF26508	Timer_32b 0 Timer Set and Forget Register	TMR_32b_0_SAF3
– 0xFFF2650C	Timer_32b 0 Timer Clear Lock Register	TMR_32b_0_CLRL
– 0xFFF26510–0xFFF265FF	reserved	
• 0xFFF26600–0xFFF267FF	Timer_32b 1 (see **Chapter 21**, *Timers*)	
– 0xFFF26600	Timer_32b 1 Channel 0 Compare 1 Register	TMR_32b_1_CMP10
– 0xFFF26604	Timer_32b 1 Channel 0 Compare 2 Register	TMR_32b_1_CMP20
– 0xFFF26608	Timer_32b 1 Channel 0 Capture Register	TMR_32b_1_CAP0
– 0xFFF2660C	Timer_32b 1 Channel 0 Load Register	TMR_32b_1_LOAD0
– 0xFFF26610	Timer_32b 1 Channel 0 Hold Register	TMR_32b_1_HOLD0
– 0xFFF26614	Timer_32b 1 Channel 0 Counter Register	TMR_32b_1_CNTR0
– 0xFFF26618	Timer_32b 1 Channel 0 Control Register	TMR_32b_1_CTL0
– 0xFFF2661C	Timer_32b 1 Channel 0 Status and Control Register	TMR_32b_1_SCTL0
– 0xFFF26620	Timer_32b 1 Channel 0 Load 1 Register	TMR_32b_1_CMPLD10
– 0xFFF26624	Timer_32b 1 Channel 0 Load 2 Register	TMR_32b_1_CMPLD20
– 0xFFF26628	Timer_32b 1 Channel 0 Comparator Status and Control Register	TMR_32b_1_COMSC0
– 0xFFF2662C–0xFFF2663F	reserved	
– 0xFFF26640	Timer_32b 1 Channel 1 Compare 1 Register	TMR_32b_1_CMP11
– 0xFFF26644	Timer_32b 1 Channel 1 Compare 2 Register	TMR_32b_1_CMP21
– 0xFFF26648	Timer_32b 1 Channel 1 Capture Register	TMR_32b_1_CAP1
– 0xFFF2664C	Timer_32b 1 Channel 1 Load Register	TMR_32b_1_LOAD1
– 0xFFF26650	Timer_32b 1 Channel 1 Hold Register	TMR_32b_1_HOLD1
– 0xFFF26654	Timer_32b 1 Channel 1 Counter Register	TMR_32b_1_CNTR1
– 0xFFF26658	Timer_32b 1 Channel 1 Control Register	TMR_32b_1_CTL1
– 0xFFF2665C	Timer_32b 1 Channel 1 Status and Control Register	TMR_32b_1_SCTL1
– 0xFFF26660	Timer_32b 1 Channel 1 Load 1 Register	TMR_32b_1_CMPLD11
– 0xFFF26664	Timer_32b 1 Channel 1 Load 2 Register	TMR_32b_1_CMPLD21
– 0xFFF26668	Timer_32b 1 Channel 1 Comparator Status and Control Register	TMR_32b_1_COMSC1
– 0xFFF2666C–0xFFF2667F	reserved	
– 0xFFF26680	Timer_32b 1 Channel 2 Compare 1 Register	TMR_32b_1_CMP12
– 0xFFF26684	Timer_32b 1 Channel 2 Compare 2 Register	TMR_32b_1_CMP22
– 0xFFF26688	Timer_32b 1 Channel 2 Capture Register	TMR_32b_1_CAP2
– 0xFFF2668C	Timer_32b 1 Channel 2 Load Register	TMR_32b_1_LOAD2
– 0xFFF26690	Timer_32b 1 Channel 2 Hold Register	TMR_32b_1_HOLD2
– 0xFFF26694	Timer_32b 1 Channel 2 Counter Register	TMR_32b_1_CNTR2
– 0xFFF26698	Timer_32b 1 Channel 2 Control Register	TMR_32b_1_CTL2
– 0xFFF2669C	Timer_32b 1 Channel 2 Status and Control Register	TMR_32b_1_SCTL2
– 0xFFF266A0	Timer_32b 1 Channel 2 Load 1 Register	TMR_32b_1_CMPLD12
– 0xFFF266A4	Timer_32b 1 Channel 2 Load 2 Register	TMR_32b_1_CMPLD22
– 0xFFF266A8	Timer_32b 1 Channel 2 Comparator Status and Control Register	TMR_32b_1_COMSC2

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF266AC–0xFFF266BF	reserved	
– 0xFFF266C0	Timer_32b 1 Channel 3 Compare 1 Register	TMR_32b_1_CMP13
– 0xFFF266C4	Timer_32b 1 Channel 3 Compare 2 Register	TMR_32b_1_CMP23
– 0xFFF266C8	Timer_32b 1 Channel 3 Capture Register	TMR_32b_1_CAP3
– 0xFFF266CC	Timer_32b 1 Channel 3 Load Register	TMR_32b_1_LOAD3
– 0xFFF266D0	Timer_32b 1 Channel 3 Hold Register	TMR_32b_1_HOLD3
– 0xFFF266D4	Timer_32b 1 Channel 3 Counter Register	TMR_32b_1_CNTR3
– 0xFFF266D8	Timer_32b 1 Channel 3 Control Register	TMR_32b_1_CTL3
– 0xFFF266DC	Timer_32b 1 Channel 3 Status and Control Register	TMR_32b_1_SCTL3
– 0xFFF266E0	Timer_32b 1 Channel 3 Load 1 Register	TMR_32b_1_CMPLD13
– 0xFFF266E4	Timer_32b 1 Channel 3 Load 2 Register	TMR_32b_1_CMPLD23
– 0xFFF266E8	Timer_32b 1 Channel 3 Comparator Status and Control Register	TMR_32b_1_COMSC3
– 0xFFF266EC–0xFFF266FF	reserved	
– 0xFFF26700	Timer_32b 1 Global System Timer Register	TMR_32b_1_GLB
– 0xFFF26704	Timer_32b 1 Global System Timer Control Register	TMR_32b_1_GLBCTL
– 0xFFF26708	Timer_32b 1 Timer Set and Forget Register	TMR_32b_1_SAF3
– 0xFFF2670C	Timer_32b 1 Timer Clear Lock Register	TMR_32b_1_CLRL
• 0xFFF26710–0xFFF26BFF	reserved	
• 0xFFF26C00–0xFFF26C3F	UART (see **Chapter 20**, *UART*)	
– 0xFFF26C00	SCI Baud-Rate Register	SCIBR
– 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	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 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
– 0xFFF28040–0xFFF2804F	reserved	
– 0xFFF28050	General Status Register 3	GSR3
– 0xFFF28054–0xFFF28060	reserved	
– 0xFFF28064	General control Register 6	GCR6
– 0xFFF28068	General Control Register 7	GCR7
– 0xFFF2806C	General Control Register 8	GCR8
– 0xFFF28070–0xFFF28073	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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 0xFFF28114–0xFFF28117	reserved	
– 0xFFF28118	General Control Register 13	GCR13
– 0xFFF2811C	General Status Register 8	GSR8
– 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	DDRC_GCR
– 0xFFF28130–0xFFF28137	reserved	
– 0xFFF28138	Core Subsystem Slave Port General Configuration Register	CORE_SLV_GCR
– 0xFFF2813C	QUICC Engine Input General Control Register	QE_PIO_IN_GCR
– 0xFFF28140	QUICC Engine Output General Status Register	QE_PIO_OUT_GCR
– 0xFFF28144–0xFFF2814F	reserved	
– 0xFFF28150	L2Q Arbitration Control for Core Subsystems 0 and 1	MEX_T2_0_1_ARB
– 0xFFF28154	L2Q Arbitration Control for Core Subsystems 2 and 3	MEX_TX_2_3_ARB
– 0xFFF28158	L2Q Arbitration Control for Core Subsystems 4 and 5	MEX_TX_4_5_ARB
– 0xFFF2815C–0xFFF2816F	reserved	
– 0xFFF28170	General Interrupt 6	GIR6
– 0xFFF28174	General Interrupt Enable Register 6 for Core 0	GIER6_0
– 0xFFF28178	General Interrupt Enable Register 6 for Core 1	GIER6_1
– 0xFFF2817C	General Interrupt Enable Register 6 for Core 2	GIER6_2
– 0xFFF28180	General Interrupt Enable Register 6 for Core 3	GIER6_3

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF28184	General Interrupt Enable Register 6 for Core 4	GIER6_4
– 0xFFF28188	General Interrupt Enable Register 6 for Core 5	GIER6_5
– 0xFFF2818C–0xFFF28193	reserved	
– 0xFFF28194	General Interrupt Register 7	GIR7
– 0xFFF28198	General Interrupt Enable Register 7 for Core 0	GIER7_0
– 0xFFF2819C	General Interrupt Enable Register 7 for Core 1	GIER7_1
– 0xFFF281A0	General Interrupt Enable Register 7 for Core 2	GIER7_2
– 0xFFF281A4	General Interrupt Enable Register 7 for Core 3	GIER7_3
– 0xFFF281A8	General Interrupt Enable Register 7 for Core 4	GIER7_4
– 0xFFF281AC	General Interrupt Enable Register 7 for Core 5	GIER7_5
– 0xFFF281B0–0xFFF281B7	reserved	
– 0xFFF281B8	DDR View Through L2 Memory Core Subsystems 0–3	L2MAP_0_3
– 0xFFF281BC	DDR View Through L2 Memory Core Subsystems 4–5	L2MAP_4_5
– 0xFFF281C0–0xFFF281DB	reserved	
– 0xFFF281DC	eMSG to QUICC Engine External Request Enable	CPCEER
– 0xFFF281E0	RGMII1 High Resolution Delay Register	UCC1_DELAY_HR
– 0xFFF281E4	RGMII2 High Resolution Delay Register	UCC3_DELAY_HR
– 0xFFF281E8	General Interrupt Register 8	GIR8
– 0xFFF281EC–0xFFF281EF	reserved	
– 0xFFF281F0	CPRI to MAPLE External Request Enable	MAPLE_EXT_REQ_EN_1
– 0xFFF281F4–0xFFF281FF	reserved	
• 0xFFF28200–0xFFF3FFFF	reserved	
• 0xFFF40000–0xFFF5FFFF	CPRI (see **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*).	
CPRI Framer Registers		
• 0xFFF40004	CPRI Status Register	CPRI_STATUS
• 0xFFF40008	CPRI Configuration	CPRI_CONFIG
• 0xFFF4001C	CPRI Receive Line Coding Violation Counter	CPRI_LCV
• 0xFFF40020	CPRI Recovered BFN Counter	CPRI_BFN
• 0xFFF40024	CPRI Recovered HFN Counter	CPRI_HFN
• 0xFFF4002C	CPRI Hardware Reset from Control Word	CPRI_HW_RESET
• 0xFFF4003C	CPRI Control and Management Configuration	CPRI_CM_CONFIG
• 0xFFF40040	CPRI Control and Management Status	CPRI_CM_STATUS
• 0xFFF40048	CPRI Receive Delay	CPRI_RX_DELAY
• 0xFFF4004C	CPRI Round Trip Delay	CPRI_ROUND_DELAY
• 0xFFF40050	CPRI Extended Delay Measurement Configuration	CPRI_EX_DELAY_CONFIG
• 0xFFF40054	CPRI Extended Delay Measurement Status	CPRI_EX_DELAY_STATUS

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF40058	CPRI Transmit Protocol Version	CPRI_TX_PROT_VER
• 0xFFF4005C	CPRI Transmit Scrambler Seed	CPRI_TX_SCR_SEED
• 0xFFF40060	CPRI Receive Scrambler Seed	CPRI_RX_SCR_SEED
• 0xFFF40080	CPRI SerDes Interface Configuration	CPRI_SERDES_CONFIG
• 0xFFF400C0	CPRI Mapping Configuration	CPRI_MAP_CONFIG
• 0xFFF400C4	CPRI Mapping Counter Configuration	CPRI_MAP_CNT_CONFIG
• 0xFFF400D0	CPRI Mapping Table Configuration	CPRI_MAP_TBL_CONFIG
• 0xFFF400E4	CPRI Mapping RX AxC Container Block Offset	CPRI_MAP_OFFSET_RX
• 0xFFF400E8	CPRI Mapping TX AxC Container Block Offset	CPRI_MAP_OFFSET_TX
• 0xFFF400F0	Offset for CPRI_TX_START Synchronization Output	CPRI_START_OFFSET_TX
• 0xFFF40100	CPRI Mapping Buffer RX Status Register	CPRI_IQ_RX_BUF_ STATUS
• 0xFFF40120	CPRI Mapping Buffer TX Status Register	CPRI_IQ_TX_BUF_ STATUS
• 0xFFF40200	Ethernet Receive Status	ETH_RX_STATUS
• 0xFFF40208	Ethernet Feature Enable/Disable and Interrupt Enable Bits	ETH_CONFIG_1
• 0xFFF4020C	Ethernet Miscellaneous Configuration	ETH_CONFIG_2
• 0xFFF40210	Ethernet RX Packet Discard	ETH_RX_CONTROL
• 0xFFF40228	Ethernet RX Additional Status	ETH_RX_EX_STATUS
• 0xFFF4022C	Ethernet MSB of MAC Address (16 bits)	ETH_ADDR_MSB
• 0xFFF40230	Ethernet LSB of MAC Address (32 bits)	ETH_ADDR_LSB
• 0xFFF40234	Ethernet Small 32 Entries Hash Table to Filter Multicast Traffic	ETH_HASH_TABLE
• 0xFFF40244	Ethernet Configuration 3	ETH_CONFIG_3
• 0xFFF40248	Ethernet Receive Frame Counter	ETH_CNT_RX_FRAME
• 0xFFF4024C	Reserved	
• 0xFFF40300	HDLC Receive Status	HDLC_RX_STATUS
• 0xFFF40308	HDLC Feature Enable/Disable and Interrupt Enable Bits	HDLC_CONFIG_1
• 0xFFF4030C	HDLC Miscellaneous Configuration	HDLC_CONFIG_2
• 0xFFF40310	HDLC RX Packet Discard	HDLC_RX_CONTROL
• 0xFFF40328	HDLC RX External Status	HDLC_RX_EX_STATUS
• 0xFFF40344	HDLC Configuration 3	HDLC_CONFIG_3
• 0xFFF40348	HDLC Receive Frame Counter	HDLC_CNT_RX_FRAME
• 0xFFF4034C	Reserved	
CPRI Complex Registers		
• 0xFFF40400	Receive IQ MBus Transaction Size	CPRInRIQMTS
• 0xFFF40404	Receive IQ Second Destination Mbus Transaction Size	CPRInRIQSDMTS
• 0xFFF40408	Transmit IQ MBus Transaction Size	CPRInTIQMTS
• 0xFFF4040c	Receive VSS MBus Transaction Size	CPRInRVSSMTS
• 0xFFF40410	Transmit VSS MBus Transaction Size	CPRInTVSSMTS
• 0xFFF4045c	Receive IQ Second Dest Base Address	CPRInRIQSDBA
• 0xFFF40460	Receive IQ Buffer Size	CPRInRIQBS
• 0xFFF40464	Receive IQ Second Destination Buffer Size	CPRInRIQSDBS

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF40468	Transmit IQ Buffer Size	CPRInTIQBS
• 0xFFF4046C	Receive VSS Buffer Size	CPRInRVSSBS
• 0xFFF40470	Transmit VSS Buffer Size	CPRInTVSSBS
• 0xFFF40474	Receive ETH Buffer Size	CPRInRETHBS
• 0xFFF40478	Receive HDLC Buffer Size	CPRInRHDLCBS
• 0xFFF4047C	Receive VSS Buffer Base Address	CPRInRVSSBA
• 0xFFF40480	Transmit VSS Buffer Base Address	CPRInTVSSBA
• 0xFFF40484	Receive Ethernet BD Ring Base Address	CPRInREBDRBA
• 0xFFF40488	Transmit Ethernet BD Ring Base Address	CPRInTEBDRBA
• 0xFFF4048C	Receive HDLC BD Ring Base Address	CPRInRHBDRBA
• 0xFFF40490	Transmit HDLC BD Ring Base Address	CPRInTHBDRBA
• 0xFFF40494	Receive Ethernet Buffer Descriptor Ring Size	CPRInREBDRS
• 0xFFF40498	Transmit Ethernet Buffer Descriptor Ring Size	CPRInTEBDRS
• 0xFFF4049C	Receive HDLC Buffer Descriptor Ring Size	CPRInRHBDRS
• 0xFFF404A0	Transmit HDLC Buffer Descriptor Ring Size	CPRInTHBDRS
• 0xFFF404A8	Receive General CPRI Mode	CPRInRGCM
• 0xFFF404AC	Transmit General CPRI Mode	CPRInTGCM
• 0xFFF404B4	Transmit Synchronization Configuration Register	CPRInTSCR
• 0xFFF404B8	Transmit CPRI Framer Buffer Size	CPRInTCFBS
• 0xFFF404BC	Tx Control Table Insert Enable 1	CPRInTCTIE1
• 0xFFF404C0	Tx Control Table Insert Enable 2	CPRInTCTIE2
• 0xFFF404C8	Timer Configuration	CPRInTMRC
• 0xFFF404CC	Receive Frame Pulse Width	CPRInRFPW
• 0xFFF404D0	Transmit Frame Pulse Width	CPRInTFPW
Control Registers		
• 0xFFF40500	Receive Control Register	CPRInRCR
• 0xFFF40504	Transmit Control Register	CPRInTCR
• 0xFFF40508	Receive AxC Control Register	CPRInRACCR
• 0xFFF40510	Transmit AxC Control Register	CPRInTACCR
• 0xFFF40514	Receive Control Attribute	CPRInRCA
• 0xFFF40518	Receive Control Data0	CPRInRCD0
• 0xFFF4051C	Receive Control Data1	CPRInRCD1
• 0xFFF40520	Receive Control Data2	CPRInRCD2
• 0xFFF40524	Transmit Control Attribute	CPRInTCA
• 0xFFF40528	Transmit Control Data0	CPRInTCD0
• 0xFFF4052C	Transmit Control Data1	CPRInTCD1
• 0xFFF40530	Transmit Control Data2	CPRInTCD2
• 0xFFF40534	Receive IQ First Threshold	CPRInRIQFT
• 0xFFF40538	Receive IQ Second Threshold	CPRInRIQST
• 0xFFF4053C	Receive IQ Threshold	CPRInRIQT
• 0xFFF40540	Transmit IQ First Threshold	CPRInTIQFT
• 0xFFF40544	Transmit IQ Second Threshold	CPRInTIQST

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF40548	Transmit IQ Threshold	CPRInTIQT
• 0xFFF4054C	Receive VSS Threshold	CPRInRVSST
• 0xFFF40550	Transmit VSS Threshold	CPRInTVSST
• 0xFFF40554	Receive Ethernet Coalescing Threshold	CPRInRETHCT
• 0xFFF40558	Transmit Ethernet Coalescing Threshold	CPRInTETHCT
• 0xFFF40560	CPRI Receive Control and Timing interrupt Enable Register	CPRInRCIER
• 0xFFF40564	CPRI transmit Control and Timing Interrupt Enable Register	CPRInTCIER
• 0xFFF40568	Receive IQ Threshold Second Destination	CPRInRIQTSD
• 0xFFF40570	CPRI Error Interrupt Enable Register	CPRInEIER
• 0xFFF40574	Timer Enable Register	CPRInTMRE
• 0xFFF40578	Receive Ethernet Write Pointer Ring	CPRInREWPR
• 0xFFF4057C	Transmit Ethernet Write Pointer Ring	CPRInTEWPR
• 0xFFF40580	Receive HDLC Write Pointer Ring	CPRInRHWPR
• 0xFFF40584	Transmit HDLC Write Pointer Ring	CPRInTHWPR
• 0xFFF405B0–0xFFF4060F	Receive AxCy Parameter Register	CPRInRACPRy
• 0xFFF40640–0xFFF4069F	Transmit AxCy Parameter Register	CPRInTACPRy
• 0xFFF40700-0xFFF4079F	CPRI Auxiliary Interface Mask Registers	CPRInMASKRy
• 0xFFF407A0	CPRI Auxiliary Control Register	CPRInAUXCR
Status Registers		
• 0xFFF40800	Receive IQ Buffer Displacement Register	CPRInRIQBDR
• 0xFFF40804	Receive IQ Buffer Second Destination Displacement Register	CPRInRIQSDBDR
• 0xFFF40808	Transmit IQ Buffer Displacement Register	CPRInTIQBDR
• 0xFFF4080C	Receive Chips Counter Register	CPRInRCCR
• 0xFFF40810	Receive VSS Buffer Displacement Register	CPRInRVSSBDR
• 0xFFF40814	Transmit VSS Buffer Displacement Register	CPRInTVSSBDR
• 0xFFF40818–0xFFF4081F	Receive Ethernet Buffer Descriptor	CPRInRETHBD
• 0xFFF40820-0xFFF40827	Transmit Ethernet Buffer Descriptor	CPRInTETHBD
• 0xFFF40828	Receive Ethernet Read Pointer Ring	CPRInRERPR
• 0xFFF4082C	Transmit Ethernet Read Pointer Ring	CPRInTERPR
• 0xFFF40830–0xFFF40837	Receive HDLC Buffer Descriptor	CPRInRHDLCBD
• 0xFFF40838–0xFFF4083F	Transmit HDLC Buffer Descriptor	CPRInTHDLCBD
• 0xFFF40840	Receive HDLC Read Pointer Ring	CPRInRHRPR
• 0xFFF40844	Transmit HDLC Read Pointer Ring	CPRInTHRPR
• 0xFFF40848	Receive Event Register	CPRInRER
• 0xFFF4084C	Transmit Event Register	CPRInTER
• 0xFFF40850	Error Event Register	CPRInEER

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF40854	Receive Ethernet Coalescing Status	CPRInRETHCS
• 0xFFF40858	Transmit Ethernet Coalescing Status	CPRInTETHCS
• 0xFFF4085C	Timer Status Register.	CPRInTMRSR
• 0xFFF40860	Receive Status Register	CPRInRSR
• 0xFFF40864	Transmit Status Register	CPRInTSR
Configuration Memories		
• 0xFFF40900- 0xFFF417FF	Receive Configuration Memory	RCM
• 0xFFF41800- 0xFFF42700	Transmit Configuration Memory	TCM
CPRI General Registers		
• 0xFFF58000	CPRI Clocks Control Register	CPRICCR
• 0xFFF58010– 0xFFF58050	CPRI Interrupt Control Register n	CPRIICRn
• 0xFFF58050	CPRI Receive CPU Control Interrupt Enable Register	CPRIRCCIER
• 0xFFF58054	CPRI Transmit CPU Control Interrupt Enable Register	CPRITCCIER
• 0xFFF58058	General Receive Synchronization Register	CPRIGRSR
• 0xFFF5805C	General TRansmit Synchronization Register	CPRIGTSR
• 0xFFF58094	CPRI Error Status Register	CPRIESR
• 0xFFF60000– 0xFFF7FFFF	eMSG (see **Chapter 16**, *Serial RapidIO Controller and Enhanced Message Complex*).	
Block 0		
• 0xFFF60000– 0xFFF6003C	Inbound Block 0 Classification Unit 0 The individual message type classification registers use this space.	IB0CU0
• 0xFFF60040– 0xFFF6007C	QMLite Inbound Block 0 Message Queue 0 The inbound message queue registers use this space.	QIB0MQ0
• 0xFFF60080– 0xFFF600BC	reserved	
• 0xFFF600C0– 0xFFF600FC	QMLite Outbound Block 0 Message Queue 0 The outbound message queue registers use this space.	QOB0MQ0
• 0xFFF60100– 0xFFF6013C	Inbound Block 0 Classification Unit 1 The individual message type classification registers use this space.	IB0CU1
• 0xFFF60140– 0xFFF6017C	QMLite Inbound Block 0 Message Queue 1 The inbound message queue registers use this space.	QIB0MQ1
• 0xFFF60180– 0xFFF601BC	reserved	
• 0xFFF601C0– 0xFFF601FC	QMLite Outbound Block 0 Message Queue1 The outbound message queue registers use this space.	QOB0MQ1
• 0xFFF60200– 0xFFF6023C	Inbound Block 0 Classification Unit 2 The individual message type classification registers use this space.	IB0CU2
• 0xFFF60240– 0xFFF6027C	QMLite Inbound Block 0 Message Queue 2 The inbound message queue registers use this space.	QIB0MQ2
• 0xFFF60280– 0xFFF602BC	reserved	
• 0xFFF602C0– 0xFFF602FC	QMLite Outbound Block 0 Message Queue 2 The outbound message queue registers use this space.	QOB0MQ2

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF60300–0xFFF6033C	Inbound Block 0 Classification Unit 3 The individual message type classification registers use this space.	IB0CU3
• 0xFFF60340–0xFFF6037C	QMLite Inbound Block 0 Message Queue 3 The inbound message queue registers use this space.	QIB0MQ3
• 0xFFF60380–0xFFF603BC	reserved	
• 0xFFF603C0–0xFFF603FC	QMLite Outbound Block 0 Message Queue 3 The outbound message queue registers use this space.	QOB0MQ3
• 0xFFF60400–0xFFF6043C	Inbound Block 0 Classification Unit 4 The individual message type classification registers use this space.	IB0CU4
• 0xFFF60440–0xFFF6047C	QMLite Inbound Block 0 Message Queue 4 The inbound message queue registers use this space.	QIB0MQ4
• 0xFFF60480–0xFFF604BC	reserved	
• 0xFFF604C0–0xFFF604FC	QMLite Outbound Block 0 Message Queue 4 The outbound message queue registers use this space.	QOB0MQ4
• 0xFFF60500–0xFFF6053C	Inbound Block 0 Classification Unit 5 The individual message type classification registers use this space.	IB0CU5
• 0xFFF60540–0xFFF6057C	QMLite Inbound Block 0 Message Queue 5 The inbound message queue registers use this space.	QIB0MQ5
• 0xFFF60580–0xFFF605BC	reserved	
• 0xFFF605C0–0xFFF605FC	QMLite Outbound Block 0 Message Queue 5 The outbound message queue registers use this space.	QOB0MQ5
• 0xFFF60600–0xFFF60063C	Inbound Block 0 Classification Unit 6 The individual message type classification registers use this space.	IB0CU6
• 0xFFF60640–0xFFF6067C	QMLite Inbound Block 0 Message Queue 6 The inbound message queue registers use this space.	QIB0MQ6
• 0xFFF60680–0xFFF606BC	reserved	
• 0xFFF606C0–0xFFF606FC	QMLite Outbound Block 0 Message Queue 6 The outbound message queue registers use this space.	QOB0MQ6
• 0xFFF60700–0xFFF6073C	Inbound Block 0 Classification Unit 7 The individual message type classification registers use this space.	IB0CU7
• 0xFFF60740–0xFFF6077C	QMLite Inbound Block 0 Message Queue 7 The inbound message queue registers use this space.	QIB0MQ7
• 0xFFF60780–0xFFF607BC	reserved	
• 0xFFF607C0–0xFFF607FC	QMLite Outbound Block 0 Message Queue 7 The outbound message queue registers use this space.	QOB0MQ7
• 0xFFF60800–0xFFF608FC	QMLite Outbound Block 0 Completion Queue The outbound completion queue registers use this space.	QOB0CQ
• 0xFFF60900–0xFFF609FC	BMLite Portal 0 The portal registers use this space.	BP0
• 0xFFF60A00–0xFFF60AFC	**QMLite Global Registers**	
• 0xFFF60A00	Message Queue Mode Register	MQMR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF60A04–0xFFF60A07	reserved	
• 0xFFF60A08	Outbound Message Queue Dequeue Scheduler Configuration Register 1	OMQDSCR1
• 0xFFF60A0C	Outbound Message Queue Dequeue Scheduler Configuration Register 2	OMQDSCR2
• 0xFFF60A10	Message Queue Interrupt Enable Register	MQIER
• 0xFFF60A14	Message Queue Error Detect Register	MQEDR
• 0xFFF60A18–0xFFF60A1B	reserved	
• 0xFFF60A1C	Message Queue Error Capture Address Register	MQECAR
• 0xFFF60A20–0xFFF60AFF	reserved	
• 0xFFF60B00–0xFFF60BFC	**eMSG Message Unit Block Revision Registers**	
• 0xFFF60B00–0xFFF60Bf7	reserved	
• 0xFFF60BF8	IP Block Revision Register 0	IPBRR0
• 0xFFF60BFC	IP Block Revision Register 1	IPBRR1
• 0xFFF60C00–0xFFF60EFC	**BMLite Global Registers**	
• 0xFFF60C00	Pool 0 Software Portal Depletion Entry Threshold Register	POOL0_SWDET
• 0xFFF60C04	Pool 0 Software Portal Depletion Exit Threshold Register	POOL0_SWDXT
• 0xFFF60C08	Pool 0 Software Portal Depletion Count Register	POOL0_SDCNT
• 0xFFF60C0C	Pool 0 Content Register	POOL0_CONTENT
• 0xFFF60C10	Pool 0 Hardware Portal Depletion Entry Threshold Register	POOL0_HWDET
• 0xFFF60C14	Pool 0 Hardware Portal Depletion Exit Threshold Register	POOL0_HWDXT
• 0xFFF60C18	Pool 0 Hardware Portal Depletion Count Register	POOL0_HDCNT
• 0xFFF60C1C–0xFFF60C1F	reserved	
• 0xFFF60C20–0xFFF60C3F	Pool 1 Registers (same registers as for Pool 0)	
• 0xFFF60C40–0xFFF60C5F	Pool 2 Registers (same registers as for Pool 0)	
• 0xFFF60C60–0xFFF60C7F	Pool 3 Registers (same registers as for Pool 0)	
• 0xFFF60C80–0xFFF60C9F	Pool 4 Registers (same registers as for Pool 0)	
• 0xFFF60CA0–0xFFF60CBF	Pool 5 Registers (same registers as for Pool 0)	
• 0xFFF60CC0–0xFFF60CDF	Pool 6 Registers (same registers as for Pool 0)	
• 0xFFF60CE0–0xFFF60CFF	Pool 7 Registers (same registers as for Pool 0)	
• 0xFFF60D00–0xFFF60D1F	Pool 8 Registers (same registers as for Pool 0)	
• 0xFFF60D20–0xFFF60D3F	Pool 9 Registers (same registers as for Pool 0)	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF60D40–0xFFF60D5F	Pool 10 Registers (same registers as for Pool 0)	
• 0xFFF60D60–0xFFF60D7F	Pool 11 Registers (same registers as for Pool 0)	
• 0xFFF60D80–0xFFF60D9F	Pool 12 Registers (same registers as for Pool 0)	
• 0xFFF60DA0–0xFFF60DBF	Pool 13 Registers (same registers as for Pool 0)	
• 0xFFF60DC0–0xFFF60DDF	Pool 14 Registers (same registers as for Pool 0)	
• 0xFFF60DE0–0xFFF60DFF	Pool 15 Registers (same registers as for Pool 0)	
• 0xFFF60E00–0xFFF60E8B	reserved	
• 0xFFF60E8C	AXI Configuration Register 1	AXI_CFG_1
• 0xFFF60E90	AXI Configuration Register 2	AXI_CFG_2
• 0xFFF60E94	Buffer Pointer Release Range Configuration Register	BPRR_CFG
• 0xFFF60E98	Buffer Pointer Release Range Address Register	BPRR_START
• 0xFFF60E9C	Buffer Pointer Release Range Stop Address Register	BPRR_END
• 0xFFF60EA0	FBPR Free Pool Count Register	FBPR_FPC
• 0xFFF60EA4–0xFFF60EA7	reserved	
• 0xFFF60EA8	FBPR Free Pool Depletion Interrupt Threshold Register	FBPR_FP_THRES
• 0xFFF60EAC	Dynamic Power Management Configuration Register	DPM_CFG
• 0xFFF60EB0	Error Interrupt Status Register	ERR_ISR
• 0xFFF60EB4	Error Interrupt Enable Register	ERR_IER
• 0xFFF60EB8	Error Interrupt Status Disable Register	ERR_ISDR
• 0xFFF60EBC	Error Interrupt Inhibit Register	ERR_IIR
• 0xFFF60EC0	Error Interrupt Force Register	ERR_IFR
• 0xFFF60EC4–0xFFF60ECF	reserved	
• 0xFFF60ED0	Single Bit ECC Error Threshold Register	SBET
• 0xFFF60ED4	Single Bit ECC Error Count Register	SBEC
• 0xFFF60ED8–0xFFF60EDF	reserved	
• 0xFFF60EE0	External Memory Access Error Capture Register	EMAI_ECR
• 0xFFF60EE4	External Memory Access Error Address Register	EMAI_EADR
• 0xFFF60EE8–0xFFF60EEF	reserved	
• 0xFFF60EF0	External Memory Corruption Error Caption Register	EMAI_ECR
• 0xFFF60EF4	External Memory Corruption Error Address Register	EMAI_EADR
• 0xFFF60EF8	IP Block Revision Register	IP_REV_1
• 0xFFF60EFC	IP Block Revision Register	IP_REV_2
• 0xFFF60F00–0xFFF60FFC	**eMSG Message Unit Global Registers**	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF60F00	Message Unit Mode Register	MUMR
• 0xFFF60F04	Message Unit Status Register	MUSR
• 0xFFF60F08–0xFFF60F1F	reserved	
• 0xFFF60F20	Message Unit Interrupt Enable Register	MUIER
• 0xFFF60F24	Message Unit Error Detect Register	MUEDR
• 0xFFF60F28	Message Unit Interrupt Coalescing Register	MUICR
• 0xFFF60F2C	Message Unit T8 Drop Counter Register	MUT8DCR
• 0xFFF60F30	Message Unit T9 Drop Counter Register	MUT9DCR
• 0xFFF60F34	Message Unit Error Capture MQ Register	MUECMQR
• 0xFFF60F38	Message Unit Error Capture CD Register 0	MUECCDR0
• 0xFFF60F3C	Message Unit Error Capture CD Register 1	MUECCDR1
• 0xFFF60F40	Message Unit Error Capture CD Register 2	MUECCDR2
• 0xFFF60F44	Message Unit Error Capture CD Register 3	MUECCDR3
• 0xFFF60F48–0xFFF60F4B	reserved	
• 0xFFF60F4C	Message Unit Error Capture Address Register	MUECAR
• 0xFFF60F50	Message Unit Arbitration Weight Register	MUAWR
• 0xFFF60F54	Message Unit Outbound Interleaving Mask Register	MUOIMR
• 0xFFF60F58–0xFFF60F6F	reserved	
• 0xFFF60F70	Message Unit Segmentation Execution Privilege Register 0	MUSEPR0
• 0xFFF60F74	Message Unit Segmentation Execution Privilege Register 1	MUSEPR1
• 0xFFF60F78–0xFFF60F7F	reserved	
• 0xFFF60F80	Message Unit Reassembly Context Assignment Register 0	MURCAR0
• 0xFFF60F84	Message Unit Reassembly Context Assignment Register 1	MURCAR1
• 0xFFF60F88	Message Unit Reassembly Context Assignment Register 2	MURCAR2
• 0xFFF60F8C–0xFFF60FFF	reserved	
Block 1		
• 0xFFF61000–0xFFF6103C	Inbound Block 1 Classification Unit 0 The individual message type classification registers use this space.	IB1CU0
• 0xFFF61040–0xFFF6107C	QMLite Inbound Block 1 Message Queue 0 The inbound message queue registers use this space.	QIB1MQ0
• 0xFFF61080–0xFFF610BC	reserved	
• 0xFFF610C0–0xFFF610FC	QMLite Outbound Block 1 Message Queue 0 The outbound message queue registers use this space.	QOB1MQ0
• 0xFFF61100–0xFFF6113C	Inbound Block 1 Classification Unit 1 The individual message type classification registers use this space.	IB1CU1
• 0xFFF61140–0xFFF6117C	QMLite Inbound Block 1 Message Queue 1 The inbound message queue registers use this space.	QIB1MQ1
• 0xFFF61180–0xFFF611BC	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF611C0–0xFFF611FC	QMLite Outbound Block 1 Message Queue1 The outbound message queue registers use this space.	QOB1MQ1
• 0xFFF61200–0xFFF6123C	Inbound Block 1 Classification Unit 2 The individual message type classification registers use this space.	IB1CU2
• 0xFFF61240–0xFFF6127C	QMLite Inbound Block 1 Message Queue 2 The inbound message queue registers use this space.	QIB1MQ2
• 0xFFF61280–0xFFF612BC	reserved	
• 0xFFF612C0–0xFFF612FC	QMLite Outbound Block 1 Message Queue 2 The outbound message queue registers use this space.	QOB1MQ2
• 0xFFF61300–0xFFF6133C	Inbound Block 1 Classification Unit 3 The individual message type classification registers use this space.	IB1CU3
• 0xFFF61340–0xFFF6137C	QMLite Inbound Block 1 Message Queue 3 The inbound message queue registers use this space.	QIB1MQ3
• 0xFFF61380–0xFFF613BC	reserved	
• 0xFFF613C0–0xFFF613FC	QMLite Outbound Block 1 Message Queue 3 The outbound message queue registers use this space.	QOB1MQ3
• 0xFFF61400–0xFFF6143C	Inbound Block 1 Classification Unit 4 The individual message type classification registers use this space.	IB1CU4
• 0xFFF61440–0xFFF6147C	QMLite Inbound Block 1 Message Queue 4 The inbound message queue registers use this space.	QIB1MQ4
• 0xFFF61480–0xFFF614BC	reserved	
• 0xFFF614C0–0xFFF614FC	QMLite Outbound Block 1 Message Queue 4 The outbound message queue registers use this space.	QOB1MQ4
• 0xFFF61500–0xFFF6153C	Inbound Block 1 Classification Unit 5 The individual message type classification registers use this space.	IB1CU5
• 0xFFF61540–0xFFF6157C	QMLite Inbound Block 1 Message Queue 5 The inbound message queue registers use this space.	QIB1MQ5
• 0xFFF61580–0xFFF615BC	reserved	
• 0xFFF615C0–0xFFF615FC	QMLite Outbound Block 1 Message Queue 5 The outbound message queue registers use this space.	QOB1MQ5
• 0xFFF61600–0xFFF61063C	Inbound Block 1 Classification Unit 6 The individual message type classification registers use this space.	IB1CU6
• 0xFFF61640–0xFFF6167C	QMLite Inbound Block 1 Message Queue 6 The inbound message queue registers use this space.	QIB1MQ6
• 0xFFF61680–0xFFF616BC	reserved	
• 0xFFF616C0–0xFFF616FC	QMLite Outbound Block 1 Message Queue 6 The outbound message queue registers use this space.	QOB1MQ6
• 0xFFF61700–0xFFF6173C	Inbound Block 1 Classification Unit 7 The individual message type classification registers use this space.	IB1CU7
• 0xFFF61740–0xFFF6177C	QMLite Inbound Block 1 Message Queue 7 The inbound message queue registers use this space.	QIB1MQ7
• 0xFFF61780–0xFFF617BC	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF617C0–0xFFF617FC	QMLite Outbound Block 1 Message Queue 7 The outbound message queue registers use this space.	QOB1MQ7
• 0xFFF61800–0xFFF618FC	QMLite Outbound Block 1 Completion Queue The outbound completion queue registers use this space.	QOB1CQ
• 0xFFF61900–0xFFF619FC	BMLite Portal 1 The portal registers use this space.	BP1
• 0xFFF61A00–0xFFF61FFC	reserved	
Block 2		
• 0xFFF62000–0xFFF6203C	Inbound Block 2 Classification Unit 0 The individual message type classification registers use this space.	IB2CU0
• 0xFFF62040–0xFFF6207C	QMLite Inbound Block 2 Message Queue 0 The inbound message queue registers use this space.	QIB2MQ0
• 0xFFF62080–0xFFF620BC	reserved	
• 0xFFF620C0–0xFFF620FC	QMLite Outbound Block 2 Message Queue 0 The outbound message queue registers use this space.	QOB2MQ0
• 0xFFF62100–0xFFF6213C	Inbound Block 2 Classification Unit 1 The individual message type classification registers use this space.	IB2CU1
• 0xFFF62140–0xFFF6217C	QMLite Inbound Block 2 Message Queue 1 The inbound message queue registers use this space.	QIB2MQ1
• 0xFFF62180–0xFFF621BC	reserved	
• 0xFFF621C0–0xFFF621FC	QMLite Outbound Block 2 Message Queue1 The outbound message queue registers use this space.	QOB2MQ1
• 0xFFF62200–0xFFF6223C	Inbound Block 2 Classification Unit 2 The individual message type classification registers use this space.	IB2CU2
• 0xFFF62240–0xFFF6227C	QMLite Inbound Block 2 Message Queue 2 The inbound message queue registers use this space.	QIB2MQ2
• 0xFFF62280–0xFFF622BC	reserved	
• 0xFFF622C0–0xFFF622FC	QMLite Outbound Block 2 Message Queue 2 The outbound message queue registers use this space.	QOB2MQ2
• 0xFFF62300–0xFFF6233C	Inbound Block 2 Classification Unit 3 The individual message type classification registers use this space.	IB2CU3
• 0xFFF62340–0xFFF6237C	QMLite Inbound Block 2 Message Queue 3 The inbound message queue registers use this space.	QIB2MQ3
• 0xFFF62380–0xFFF623BC	reserved	
• 0xFFF623C0–0xFFF623FC	QMLite Outbound Block 2 Message Queue 3 The outbound message queue registers use this space.	QOB2MQ3
• 0xFFF62400–0xFFF6243C	Inbound Block 2 Classification Unit 4 The individual message type classification registers use this space.	IB2CU4
• 0xFFF62440–0xFFF6247C	QMLite Inbound Block 2 Message Queue 4 The inbound message queue registers use this space.	QIB2MQ4
• 0xFFF62480–0xFFF624BC	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF624C0–0xFFF624FC	QMLite Outbound Block 2 Message Queue 4 The outbound message queue registers use this space.	QOB2MQ4
• 0xFFF62500–0xFFF6253C	Inbound Block 2 Classification Unit 5 The individual message type classification registers use this space.	IB2CU5
• 0xFFF62540–0xFFF6257C	QMLite Inbound Block 2 Message Queue 5 The inbound message queue registers use this space.	QIB2MQ5
• 0xFFF62580–0xFFF625BC	reserved	
• 0xFFF625C0–0xFFF625FC	QMLite Outbound Block 2 Message Queue 5 The outbound message queue registers use this space.	QOB2MQ5
• 0xFFF62600–0xFFF62063C	Inbound Block 2 Classification Unit 6 The individual message type classification registers use this space.	IB2CU6
• 0xFFF62640–0xFFF6267C	QMLite Inbound Block 2 Message Queue 6 The inbound message queue registers use this space.	QIB2MQ6
• 0xFFF62680–0xFFF626BC	reserved	
• 0xFFF626C0–0xFFF626FC	QMLite Outbound Block 2 Message Queue 6 The outbound message queue registers use this space.	QOB2MQ6
• 0xFFF62700–0xFFF6273C	Inbound Block 2 Classification Unit 7 The individual message type classification registers use this space.	IB2CU7
• 0xFFF62740–0xFFF6277C	QMLite Inbound Block 2 Message Queue 7 The inbound message queue registers use this space.	QIB2MQ7
• 0xFFF62780–0xFFF627BC	reserved	
• 0xFFF627C0–0xFFF627FC	QMLite Outbound Block 2 Message Queue 7 The outbound message queue registers use this space.	QOB2MQ7
• 0xFFF62800–0xFFF628FC	QMLite Outbound Block 2 Completion Queue The outbound completion queue registers use this space.	QOB2CQ
• 0xFFF62900–0xFFF629FC	BMLite Portal 2 The portal registers use this space.	BP1
• 0xFFF62A00–0xFFF62FFC	reserved	
Block 3		
• 0xFFF63000–0xFFF6303C	Inbound Block 3 Classification Unit 0 The individual message type classification registers use this space.	IB3CU0
• 0xFFF63040–0xFFF6307C	QMLite Inbound Block 3 Message Queue 0 The inbound message queue registers use this space.	QIB3MQ0
• 0xFFF63080–0xFFF630BC	reserved	
• 0xFFF630C0–0xFFF630FC	QMLite Outbound Block 3 Message Queue 0 The outbound message queue registers use this space.	QOB3MQ0
• 0xFFF63100–0xFFF6313C	Inbound Block 3 Classification Unit 1 The individual message type classification registers use this space.	IB3CU1
• 0xFFF63140–0xFFF6317C	QMLite Inbound Block 3 Message Queue 1 The inbound message queue registers use this space.	QIB3MQ1
• 0xFFF63180–0xFFF631BC	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF631C0–0xFFF631FC	QMLite Outbound Block 3 Message Queue1 The outbound message queue registers use this space.	QOB3MQ1
• 0xFFF63200–0xFFF6323C	Inbound Block 3 Classification Unit 2 The individual message type classification registers use this space.	IB3CU2
• 0xFFF63240–0xFFF6327C	QMLite Inbound Block 3 Message Queue 2 The inbound message queue registers use this space.	QIB3MQ2
• 0xFFF63280–0xFFF632BC	reserved	
• 0xFFF632C0–0xFFF632FC	QMLite Outbound Block 3 Message Queue 2 The outbound message queue registers use this space.	QOB3MQ2
• 0xFFF63300–0xFFF6333C	Inbound Block 3 Classification Unit 3 The individual message type classification registers use this space.	IB3CU3
• 0xFFF63340–0xFFF6337C	QMLite Inbound Block 3 Message Queue 3 The inbound message queue registers use this space.	QIB3MQ3
• 0xFFF63380–0xFFF633BC	reserved	
• 0xFFF633C0–0xFFF633FC	QMLite Outbound Block 3 Message Queue 3 The outbound message queue registers use this space.	QOB3MQ3
• 0xFFF63400–0xFFF6343C	Inbound Block 3 Classification Unit 4 The individual message type classification registers use this space.	IB3CU4
• 0xFFF63440–0xFFF6347C	QMLite Inbound Block 3 Message Queue 4 The inbound message queue registers use this space.	QIB3MQ4
• 0xFFF63480–0xFFF634BC	reserved	
• 0xFFF634C0–0xFFF634FC	QMLite Outbound Block 3 Message Queue 4 The outbound message queue registers use this space.	QOB3MQ4
• 0xFFF63500–0xFFF6353C	Inbound Block 3 Classification Unit 5 The individual message type classification registers use this space.	IB3CU5
• 0xFFF63540–0xFFF6357C	QMLite Inbound Block 3 Message Queue 5 The inbound message queue registers use this space.	QIB3MQ5
• 0xFFF63580–0xFFF635BC	reserved	
• 0xFFF635C0–0xFFF635FC	QMLite Outbound Block 3 Message Queue 5 The outbound message queue registers use this space.	QOB3MQ5
• 0xFFF63600–0xFFF63063C	Inbound Block 3 Classification Unit 6 The individual message type classification registers use this space.	IB3CU6
• 0xFFF63640–0xFFF6367C	QMLite Inbound Block 3 Message Queue 6 The inbound message queue registers use this space.	QIB3MQ6
• 0xFFF63680–0xFFF636BC	reserved	
• 0xFFF636C0–0xFFF636FC	QMLite Outbound Block 3 Message Queue 6 The outbound message queue registers use this space.	QOB3MQ6
• 0xFFF63700–0xFFF6373C	Inbound Block 3 Classification Unit 7 The individual message type classification registers use this space.	IB3CU7
• 0xFFF63740–0xFFF6377C	QMLite Inbound Block 3 Message Queue 7 The inbound message queue registers use this space.	QIB3MQ7
• 0xFFF63780–0xFFF637BC	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF637C0–0xFFF637FC	QMLite Outbound Block 3 Message Queue 7 The outbound message queue registers use this space.	QOB3MQ7
• 0xFFF63800–0xFFF638FC	QMLite Outbound Block 3 Completion Queue The outbound completion queue registers use this space.	QOB3CQ
• 0xFFF63900–0xFFF639FC	BMLite Portal 3 The portal registers use this space.	BP1
• 0xFFF63A00–0xFFF63FFC	reserved	
Block 4		
• 0xFFF64000–0xFFF6403C	Inbound Block 4 Classification Unit 0 The individual message type classification registers use this space.	IB4CU0
• 0xFFF64040–0xFFF6407C	QMLite Inbound Block 4 Message Queue 0 The inbound message queue registers use this space.	QIB4MQ0
• 0xFFF64080–0xFFF640BC	reserved	
• 0xFFF640C0–0xFFF640FC	QMLite Outbound Block 4 Message Queue 0 The outbound message queue registers use this space.	QOB4MQ0
• 0xFFF64100–0xFFF6413C	Inbound Block 4 Classification Unit 1 The individual message type classification registers use this space.	IB4CU1
• 0xFFF64140–0xFFF6417C	QMLite Inbound Block 4 Message Queue 1 The inbound message queue registers use this space.	QIB4MQ1
• 0xFFF64180–0xFFF641BC	reserved	
• 0xFFF641C0–0xFFF641FC	QMLite Outbound Block 4 Message Queue1 The outbound message queue registers use this space.	QOB4MQ1
• 0xFFF64200–0xFFF6423C	Inbound Block 4 Classification Unit 2 The individual message type classification registers use this space.	IB4CU2
• 0xFFF64240–0xFFF6427C	QMLite Inbound Block 4 Message Queue 2 The inbound message queue registers use this space.	QIB4MQ2
• 0xFFF64280–0xFFF642BC	reserved	
• 0xFFF642C0–0xFFF642FC	QMLite Outbound Block 4 Message Queue 2 The outbound message queue registers use this space.	QOB4MQ2
• 0xFFF64300–0xFFF6433C	Inbound Block 4 Classification Unit 3 The individual message type classification registers use this space.	IB4CU3
• 0xFFF64340–0xFFF6437C	QMLite Inbound Block 4 Message Queue 3 The inbound message queue registers use this space.	QIB4MQ3
• 0xFFF64380–0xFFF643BC	reserved	
• 0xFFF643C0–0xFFF643FC	QMLite Outbound Block 4 Message Queue 3 The outbound message queue registers use this space.	QOB4MQ3
• 0xFFF64400–0xFFF6443C	Inbound Block 4 Classification Unit 4 The individual message type classification registers use this space.	IB4CU4
• 0xFFF64440–0xFFF6447C	QMLite Inbound Block 4 Message Queue 4 The inbound message queue registers use this space.	QIB4MQ4
• 0xFFF64480–0xFFF644BC	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF644C0–0xFFF644FC	QMLite Outbound Block 4 Message Queue 4 The outbound message queue registers use this space.	QOB4MQ4
• 0xFFF64500–0xFFF6453C	Inbound Block 4 Classification Unit 5 The individual message type classification registers use this space.	IB4CU5
• 0xFFF64540–0xFFF6457C	QMLite Inbound Block 4 Message Queue 5 The inbound message queue registers use this space.	QIB4MQ5
• 0xFFF64580–0xFFF645BC	reserved	
• 0xFFF645C0–0xFFF645FC	QMLite Outbound Block 4 Message Queue 5 The outbound message queue registers use this space.	QOB4MQ5
• 0xFFF64600–0xFFF64063C	Inbound Block 4 Classification Unit 6 The individual message type classification registers use this space.	IB4CU6
• 0xFFF64640–0xFFF6467C	QMLite Inbound Block 4 Message Queue 6 The inbound message queue registers use this space.	QIB4MQ6
• 0xFFF64680–0xFFF646BC	reserved	
• 0xFFF646C0–0xFFF646FC	QMLite Outbound Block 4 Message Queue 6 The outbound message queue registers use this space.	QOB4MQ6
• 0xFFF64700–0xFFF6473C	Inbound Block 4 Classification Unit 7 The individual message type classification registers use this space.	IB4CU7
• 0xFFF64740–0xFFF6477C	QMLite Inbound Block 4 Message Queue 7 The inbound message queue registers use this space.	QIB4MQ7
• 0xFFF64780–0xFFF647BC	reserved	
• 0xFFF647C0–0xFFF647FC	QMLite Outbound Block 4 Message Queue 7 The outbound message queue registers use this space.	QOB4MQ7
• 0xFFF64800–0xFFF648FC	QMLite Outbound Block 4 Completion Queue The outbound completion queue registers use this space.	QOB4CQ
• 0xFFF64900–0xFFF649FC	BMLite Portal 4 The portal registers use this space.	BP1
• 0xFFF64A00–0xFFF64FFC	reserved	
Block 5		
• 0xFFF65000–0xFFF6503C	Inbound Block 5 Classification Unit 0 The individual message type classification registers use this space.	IB5CU0
• 0xFFF65040–0xFFF6507C	QMLite Inbound Block 5 Message Queue 0 The inbound message queue registers use this space.	QIB5MQ0
• 0xFFF65080–0xFFF650BC	reserved	
• 0xFFF650C0–0xFFF650FC	QMLite Outbound Block 5 Message Queue 0 The outbound message queue registers use this space.	QOB5MQ0
• 0xFFF65100–0xFFF6513C	Inbound Block 5 Classification Unit 1 The individual message type classification registers use this space.	IB5CU1
• 0xFFF65140–0xFFF6517C	QMLite Inbound Block 5 Message Queue 1 The inbound message queue registers use this space.	QIB5MQ1
• 0xFFF65180–0xFFF651BC	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF651C0–0xFFF651FC	QMLite Outbound Block 5 Message Queue1 The outbound message queue registers use this space.	QOB5MQ1
• 0xFFF65200–0xFFF6523C	Inbound Block 5 Classification Unit 2 The individual message type classification registers use this space.	IB5CU2
• 0xFFF65240–0xFFF6527C	QMLite Inbound Block 5 Message Queue 2 The inbound message queue registers use this space.	QIB5MQ2
• 0xFFF65280–0xFFF652BC	reserved	
• 0xFFF652C0–0xFFF652FC	QMLite Outbound Block 5 Message Queue 2 The outbound message queue registers use this space.	QOB5MQ2
• 0xFFF65300–0xFFF6533C	Inbound Block 5 Classification Unit 3 The individual message type classification registers use this space.	IB5CU3
• 0xFFF65340–0xFFF6537C	QMLite Inbound Block 5 Message Queue 3 The inbound message queue registers use this space.	QIB5MQ3
• 0xFFF65380–0xFFF653BC	reserved	
• 0xFFF653C0–0xFFF653FC	QMLite Outbound Block 5 Message Queue 3 The outbound message queue registers use this space.	QOB5MQ3
• 0xFFF65400–0xFFF6543C	Inbound Block 5 Classification Unit 4 The individual message type classification registers use this space.	IB5CU4
• 0xFFF65440–0xFFF6547C	QMLite Inbound Block 5 Message Queue 4 The inbound message queue registers use this space.	QIB5MQ4
• 0xFFF65480–0xFFF654BC	reserved	
• 0xFFF654C0–0xFFF654FC	QMLite Outbound Block 5 Message Queue 4 The outbound message queue registers use this space.	QOB5MQ4
• 0xFFF65500–0xFFF6553C	Inbound Block 5 Classification Unit 5 The individual message type classification registers use this space.	IB5CU5
• 0xFFF65540–0xFFF6557C	QMLite Inbound Block 5 Message Queue 5 The outbound message queue registers use this space.	QIB5MQ5
• 0xFFF65580–0xFFF655BC	reserved	
• 0xFFF655C0–0xFFF655FC	QMLite Outbound Block 5 Message Queue 5 The outbound message queue registers use this space.	QOB5MQ5
• 0xFFF65600–0xFFF65063C	Inbound Block 5 Classification Unit 6 The individual message type classification registers use this space.	IB5CU6
• 0xFFF65640–0xFFF6567C	QMLite Inbound Block 5 Message Queue 6 The inbound message queue registers use this space.	QIB5MQ6
• 0xFFF65680–0xFFF656BC	reserved	
• 0xFFF656C0–0xFFF656FC	QMLite Outbound Block 5 Message Queue 6 The outbound message queue registers use this space.	QOB5MQ6
• 0xFFF65700–0xFFF6573C	Inbound Block 5 Classification Unit 7 The individual message type classification registers use this space.	IB5CU7
• 0xFFF65740–0xFFF6577C	QMLite Inbound Block 5 Message Queue 7 The inbound message queue registers use this space.	QIB5MQ7
• 0xFFF65780–0xFFF657BC	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF657C0–0xFFF657FC	QMLite Outbound Block 5 Message Queue 7 The outbound message queue registers use this space.	QOB5MQ7
• 0xFFF65800–0xFFF658FC	QMLite Outbound Block 5 Completion Queue The outbound completion queue registers use this space.	QOB5CQ
• 0xFFF65900–0xFFF659FC	BMLite Portal 5 The portal registers use this space.	BP1
• 0xFFF65A00–0xFFF65FFC	reserved	
Block 6		
• 0xFFF66000–0xFFF6603C	Inbound Block 6 Classification Unit 0 The individual message type classification registers use this space.	IB6CU0
• 0xFFF66040–0xFFF6607C	QMLite Inbound Block 6 Message Queue 0 The inbound message queue registers use this space.	QIB6MQ0
• 0xFFF66080–0xFFF660BC	reserved	
• 0xFFF660C0–0xFFF660FC	QMLite Outbound Block 6 Message Queue 0 The outbound message queue registers use this space.	QOB6MQ0
• 0xFFF66100–0xFFF6613C	Inbound Block 6 Classification Unit 1 The individual message type classification registers use this space.	IB6CU1
• 0xFFF66140–0xFFF6617C	QMLite Inbound Block 6 Message Queue 1 The inbound message queue registers use this space.	QIB6MQ1
• 0xFFF66180–0xFFF661BC	reserved	
• 0xFFF661C0–0xFFF661FC	QMLite Outbound Block 6 Message Queue1 The outbound message queue registers use this space.	QOB6MQ1
• 0xFFF66200–0xFFF6623C	Inbound Block 6 Classification Unit 2 The individual message type classification registers use this space.	IB6CU2
• 0xFFF66240–0xFFF6627C	QMLite Inbound Block 6 Message Queue 2 The inbound message queue registers use this space.	QIB6MQ2
• 0xFFF66280–0xFFF662BC	reserved	
• 0xFFF662C0–0xFFF662FC	QMLite Outbound Block 6 Message Queue 2 The outbound message queue registers use this space.	QOB6MQ2
• 0xFFF66300–0xFFF6633C	Inbound Block 6 Classification Unit 3 The individual message type classification registers use this space.	IB6CU3
• 0xFFF66340–0xFFF6637C	QMLite Inbound Block 6 Message Queue 3 The inbound message queue registers use this space.	QIB6MQ3
• 0xFFF66380–0xFFF663BC	reserved	
• 0xFFF663C0–0xFFF663FC	QMLite Outbound Block 6 Message Queue 3 The outbound message queue registers use this space.	QOB6MQ3
• 0xFFF66400–0xFFF6643C	Inbound Block 6 Classification Unit 4 The individual message type classification registers use this space.	IB6CU4
• 0xFFF66440–0xFFF6647C	QMLite Inbound Block 6 Message Queue 4 The inbound message queue registers use this space.	QIB6MQ4
• 0xFFF66480–0xFFF664BC	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF664C0–0xFFF664FC	QMLite Outbound Block 6 Message Queue 4 The outbound message queue registers use this space.	QOB6MQ4
• 0xFFF66500–0xFFF6653C	Inbound Block 6 Classification Unit 5 The individual message type classification registers use this space.	IB6CU5
• 0xFFF66540–0xFFF6657C	QMLite Inbound Block 6 Message Queue 5 The inbound message queue registers use this space.	QIB6MQ5
• 0xFFF66580–0xFFF665BC	reserved	
• 0xFFF665C0–0xFFF665FC	QMLite Outbound Block 6 Message Queue 5 The outbound message queue registers use this space.	QOB6MQ5
• 0xFFF66600–0xFFF66063C	Inbound Block 6 Classification Unit 6 The individual message type classification registers use this space.	IB6CU6
• 0xFFF66640–0xFFF6667C	QMLite Inbound Block 6 Message Queue 6 The inbound message queue registers use this space.	QIB6MQ6
• 0xFFF66680–0xFFF666BC	reserved	
• 0xFFF666C0–0xFFF666FC	QMLite Outbound Block 6 Message Queue 6 The outbound message queue registers use this space.	QOB6MQ6
• 0xFFF66700–0xFFF6673C	Inbound Block 6 Classification Unit 7 The individual message type classification registers use this space.	IB6CU7
• 0xFFF66740–0xFFF6677C	QMLite Inbound Block 6 Message Queue 7 The inbound message queue registers use this space.	QIB6MQ7
• 0xFFF66780–0xFFF667BC	reserved	
• 0xFFF667C0–0xFFF667FC	QMLite Outbound Block 6 Message Queue 7 The outbound message queue registers use this space.	QOB6MQ7
• 0xFFF66800–0xFFF668FC	QMLite Outbound Block 6 Completion Queue The outbound completion queue registers use this space.	QOB6CQ
• 0xFFF66900–0xFFF669FC	BMLite Portal 6 The portal registers use this space.	BP1
• 0xFFF66A00–0xFFF66FFC	reserved	
Block 7		
• 0xFFF67000–0xFFF6703C	Inbound Block 7 Classification Unit 0 The individual message type classification registers use this space.	IB7CU0
• 0xFFF67040–0xFFF6707C	QMLite Inbound Block 7 Message Queue 0 The inbound message queue registers use this space.	QIB7MQ0
• 0xFFF67080–0xFFF670BC	reserved	
• 0xFFF670C0–0xFFF670FC	QMLite Outbound Block 7 Message Queue 0 The outbound message queue registers use this space.	QOB7MQ0
• 0xFFF67100–0xFFF6713C	Inbound Block 7 Classification Unit 1 The individual message type classification registers use this space.	IB7CU1
• 0xFFF67140–0xFFF6717C	QMLite Inbound Block 7 Message Queue 1 The inbound message queue registers use this space.	QIB7MQ1
• 0xFFF67180–0xFFF671BC	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF671C0–0xFFF671FC	QMLite Outbound Block 7 Message Queue1 The outbound message queue registers use this space.	QOB7MQ1
• 0xFFF67200–0xFFF6723C	Inbound Block 7 Classification Unit 2 The individual message type classification registers use this space.	IB7CU2
• 0xFFF67240–0xFFF6727C	QMLite Inbound Block 7 Message Queue 2 The inbound message queue registers use this space.	QIB7MQ2
• 0xFFF67280–0xFFF672BC	reserved	
• 0xFFF672C0–0xFFF672FC	QMLite Outbound Block 7 Message Queue 2 The outbound message queue registers use this space.	QOB7MQ2
• 0xFFF67300–0xFFF6733C	Inbound Block 7 Classification Unit 3 The individual message type classification registers use this space.	IB7CU3
• 0xFFF67340–0xFFF6737C	QMLite Inbound Block 7 Message Queue 3 The inbound message queue registers use this space.	QIB7MQ3
• 0xFFF67380–0xFFF673BC	reserved	
• 0xFFF673C0–0xFFF673FC	QMLite Outbound Block 7 Message Queue 3 The outbound message queue registers use this space.	QOB7MQ3
• 0xFFF67400–0xFFF6743C	Inbound Block 7 Classification Unit 4 The individual message type classification registers use this space.	IB7CU4
• 0xFFF67440–0xFFF6747C	QMLite Inbound Block 7 Message Queue 4 The inbound message queue registers use this space.	QIB7MQ4
• 0xFFF67480–0xFFF674BC	reserved	
• 0xFFF674C0–0xFFF674FC	QMLite Outbound Block 7 Message Queue 4 The outbound message queue registers use this space.	QOB7MQ4
• 0xFFF67500–0xFFF6753C	Inbound Block 7 Classification Unit 5 The individual message type classification registers use this space.	IB7CU5
• 0xFFF67540–0xFFF6757C	QMLite Inbound Block 7 Message Queue 5 The inbound message queue registers use this space.	QIB7MQ5
• 0xFFF67580–0xFFF675BC	reserved	
• 0xFFF675C0–0xFFF675FC	QMLite Outbound Block 7 Message Queue 5 The outbound message queue registers use this space.	QOB7MQ5
• 0xFFF67600–0xFFF67063C	Inbound Block 7 Classification Unit 6 The individual message type classification registers use this space.	IB7CU6
• 0xFFF67640–0xFFF6767C	QMLite Inbound Block 7 Message Queue 6 The inbound message queue registers use this space.	QIB7MQ6
• 0xFFF67680–0xFFF676BC	reserved	
• 0xFFF676C0–0xFFF676FC	QMLite Outbound Block 7 Message Queue 6 The outbound message queue registers use this space.	QOB7MQ6
• 0xFFF67700–0xFFF6773C	Inbound Block 7 Classification Unit 7 The individual message type classification registers use this space.	IB7CU7
• 0xFFF67740–0xFFF6777C	QMLite Inbound Block 7 Message Queue 7 The inbound message queue registers use this space.	QIB7MQ7
• 0xFFF67780–0xFFF677BC	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFF677C0–0xFFF677FC	QMLite Outbound Block 7 Message Queue 7 The outbound message queue registers use this space.	QOB7MQ7
• 0xFFF67800–0xFFF678FC	QMLite Outbound Block 7 Completion Queue The outbound completion queue registers use this space.	QOB7CQ
• 0xFFF67900–0xFFF679FC	BMLite Portal 7 The portal registers use this space.	BP1
• 0xFFF67A00–0xFFF7FFFF	reserved	
• 0xFFF80000–0xFFF9FFFF	RapidIO (see **Chapter 16**, *Serial RapidIO Controller and Enhanced Message Complex*)	
– 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
– 0xFFF80020–0xFFF8003B	reserved	
– 0xFFF8003C	Data Streaming Information Capability Register	DSICAR
– 0xFFF80040–0xFFF80047	reserve	
– 0xFFF80048	Data Streaming Logical Layer Control Command and Status Register	DSLLCCSR
– 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 1 Link Maintenance Request Command and Status Register	P1LMREQCSR
– 0xFFF80144	Port 1 Link Maintenance Response Command and Status Register	P1LMRESPCSR
– 0xFFF80148	Port 1 Local ackID Status Command and Status Register	P1LASCSR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF8014C–0xFFF80157	reserved	
– 0xFFF80158	Port 1 Error and Status Command and Status Register	P1ESCSR
– 0xFFF8015C	Port 1 Control Command and Status Register	P1CCSR
– 0xFFF80160	Port 2 Link Maintenance Request Command and Status Register	P2LMREQCSR
– 0xFFF80164	Port 2 Link Maintenance Response Command and Status Register	P2LMRESPCSR
– 0xFFF80168	Port 2 Local ackID Status Command and Status Register	P2LASCSR
– 0xFFF8016C–0xFFF80177	reserved	
– 0xFFF80178	Port 2 Error and Status Command and Status Register	P2ESCSR
– 0xFFF8017C	Port 2 Control Command and Status Register	P2CCSR
– 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 1 Error Detect Command and Status Register	P1EDCSR
– 0xFFF80644	Port 1 Error Rate Enable Command and Status Register	P1ERECSR
– 0xFFF80648	Port 1 Error Capture Attributes Command and Status Register	P1ECACSR
– 0xFFF8064C	Port 1 Packet/Control Symbol Error Capture Command and Status Register 0	P1PCSECCSR0
– 0xFFF80650	Port 1 Packet Error Capture Command and Status Register 1	P1PECCSR1
– 0xFFF80654	Port 1 Packet Error Capture Command and Status Register 2	P1PECCSR2
– 0xFFF80658	Port 1 Packet Error Capture Command and Status Register 3	P1PECCSR3
– 0xFFF8065C–0xFFF80667	reserved	
– 0xFFF80668	Port 1 Error Rate Command and Status Register	P1ERCSR
– 0xFFF8066C	Port 1 Error Rate Threshold Command and Status Register	P1ERTCSR
– 0xFFF80670–0xFFF8067F	reserved	
– 0xFFF80680	Port 2 Error Detect Command and Status Register	P2EDCSR
– 0xFFF80684	Port 2 Error Rate Enable Command and Status Register	P2ERECSR
– 0xFFF80688	Port 2 Error Capture Attributes Command and Status Register	P2ECACSR
– 0xFFF8068C	Port 2 Packet/Control Symbol Error Capture Command and Status Register 0	P2PCSECCSR0

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF80690	Port 2 Packet Error Capture Command and Status Register 1	P2PECCSR1
– 0xFFF80694	Port 2 Packet Error Capture Command and Status Register 2	P2PECCSR2
– 0xFFF80698	Port 2 Packet Error Capture Command and Status Register 3	P2PECCSR3
– 0xFFF8069C–0xFFF806A7	reserved	
– 0xFFF806A8	Port 2 Error Rate Command and Status Register	P2ERCSR
– 0xFFF806AC	Port 2 Error Rate Threshold Command and Status Register	P2ERTCSR
– 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 1 Alternate Device ID Command and Status Register	P1ADIDCSR
– 0xFFF90104–0xFFF9011F	reserved	
– 0xFFF90120	Port 1 Pass-Through Accept-All Configuration Register	P1PTAACR
– 0xFFF90124	Port 1 Logical Outbound Packet Time-to-Live Configuration Register	P1LOPTTLCR
– 0xFFF90128–0xFFF9012F	reserved	
– 0xFFF90130	Port 1 Implementation Error Command and Status Register	P1IECSR
– 0xFFF90134–0xFFF9013F	reserved	
– 0xFFF90140	Port 1 Physical Configuration Register	P1PCR
– 0xFFF90144–0xFFF90157	reserved	
– 0xFFF90158	Port 1 Serial Link Command And Status Register	P1SLCSR
– 0xFFF9015C–0xFFF9015F	reserved	
– 0xFFF90160	Port 1 Serial Link Error Injection Configuration Register	P1SLEICR
– 0xFFF90164	Port 1 Arbitration 0 Tx Configuration Register	P1A0TxCR
– 0xFFF90168	Port 1 Arbitration 1 Tx Configuration Register	P1A1TxCR
– 0xFFF9016C	Port 1 Arbitration 2 Tx Configuration Register	P1A2TxCR
– 0xFFF90170	Port 1 Message Request Tx Buffer Allocation Configuration Register 0	P1MReqTxBACR0
– 0xFFF90174	Port 1 Message Request Tx Buffer Allocation Configuration Register 1	P1MReqTxBACR1
– 0xFFF90178	Port 1 Message Request Tx Buffer Allocation Configuration Register 2	P1MReqTxBACR2

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF9017C	Port 1 Message Response/Flow Control Tx Buffer Allocation Configuration Register	P1MRspFcTxBACR
– 0xFFF90180–0xFFF9017F	reserved	
– 0xFFF90180	Port 2 Alternate Device ID Command and Status Register	P2ADIDCSR
– 0xFFF90184–0xFFF9019F	reserved	
– 0xFFF901A0	Port 2 Pass-Through Accept-All Configuration Register	P2PTAACR
– 0xFFF901A4	Port 2 Logical Outbound Packet Time-to-Live Configuration Register	P2LOPTTLCR
– 0xFFF901A8–0xFFF901AF	reserved	
– 0xFFF901B0	Port 2 Implementation Error Command and Status Register	P2IECSR
– 0xFFF901B4–0xFFF901BF	reserved	
– 0xFFF901C0	Port 2 Physical Configuration Register	P2PCR
– 0xFFF901C4–0xFFF901D7	reserved	
– 0xFFF901D8	Port 2 Serial Link Command And Status Register	P2SLCSR
– 0xFFF901DC–0xFFF901DF	reserved	
– 0xFFF901E0	Port 2 Serial Link Error Injection Configuration Register	P2SLEICR
– 0xFFF901E4	Port 2 Arbitration 0 Tx Configuration Register	P2A0TxCR
– 0xFFF901E8	Port 2 Arbitration 1 Tx Configuration Register	P2A1TxCR
– 0xFFF901EC	Port 2 Arbitration 2 Tx Configuration Register	P2A2TxCR
– 0xFFF901F0	Port 2 Message Request Tx Buffer Allocation Configuration Register 0	P2MReqTxBACR0
– 0xFFF901F4	Port 2 Message Request Tx Buffer Allocation Configuration Register 1	P2MReqTxBACR1
– 0xFFF901F8	Port 2 Message Request Tx Buffer Allocation Configuration Register 2	P2MReqTxBACR2
– 0xFFF901FC	Port 2 Message Response/Flow Control Tx Buffer Allocation Configuration Register	P2MRspFcTxBACR
– 0xFFF901A0–0xFFF90BF7	reserved	
– 0xFFF90BF8	IP Block Revision Register 1	IPBRR1
– 0xFFF90BFC	IP Block Revision Register 2	IPBRR2
– 0xFFF90C00	Port 1 RapidIO Outbound Window Translation Address Register 0	P1ROWTAR0
– 0xFFF90C04	Port 1 RapidIO Outbound Window Translation Extended Address Register 0	P1ROWTEAR0
– 0xFFF90C08–0xFFF90C0F	reserved	
– 0xFFF90C10	Port 1 RapidIO Outbound Window Attributes Register 0	P1ROWAR0
– 0xFFF90C14–0xFFF90C1F	reserved	
– 0xFFF90C20	Port 1 RapidIO Outbound Window Translation Address Register 1	P1ROWTAR1
– 0xFFF90C24	Port 1 RapidIO Outbound Window Translation Extended Address Register 1	P1ROWTEAR1
– 0xFFF90C28–0xFFF90C2F	reserved	
– 0xFFF90C30	Port 1 RapidIO Outbound Window Attributes Register 1	P1ROWAR1

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF90C34	Port 1 RapidIO Outbound Window Segment 1 Register 1	P1ROWS1R1
– 0xFFF90C38	Port 1 RapidIO Outbound Window Segment 2 Register 1	P1ROWS2R1
– 0xFFF90C3C	Port 1 RapidIO Outbound Window Segment 3 Register 1	P1ROWS3R1
– 0xFFF90C40	Port 1 RapidIO Outbound Window Translation Address Register 2	P1ROWTAR2
– 0xFFF90C44	Port 1 RapidIO Outbound Window Translation Extended Address Register 2	P1ROWTEAR2
– 0xFFF90C48	Port 1 RapidIO Outbound Window Base Address Register 2	P1ROWBAR2
– 0xFFF90C4C–0xFFF90C4F	reserved	
– 0xFFF90C50	Port 1 RapidIO Outbound Window Attributes Register 2	P1ROWAR2
– 0xFFF90C54	Port 1 RapidIO Outbound Window Segment 1 Register 2	P1ROWS1R2
– 0xFFF90C58	Port 1 RapidIO Outbound Window Segment 2 Register 2	P1ROWS2R2
– 0xFFF90C5C	Port 1 RapidIO Outbound Window Segment 3 Register 2	P1ROWS3R2
– 0xFFF90C60	Port 1 RapidIO Outbound Window Translation Address Register 3	P1ROWTAR3
– 0xFFF90C64	Port 1 RapidIO Outbound Window Translation Extended Address Register 3	P1ROWTEAR3
– 0xFFF90C68	Port 1 RapidIO Outbound Window Base Address Register 3	P1ROWBAR3
– 0xFFF90C6C–0xFFF90C6F	reserved	
– 0xFFF90C70	Port 1 RapidIO Outbound Window Attributes Register 3	P1ROWAR3
– 0xFFF90C74	Port 1 RapidIO Outbound Window Segment 1 Register 3	P1ROWS1R3
– 0xFFF90C78	Port 1 RapidIO Outbound Window Segment 2 Register 3	P1ROWS2R3
– 0xFFF90C7C	Port 1 RapidIO Outbound Window Segment 3 Register 3	P1ROWS3R3
– 0xFFF90C80	Port 1 RapidIO Outbound Window Translation Address Register 4	P1ROWTAR4
– 0xFFF90C84	Port 1 RapidIO Outbound Window Translation Extended Address Register 4	P1ROWTEAR4
– 0xFFF90C88	Port 1 RapidIO Outbound Window Base Address Register 4	P1ROWBAR4
– 0xFFF90C8C–0xFFF90C8F	reserved	
– 0xFFF90C90	Port 1 RapidIO Outbound Window Attributes Register 4	P1ROWAR4
– 0xFFF90C94	Port 1 RapidIO Outbound Window Segment 1 Register 4	P1ROWS1R4
– 0xFFF90C98	Port 1 RapidIO Outbound Window Segment 2 Register 4	P1ROWS2R4
– 0xFFF90C9C	Port 1 RapidIO Outbound Window Segment 3 Register 4	P1ROWS3R4
– 0xFFF90CA0	Port 1 RapidIO Outbound Window Translation Address Register 5	P1ROWTAR5
– 0xFFF90CA4	Port 1 RapidIO Outbound Window Translation Extended Address Register 5	P1ROWTEAR5
– 0xFFF90CA8	Port 1 RapidIO Outbound Window Base Address Register 5	P1ROWBAR5
– 0xFFF90CAC–0xFFF90CAF	reserved	
– 0xFFF90CB0	Port 1 RapidIO Outbound Window Attributes Register 5	P1ROWAR5
– 0xFFF90CB4	Port 1 RapidIO Outbound Window Segment 1 Register 5	P1ROWS1R5
– 0xFFF90CB8	Port 1 RapidIO Outbound Window Segment 2 Register 5	P1ROWS2R5
– 0xFFF90CBC	Port 1 RapidIO Outbound Window Segment 3 Register 5	P1ROWS3R5
– 0xFFF90CC0	Port 1 RapidIO Outbound Window Translation Address Register 6	P1ROWTAR6
– 0xFFF90CC4	Port 1 RapidIO Outbound Window Translation Extended Address Register 6	P1ROWTEAR6
– 0xFFF90CC8	Port 1 RapidIO Outbound Window Base Address Register 6	P1ROWBAR6

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF90CCC–0xFFF90CCF	reserved	
– 0xFFF90CD0	Port 1 RapidIO Outbound Window Attributes Register 6	P1ROWAR6
– 0xFFF90CD4	Port 1 RapidIO Outbound Window Segment 1 Register 6	P1ROWS1R6
– 0xFFF90CD8	Port 1 RapidIO Outbound Window Segment 2 Register 6	P1ROWS2R6
– 0xFFF90CDC	Port 1 RapidIO Outbound Window Segment 3 Register 6	P1ROWS3R6
– 0xFFF90CE0	Port 1 RapidIO Outbound Window Translation Address Register 7	P1ROWTAR7
– 0xFFF90CE4	Port 1 RapidIO Outbound Window Translation Extended Address Register 7	P1ROWTEAR7
– 0xFFF90CE8	Port 1 RapidIO Outbound Window Base Address Register 7	P1ROWBAR7
– 0xFFF90CEC–0xFFF90CEF	reserved	
– 0xFFF90CF0	Port 1 RapidIO Outbound Window Attributes Register 7	P1ROWAR7
– 0xFFF90CF4	Port 1 RapidIO Outbound Window Segment 1 Register 7	P1ROWS1R7
– 0xFFF90CF8	Port 1 RapidIO Outbound Window Segment 2 Register 7	P1ROWS2R7
– 0xFFF90CFC	Port 1 RapidIO Outbound Window Segment 3 Register 7	P1ROWS3R7
– 0xFFF90D00	Port 1 RapidIO Outbound Window Translation Address Register 8	P1ROWTAR8
– 0xFFF90D04	Port 1 RapidIO Outbound Window Translation Extended Address Register 8	P1ROWTEAR8
– 0xFFF90D08	Port 1 RapidIO Outbound Window Base Address Register 8	P1ROWBAR8
– 0xFFF90D0C–0xFFF90D0F	reserved	
– 0xFFF90D10	Port 1 RapidIO Outbound Window Attributes Register 8	P1ROWAR8
– 0xFFF90D14	Port 1 RapidIO Outbound Window Segment 1 Register 8	P1ROWS1R8
– 0xFFF90D18	Port 1 RapidIO Outbound Window Segment 2 Register 8	P1ROWS2R8
– 0xFFF90D1C	Port 1 RapidIO Outbound Window Segment 3 Register 8	P1ROWS3R8
– 0xFFF90D20–0xFFF90D5F	reserved	
– 0xFFF90D60	Port 1 RapidIO Inbound Window Translation Address Register 4	P1RIWTAR4
– 0xFFF90D64–0xFFF90D67	reserved	
– 0xFFF90D68	Port 1 RapidIO Inbound Window Base Address Register 4	P1RIWBAR4
– 0xFFF90D6C–0xFFF90D6F	reserved	
– 0xFFF90D70	Port 1 RapidIO Inbound Window Attributes Register 4	P1RIWAR4
– 0xFFF90D74–0xFFF90D7F	reserved	
– 0xFFF90D80	Port 1 RapidIO Inbound Window Translation Address Register 3	P1RIWTAR3
– 0xFFF90D84–0xFFF90D87	reserved	
– 0xFFF90D88	Port 1 RapidIO Inbound Window Base Address Register 3	P1RIWBAR3
– 0xFFF90D8C–0xFFF90D8F	reserved	
– 0xFFF90D90	Port 1 RapidIO Inbound Window Attributes Register 3	P1RIWAR3
– 0xFFF90D94–0xFFF90D9F	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF90DA0	Port 1 RapidIO Inbound Window Translation Address Register 2	P1RIWTAR2
– 0xFFF90DA4–0xFFF90DA7	reserved	
– 0xFFF90DA8	Port 1 RapidIO Inbound Window Base Address Register 2	P1RIWBAR2
– 0xFFF90DAC–0xFFF90DAF	reserved	
– 0xFFF90DB0	Port 1 RapidIO inbound window attributes register 2	P1RIWAR2
– 0xFFF90DB4–0xFFF90DBF	reserved	
– 0xFFF90DC0	Port 1 RapidIO Inbound Window Translation Address Register 1	P1RIWTAR1
– 0xFFF90DC4–0xFFF90DC7	reserved	
– 0xFFF90DC8	Port 1 RapidIO Inbound Window Base Address Register 1	P1RIWBAR1
– 0xFFF90DCC–0xFFF90DCF	reserved	
– 0xFFF90DD0	Port 1 RapidIO Inbound Window Attributes Register 1	P1RIWAR1
– 0xFFF90DD4–0xFFF90DDF	reserved	
– 0xFFF90DE0	Port 1 RapidIO Inbound Window Translation Address Register 0	P1RIWTAR0
– 0xFFF90DE4–0xFFF90DEF	reserved	
– 0xFFF90DF0	Port 1 RapidIO Inbound Window Attributes Register 0	P1RIWAR0
– 0xFFF90DF4–0xFFF92DFF	reserved	
– 0xFFF90E00	Port 2 RapidIO Outbound Window Translation Address Register 0	P2ROWTAR0
– 0xFFF90E04	Port 2 RapidIO Outbound Window Translation Extended Address Register 0	P2ROWTEAR0
– 0xFFF90E08	Port 2 RapidIO Outbound Window Base Address Register 0	P2ROWBAR0
– 0xFFF90E0C–0xFFF90E0F	reserved	
– 0xFFF90E10	Port 2 RapidIO Outbound Window Attributes Register 0	P2ROWAR0
– 0xFFF90E14–0xFFF90E1F	reserved	
– 0xFFF90E20	Port 2 RapidIO Outbound Window Translation Address Register 1	P2ROWTAR1
– 0xFFF90E24	Port 2 RapidIO Outbound Window Translation Extended Address Register 1	P2ROWTEAR1
– 0xFFF90E28	Port 2 RapidIO Outbound Window Base Address Register 1	P2ROWBAR1
– 0xFFF90E2C–0xFFF90E2F	reserved	
– 0xFFF90E30	Port 2 RapidIO Outbound Window Attributes Register 1	P2ROWAR1
– 0xFFF90E34	Port 2 RapidIO Outbound Window Segment 1 Register 1	P2ROWS1R1
– 0xFFF90E38	Port 2 RapidIO Outbound Window Segment 2 Register 1	P2ROWS2R1
– 0xFFF90E3C	Port 2 RapidIO Outbound Window Segment 3 Register 1	P2ROWS3R1
– 0xFFF90E40	Port 2 RapidIO Outbound Window Translation Address Register 2	P2ROWTAR2
– 0xFFF90E44	Port 2 RapidIO Outbound Window Translation Extended Address Register 2	P2ROWTEAR2
– 0xFFF90E48	Port 2 RapidIO Outbound Window Base Address Register 2	P2ROWBAR2

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF90E4C–0xFFF90E4F	reserved	
– 0xFFF90E50	Port 2 RapidIO Outbound Window Attributes Register 2	P2ROWAR2
– 0xFFF90E54	Port 2 RapidIO Outbound Window Segment 1 Register 2	P2ROWS1R2
– 0xFFF90E58	Port 2 RapidIO Outbound Window Segment 2 Register 2	P2ROWS2R2
– 0xFFF90E5C	Port 2 RapidIO Outbound Window Segment 3 Register 2	P2ROWS3R2
– 0xFFF90E60	Port 2 RapidIO Outbound Window Translation Address Register 3	P2ROWTAR3
– 0xFFF90E64	Port 2 RapidIO Outbound Window Translation Extended Address Register 3	P2ROWTEAR3
– 0xFFF90E68	Port 2 RapidIO Outbound Window Base Address Register 3	P2ROWBAR3
– 0xFFF90E6C–0xFFF90E6F	reserved	
– 0xFFF90E70	Port 2 RapidIO Outbound Window Attributes Register 3	P2ROWAR3
– 0xFFF90E74	Port 2 RapidIO Outbound Window Segment 1 Register 3	P2ROWS1R3
– 0xFFF90E78	Port 2 RapidIO Outbound Window Segment 2 Register 3	P2ROWS2R3
– 0xFFF90E7C	Port 2 RapidIO Outbound Window Segment 3 Register 3	P2ROWS3R3
– 0xFFF90E80	Port 2 RapidIO Outbound Window Translation Address Register 4	P2ROWTAR4
– 0xFFF90E84	Port 2 RapidIO Outbound Window Translation Extended Address Register 4	P2ROWTEAR4
– 0xFFF90E88	Port 2 RapidIO Outbound Window Base Address Register 4	P2ROWBAR4
– 0xFFF90E8C–0xFFF90E8F	reserved	
– 0xFFF90E90	Port 2 RapidIO Outbound Window Attributes Register 4	P2ROWAR4
– 0xFFF90E94	Port 2 RapidIO Outbound Window Segment 1 Register 4	P2ROWS1R4
– 0xFFF90E98	Port 2 RapidIO Outbound Window Segment 2 Register 4	P2ROWS2R4
– 0xFFF90E9C	Port 2 RapidIO Outbound Window Segment 3 Register 4	P2ROWS3R4
– 0xFFF90EA0	Port 2 RapidIO Outbound Window Translation Address Register 5	P2ROWTAR5
– 0xFFF90EA4	Port 2 RapidIO Outbound Window Translation Extended Address Register 5	P2ROWTEAR5
– 0xFFF90EA8	Port 2 RapidIO Outbound Window Base Address Register 5	P2ROWBAR5
– 0xFFF90EAC–0xFFF90EAF	reserved	
– 0xFFF90EB0	Port 2 RapidIO Outbound Window Attributes Register 5	P2ROWAR5
– 0xFFF90EB4	Port 2 RapidIO Outbound Window Segment 1 Register 5	P2ROWS1R5
– 0xFFF90EB8	Port 2 RapidIO Outbound Window Segment 2 Register 5	P2ROWS2R5
– 0xFFF90EBC	Port 2 RapidIO Outbound Window Segment 3 Register 5	P2ROWS3R5
– 0xFFF90EC0	Port 2 RapidIO Outbound Window Translation Address Register 6	P2ROWTAR6
– 0xFFF90EC4	Port 2 RapidIO Outbound Window Translation Extended Address Register 6	P2ROWTEAR6
– 0xFFF90EC8	Port 2 RapidIO Outbound Window Base Address Register 6	P2ROWBAR6
– 0xFFF90ECC–0xFFF90ECF	reserved	
– 0xFFF90ED0	Port 2 RapidIO Outbound Window Attributes Register 6	P2ROWAR6
– 0xFFF90ED4	Port 2 RapidIO Outbound Window Segment 1 Register 6	P2ROWS1R6
– 0xFFF90ED8	Port 2 RapidIO Outbound Window Segment 2 Register 6	P2ROWS2R6
– 0xFFF90EDC	Port 2 RapidIO Outbound Window Segment 3 Register 6	P2ROWS3R6
– 0xFFF90EE0	Port 2 RapidIO Outbound Window Translation Address Register 7	P2ROWTAR7

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF90EE4	Port 2 RapidIO Outbound Window Translation Extended Address Register 7	P2ROWTEAR7
– 0xFFF90EE8	Port 2 RapidIO Outbound Window Base Address Register 7	P2ROWBAR7
– 0xFFF90EEC–0xFFF90EEF	reserved	
– 0xFFF90EF0	Port 2 RapidIO Outbound Window Attributes Register 7	P2ROWAR7
– 0xFFF90EF4	Port 2 RapidIO Outbound Window Segment 1 Register 7	P2ROWS1R7
– 0xFFF90EF8	Port 2 RapidIO Outbound Window Segment 2 Register 7	P2ROWS2R7
– 0xFFF90EFC	Port 2 RapidIO Outbound Window Segment 3 Register 7	P2ROWS3R7
– 0xFFF90F00	Port 2 RapidIO Outbound Window Translation Address Register 8	P2ROWTAR8
– 0xFFF90F04	Port 2 RapidIO Outbound Window Translation Extended Address Register 8	P2ROWTEAR8
– 0xFFF90F08	Port 2 RapidIO Outbound Window Base Address Register 8	P2ROWBAR8
– 0xFFF90F0C–0xFFF90F0F	reserved	
– 0xFFF90F10	Port 2 RapidIO Outbound Window Attributes Register 8	P2ROWAR8
– 0xFFF90F14	Port 2 RapidIO Outbound Window Segment 1 Register 8	P2ROWS1R8
– 0xFFF90F18	Port 2 RapidIO Outbound Window Segment 2 Register 8	P2ROWS2R8
– 0xFFF90F1C	Port 2 RapidIO Outbound Window Segment 3 Register 8	P2ROWS3R8
– 0xFFF90F20–0xFFF90F5F	reserved	
– 0xFFF90F60	Port 2 RapidIO Inbound Window Translation Address Register 4	P2RIWTAR4
– 0xFFF90F64–0xFFF90F67	reserved	
– 0xFFF90F68	Port 2 RapidIO Inbound Window Base Address Register 4	P2RIWBAR4
– 0xFFF90F6C–0xFFF90F6F	reserved	
– 0xFFF90F70	Port 2 RapidIO Inbound Window Attributes Register 4	P2RIWAR4
– 0xFFF90F74–0xFFF90F7F	reserved	
– 0xFFF90F80	Port 2 RapidIO Inbound Window Translation Address Register 3	P2RIWTAR3
– 0xFFF90F84–0xFFF90F87	reserved	
– 0xFFF90F88	Port 2 RapidIO Inbound Window Base Address Register 3	P2RIWBAR3
– 0xFFF90F8C–0xFFF90F8F	reserved	
– 0xFFF90F90	Port 2 RapidIO Inbound Window Attributes Register 3	P2RIWAR3
– 0xFFF90F94–0xFFF90F9F	reserved	
– 0xFFF90FA0	Port 2 RapidIO Inbound Window Translation Address Register 2	P2RIWTAR2
– 0xFFF90FA4–0xFFF90FA7	reserved	
– 0xFFF90FA8	Port 2 RapidIO Inbound Window Base Address Register 2	P2RIWBAR2
– 0xFFF90FAC–0xFFF90FAF	reserved	
– 0xFFF90FB0	Port 2 RapidIO inbound window attributes register 2	P2RIWAR2

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF90FB4–0xFFF90FBF	reserved	
– 0xFFF90FC0	Port 2 RapidIO Inbound Window Translation Address Register 1	P2RIWTAR1
– 0xFFF90FC4–0xFFF90FC7	reserved	
– 0xFFF90FC8	Port 2 RapidIO Inbound Window Base Address Register 1	P2RIWBAR1
– 0xFFF90FCC–0xFFF90FCF	reserved	
– 0xFFF90FD0	Port 2 RapidIO Inbound Window Attributes Register 1	P2RIWAR1
– 0xFFF90FD4–0xFFF90FDF	reserved	
– 0xFFF90FE0	Port 2 RapidIO Inbound Window Translation Address Register 0	P2RIWTAR0
– 0xFFF90FE4–0xFFF90FE7	reserved	
– 0xFFF90FE8	Port 2 RapidIO Inbound Window Base Address Register 1	P2RIWBAR1
– 0xFFF90FEC–0xFFF90FEF	reserved	
– 0xFFF90FF0	Port 2 RapidIO Inbound Window Attributes Register 0	P2RIWAR0
– 0xFFF90FF4–0xFFF9FFFF	reserved	
• 0xFFFA0000–0xFFFA0FFF	reserved	
• 0xFFFA1000–0xFFFA103F	HSSI OCN Crossbar Switch to MBus0 (see **Chapter 15**, *High Speed Serial Interface (HSSI) Subsystem*)	
• 0xFFFA1040–0xFFFA107F	HSSI OCN Crossbar Switch to MBus1 (see **Chapter 15**, *High Speed Serial Interface (HSSI) Subsystem*)	
• 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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	
– 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	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 0xFFFABD2C–0xFFFABD2F	reserved	
– 0xFFFABD30	Local Access Window Attributes Register 9	D1LAWAR9
– 0xFFFABD34–0xFFFA8FFF	reserved	
• 0xFFFAB000–0xFFFABFFF	DMA Controller 1 to OCN	
• 0xFFFAC000–0xFFFACFFF	reserved	
• 0xFFFAD000–0xFFFAD01F	SerDes PHY Registers	
• 0xFFFAD000	SRDS Bank 1 Reset Control Register	SRDSB1RSTCTL
• 0xFFFAD004	SRDS Bank 1 PLL Control Register 0	SRDSB1PLLCR0
• 0xFFFAD008–0xFFFAD01F	reserved	
• 0xFFFAD020	SRDS Bank 2 Reset Control Register	SRDSB2RSTCTL
• 0xFFFAD024	SRDS Bank 2 PLL Control Register 0	SRDSB2PLLCR0
• 0xFFFAD028–0xFFFAD1FF	reserved	
• 0xFFFAD200	Lane A General Control Register 0	LAGCR0
• 0xFFFAD204	Lane A General Control Register 1	LAGCR1
• 0xFFFAD208–0xFFFAD20F	reserved	
• 0xFFFAD210	Lane A Receive Equalization Control Register 0	LARECR0
• 0xFFFAD214–0xFFFAD217	reserved	
• 0xFFFAD218	Lane A Transmit Equalization Control Register 0	LATECR0
• 0xFFFAD21C–0xFFFAD23B	reserved	
• 0xFFFAD23C	Lane A Test Control/Status Register 3	LATCSR3

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFFAD240	Lane B General Control Register 0	LBGCR0
• 0xFFFAD244	Lane B General Control Register 1	LBGCR1
• 0xFFFAD248–0xFFFAD24F	reserved	
• 0xFFFAD250	Lane B Receive Equalization Control Register 0	LBRECR0
• 0xFFFAD254–0xFFFAD257	reserved	
• 0xFFFAD258	Lane B Transmit Equalization Control Register 0	LBTECR0
• 0xFFFAD25C–0xFFFAD27B	reserved	
• 0xFFFAD27C	Lane B Test Control/Status Register 3	LBTCSR3
• 0xFFFAD280	Lane C General Control Register 0	LCGCR0
• 0xFFFAD284	Lane C General Control Register 1	LCGCR1
• 0xFFFAD288–0xFFFAD28F	reserved	
• 0xFFFAD290	Lane C Receive Equalization Control Register 0	LCRECR0
• 0xFFFAD294–0xFFFAD297	reserved	
• 0xFFFAD298	Lane C Transmit Equalization Control Register 0	LCTECR0
• 0xFFFAD29C–0xFFFAD2BB	reserved	
• 0xFFFAD2BC	Lane C Test Control/Status Register 3	LCTCSR3
• 0xFFFAD2C0	Lane D General Control Register 0	LDGCR0
• 0xFFFAD2C4	Lane D General Control Register 1	LDGCR1
• 0xFFFAD2C8–0xFFFAD2CF	reserved	
• 0xFFFAD2D0	Lane D Receive Equalization Control Register 0	LDRECR0
• 0xFFFAD2D4–0xFFFAD2D7	reserved	
• 0xFFFAD2D8	Lane D Transmit Equalization Control Register 0	LDTECR0
• 0xFFFAD2DC–0xFFFAD2FB	reserved	
• 0xFFFAD2FC	Lane D Test Control/Status Register 3	LDTCSR3
• 0xFFFAD300	Lane E General Control Register 0	LEGCR0
• 0xFFFAD304	Lane E General Control Register 1	LEGCR1
• 0xFFFAD308–0xFFFAD30F	reserved	
• 0xFFFAD310	Lane E Receive Equalization Control Register 0	LERECR0
• 0xFFFAD314–0xFFFAD317	reserved	
• 0xFFFAD318	Lane E Transmit Equalization Control Register 0	LETECR0
• 0xFFFAD31C–0xFFFAD33B	reserved	
• 0xFFFAD33C	Lane e Test Control/Status Register 3	LETCSR3
• 0xFFFAD340	Lane F General Control Register 0	LFGCR0

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFFAD344	Lane F General Control Register 1	LFGCR1
• 0xFFFAD348–0xFFFAD34F	reserved	
• 0xFFFAD350	Lane F Receive Equalization Control Register 0	LFRECR0
• 0xFFFAD354–0xFFFAD357	reserved	
• 0xFFFAD358	Lane F Transmit Equalization Control Register 0	LFTECR0
• 0xFFFAD35C–0xFFFAD37B	reserved	
• 0xFFFAD37C	Lane F Test Control/Status Register 3	LFTCSR3
• 0xFFFAD380	Lane G General Control Register 0	LGGCR0
• 0xFFFAD384	Lane G General Control Register 1	LGGCR1
• 0xFFFAD388–0xFFFAD38F	reserved	
• 0xFFFAD390	Lane G Receive Equalization Control Register 0	LGRECR0
• 0xFFFAD394–0xFFFAD397	reserved	
• 0xFFFAD398	Lane G Transmit Equalization Control Register 0	LGTECR0
• 0xFFFAD39C–0xFFFAD3BB	reserved	
• 0xFFFAD3BC	Lane G Test Control/Status Register 3	LGTCSR3
• 0xFFFAD3C0	Lane H General Control Register 0	LHGCR0
• 0xFFFAD3C4	Lane H General Control Register 1	LHGCR1
• 0xFFFAD3C8–0xFFFAD3CF	reserved	
• 0xFFFAD3D0	Lane H Receive Equalization Control Register 0	LHRECR0
• 0xFFFAD3D4–0xFFFAD3D7	reserved	
• 0xFFFAD3D8	Lane A Transmit Equalization Control Register 0	LHTECR0
• 0xFFFAD3DC–0xFFFAD3FB	reserved	
• 0xFFFAD3FC	Lane H Test Control/Status Register 3	LHTCSR3
• 0xFFFAD400	Lane I General Control Register 0	LIGCR0
• 0xFFFAD404	Lane I General Control Register 1	LIGCR1
• 0xFFFAD408–0xFFFAD40F	reserved	
• 0xFFFAD410	Lane I Receive Equalization Control Register 0	LIRECR0
• 0xFFFAD414–0xFFFAD417	reserved	
• 0xFFFAD418	Lane I Transmit Equalization Control Register 0	LITECR0
• 0xFFFAD41C–0xFFFAD43B	reserved	
• 0xFFFAD43C	Lane I Test Control/Status Register 3	LITCSR3
• 0xFFFAD440	Lane J General Control Register 0	LJGCR0
• 0xFFFAD444	Lane J General Control Register 1	LJGCR1

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
• 0xFFFAD448–0xFFFAD44F	reserved	
• 0xFFFAD450	Lane J Receive Equalization Control Register 0	LJRECR0
• 0xFFFAD454–0xFFFAD457	reserved	
• 0xFFFAD458	Lane J Transmit Equalization Control Register 0	LJTECR0
• 0xFFFAD45C–0xFFFAD47B	reserved	
• 0xFFFAD47C	Lane J Test Control/Status Register 3	LJTCSR3
• 0xFFFAD480–0xFFFB6FFF	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	PCI Express Interrupt Status Register	PEX_INT_STAT
– 0xFFFB701C–0xFFFB701F	reserved	
– 0xFFFB7020	PCI Express PME and Message Detect Register	PEX_PME_MES_DR
– 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
– 0xFFF87108	PCI Express Link Speed Control Register	PEX_LSCR
– 0xFFF8710C	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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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
– 0xFFFB7C9C–0xFFFB7CFF	reserved	
– 0xFFF87D00	PCI Express MSI Inbound Translation Address Register	PEXMSIITAR
– 0xFFF87D04–0xFFF87D07	reserved	
– 0xFFF87D08	PCI Express MSI Inbound Window Base Address Register	PEXMSIIWBAR
– 0xFFF87D0C	PCI Express MSI Inbound Window Base Extended Address Register	PEXMSIIWBEAR
– 0xFFF87D10	PCI Express MSI Inbound Window Attributes Register	PEXMSIIWAR
– 0xFFF87D14–0xFFF87D7F	reserved	
– 0xFFF87D80	PCI Express Inbound Translation Address Register 3	PEXITAR3
– 0xFFFB7D84–0xFFFB7D87	reserved	
– 0xFFFB7D88	PCI Express Inbound Window Base Address Register 3	PEXIWBAR3
– 0xFFFB7D8C	PCI Express Inbound Window Base Extended Address Register 3	PEXIWBEAR3
– 0xFFFB7D90	PCI Express Inbound Window Attributes Register 3	PEXIWAR3

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFB7D94–0xFFFB7D9F	reserved	
– 0xFFFB7DA0	PCI Express Inbound Translation Address Register 2	PEXITAR2
– 0xFFFB7DA4–0xFFFB7DA7	reserved	
– 0xFFFB7DA8	PCI Express Inbound Window Base Address Register 2	PEXIWBAR2
– 0xFFFB7DAC	PCI Express Inbound Window Base Extended Address Register 2	PEXIWBEAR2
– 0xFFFB7DB0	PCI Express Inbound Window Attributes Register 2	PEXIWAR2
– 0xFFFB7DB4–0xFFFB7DBF	reserved	
– 0xFFFB7DC0	PCI Express Inbound Translation Address Register 1	PEXITAR1
– 0xFFFB7DC4–0xFFFB7DC7	reserved	
– 0xFFFB7DC8	PCI Express Inbound Window Base Address Register 1	PEXIWBAR1
– 0xFFFB7DCC–0xFFFB7DCF	reserved	
– 0xFFFB7DD0	PCI Express Inbound Window Attributes Register 1	PEXIWAR1
– 0xFFFB7DD4–0xFFFB7DDF	reserved	
– 0xFFFB7DE0	PCI Express Inbound Translation Address Register 0	PEXITAR0
– 0xFFFB7DE4–0xFFFB7DE7	reserved	
– 0xFFFB7DE8	PCI Express Inbound Window Base Address Register 1	PEXIWBAR0
– 0xFFFB7DEC–0xFFFB7DEF	reserved	
– 0xFFFB7DF0	PCI Express Inbound Window Attributes Register 1	PEXIWAR0
– 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	
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFB7E38–0xFFFB9FFF	reserved	
– 0xFFFBA000–0xFFFBAFFF	CLASS1	
– 0xFFFBA000–0xFFFBA7FF	reserved	
– 0xFFFBA800	CLASS1 Priority Mapping Register 0	C1PMR0
– 0xFFFBA804	CLASS1 Priority Mapping Register 1	C1PMR1
– 0xFFFBA808	CLASS1 Priority Mapping Register 2	C1PMR2
– 0xFFFBA80C	CLASS1 Priority Mapping Register 3	C1PMR3
– 0xFFFBA810	CLASS1 Priority Mapping Register 4	C1PMR4
– 0xFFFBA814	CLASS1 Priority Mapping Register 5	C1PMR5
– 0xFFFBA818–0xFFFBA83F	reserved	
– 0xFFFBA840	CLASS1 Priority Auto Upgrade Value Register 0	C1PAVR0
– 0xFFFBA844	CLASS1 Priority Auto Upgrade Value Register 1	C1PAVR1
– 0xFFFBA848	CLASS1 Priority Auto Upgrade Value Register 2	C1PAVR2
– 0xFFFBA84C	CLASS1 Priority Auto Upgrade Value Register 3	C1PAVR3
– 0xFFFBA850	CLASS1 Priority Auto Upgrade Value Register 4	C1PAVR4
– 0xFFFBA854	CLASS1 Priority Auto Upgrade Value Register 5	C1PAVR5
– 0xFFFBA858–0xFFFBA87F	reserved	
– 0xFFFBA880	CLASS1 Priority Auto Upgrade Control Register 0	C1PACR0
– 0xFFFBA884	CLASS1 Priority Auto Upgrade Control Register 1	C1PACR1
– 0xFFFBA888	CLASS1 Priority Auto Upgrade Control Register 2	C1PACR2
– 0xFFFBA88C	CLASS1 Priority Auto Upgrade Control Register 3	C1PACR3
– 0xFFFBA890	CLASS1 Priority Auto Upgrade Control Register 4	C1PACR4
– 0xFFFBA894	CLASS1 Priority Auto Upgrade Control Register 5	C1PACR5
– 0xFFFBA898–0xFFFBA97F	reserved	
– 0xFFFBA980	CLASS1 Error Address Register 0	C1EAR0
– 0xFFFBA984	CLASS1 Error Address Register 1	C1EAR1
– 0xFFFBA988	CLASS1 Error Address Control Register 2	C1EAR2
– 0xFFFBA98C	CLASS1 Error Address Control Register 3	C1EAR3
– 0xFFFBA990	CLASS1 Error Address Control Register 4	C1EAR4
– 0xFFFBA994	CLASS1 Error Address Control Register 5	C1EAR5
– 0xFFFBA998–0xFFFBA9BF	reserved	
– 0xFFFBA9C0	CLASS1 Error Extended Address Register 0	C1EEAR0
– 0xFFFBA9C4	CLASS1 Error Extended Address Register 1	C1EEAR1
– 0xFFFBA9C8	CLASS1 Error Extended Address Register 2	C1EEAR2
– 0xFFFBA9CC	CLASS1 Error Extended Address Register 3	C1EEAR3
– 0xFFFBA9D0	CLASS1 Error Extended Address Register 4	C1EEAR4

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFBA9D4	CLASS1 Error Extended Address Register 5	C1EEAR5
– 0xFFFBA9D8–0xFFFBA9FF	reserved	
– 0xFFFBAA00	CLASS1 Initiator Profiling Configuration Register 0	C1IPCR0
– 0xFFFBAA04	CLASS1 Initiator Profiling Configuration Register 1	C1IPCR1
– 0xFFFBAA08	CLASS1 Initiator Profiling Configuration Register 2	C1IPCR2
– 0xFFFBAA0C	CLASS1 Initiator Profiling Configuration Register 3	C1IPCR3
– 0xFFFBAA10	CLASS1 Initiator Profiling Configuration Register 4	C1IPCR4
– 0xFFFBAA14	CLASS1 Initiator Profiling Configuration Register 5	C1IPCR5
– 0xFFFBAA18–0xFFFBAA3F	reserved	
– 0xFFFBAA40	CLASS1 Initiator Watch Point Control Register 0	C1IWPCR0
– 0xFFFBAA44	CLASS1 Initiator Watch Point Control Register 1	C1IWPCR1
– 0xFFFBAA48	CLASS1 Initiator Watch Point Control Register 2	C1IWPCR2
– 0xFFFBAA4C	CLASS1 Initiator Watch Point Control Register 3	C1IWPCR3
– 0xFFFBAA50	CLASS1 Initiator Watch Point Control Register 4	C1IWPCR4
– 0xFFFBAA54	CLASS1 Initiator Watch Point Control Register 5	C1IWPCR5
– 0xFFFBAA58–0xFFFBAA7F	reserved	
– 0xFFFBAA80	CLASS1 Arbitration Weight Register 0	C1AWR0
– 0xFFFBAA84	CLASS1 Arbitration Weight Register 1	C1AWR1
– 0xFFFBAA88	CLASS1 Arbitration Weight Register 2	C1AWR2
– 0xFFFBAA8C	CLASS1 Arbitration Weight Register 3	C1AWR3
– 0xFFFBAA90	CLASS1 Arbitration Weight Register 4	C1AWR4
– 0xFFFBAA94	CLASS1 Arbitration Weight Register 5	C1AWR5
– 0xFFFBAA98–0xFFFBAC00	reserved	
– 0xFFFBAC04	CLASS1 Start Address Decoder 1	C1SAD1
– 0xFFFBAC08	CLASS1 Start Address Decoder 2	C1SAD2
– 0xFFFBAC0C–0xFFFBAC00	reserved	
– 0xFFFBAC44	CLASS1 End Address Decoder 1	C1EAD1
– 0xFFFBAC48	CLASS1 End Address Decoder 2	C1EAD2
– 0xFFFBAC4C–0xFFFBAC83	reserved	
– 0xFFFBAC84	CLASS1 Attributes Decoder 1	C1ATD1
– 0xFFFBAC88	CLASS1 Attributes Decoder 2	C1ATD2
– 0xFFFBAC8C–0xFFFBAD7F	reserved	
– 0xFFFBAD80	CLASS1 IRQ Status Register	C1ISR
– 0xFFFBAC8C–0xFFFBADBF	reserved	
– 0xFFFBADC0	CLASS1 IRQ Enable Register	C1IER

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFBADC4–0xFFFBADFF	reserved	
– 0xFFFBAE00	CLASS1 Target Profiling Configuration Register	C1TPCR
– 0xFFFBAE04	CLASS1 Profiling Control Register	C1PCR
– 0xFFFBAE08	CLASS1 Watch Point Control Register	C1WPCR
– 0xFFFBAE0C	CLASS1 Watch Point Access Configuration Register	C1WPACR
– 0xFFFBAE10	CLASS1 Watch Point Extended Access Configuration Register	C1WPEACR
– 0xFFFBAE14	CLASS1 Watch Point Address Mask Register	C1WPAMR
– 0xFFFBAE18	CLASS1 Profiling Time-Out Register	C1PTOR
– 0xFFFBAE1C	CLASS1 Target Watch Point Control Register	C1TWPCR
– 0xFFFBAE20	CLASS1 Profiling IRQ Status Register	C1PISR
– 0xFFFBAE24	CLASS1 Profiling IRQ Enable Register	C1PIER
– 0xFFFBAE28–0xFFFBAE3F	reserved	
– 0xFFFBAE40	CLASS1 Profiling Reference Counter Register	C1PRCR
– 0xFFFBAE44	CLASS1 Profiling General Counter Register 0	C1PGCR0
– 0xFFFBAE48	CLASS1 Profiling General Counter Register 1	C1PGCR1
– 0xFFFBAE4C	CLASS1 Profiling General Counter Register 2	C1PGCR2
– 0xFFFBAE50	CLASS1 Profiling General Counter Register 3	C1PGCR3
– 0xFFFBAE54–0xFFFBAFBF	reserved	
– 0xFFFBAFC0	CLASS1 Arbitration Control Register	C1ACR
– 0xFFFBAFC4–0xFFFBAFFF	reserved	
– 0xFFFBB000–0xFFFBB7FF	reserved	
• 0xFFFBB800–0xFFFBB8FF	Performance Monitor	
– 0xFFFBB800	Performance Monitor Global Control Register	PMGCR
– 0xFFFBB804–0xFFFBB80F	reserved	
– 0xFFFBB810	Performance Monitor Local Control Register A0	PMLCA0
– 0xFFFBB814	Performance Monitor Local Control Register B0	PMLCB0
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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	—

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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	
– 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)	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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	—
– 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	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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	
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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	—

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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	—

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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
– 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

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 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	
– 0xFFFDF800– 0xFFFDFFFF	CRCU Input FIFO	—
• 0xFFFE0000– 0xFFFFEFFF	reserved	
0xFFFFF000– 0xFFFFFFFF	reserved	

10 SC3850 DSP Subsystem

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).

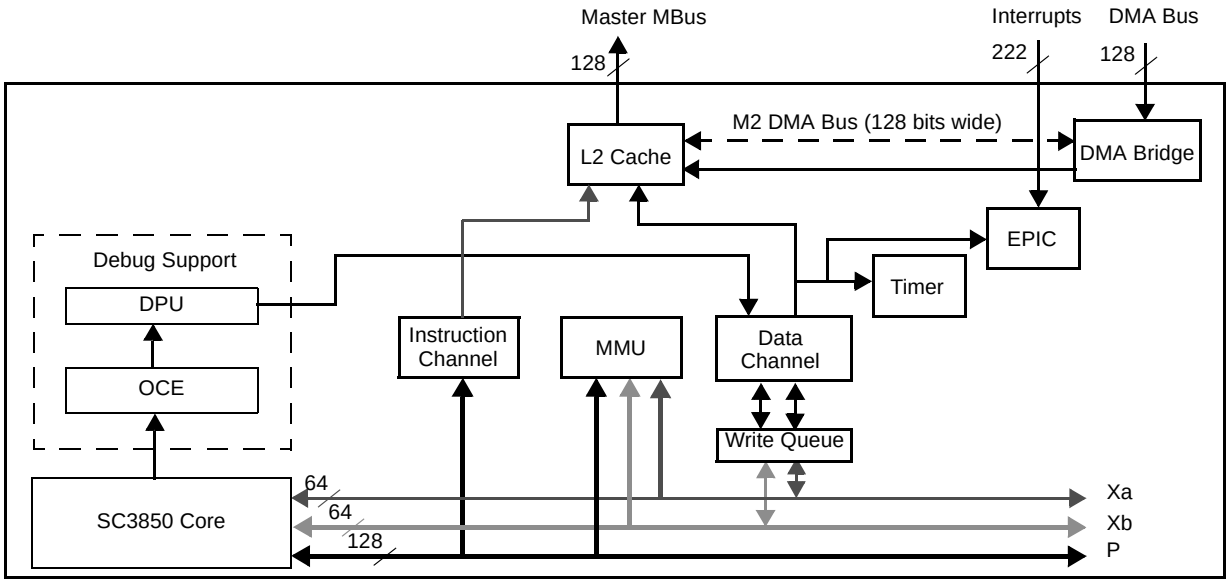


Figure 10-1. DSP Core Subsystem

Note: *The SC3850 DSP Core Reference Manual* and the *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
  - 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
  - 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 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.

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 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, 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, 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 *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 *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 8.2.38**, *Core Subsystem Slave Port General Configuration Register (CORE_SLV_GCR)*, on page 8-66 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 8.2.29**, *General Interrupt Register 5 (GIR5)*, on page 8-46 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 8-6 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 8.2.38***, *Core Subsystem Slave Port General Configuration Register (CORE_SLV_GCR)*, on page 8-66).

10.12 Programming Restrictions

The MSC8157E has the following programming restrictions when using the SC3850 DSP Subsystem:

- A write hit in the cache combined with DFLUSH/DSYNC can be lost for the following scenarios:
  - Write hit lost after DSYNC/DFLUSH with parallel WRITE/FETCH with no match.
  - Write hit before DSYNC is lost when the entrance to LWB is delayed by contentions.

  Use the following steps to prevent the write hit loss:

1. Insert a SYNCM instruction between a READ/WRITE/DFETCH/DMALLOC to the same 256 bytes followed by a DFLUSH/DSYNC command.
2. Use a SYNCM instruction after the last DFLUSH/DSYNC instruction.
3. No write is allowed between the first DFLUSH/DSYNC and the last DFLUSH/DSYNC SYNCM.

- To reduce performance degradation when the slave port is configured to low priority, always configure the Subsystem Port Priority Control Register (TSPPCR) general configuration register to high priority. The L2 cache arbiter does not treat the slave port access as part of the Round Robin arbitration scheme. Therefore, when the slave port accesses the cache with high priority, it always wins the arbitration. Because the slave port operates at half the frequency of the L2 and L1 caches, the cumulative L1 (I$ and D$) theoretical bandwidth can drop to 50% versus 33% in full Round Robin. When the slave port accesses the cache at low priority, the L1 (I$ and D$) always wins the arbitration and the slave port performance can decrease significantly. Arbitration between I$ and D$ is round robin with priority as defined.

11 Internal Memory Subsystem

The internal memory system supports 6624 KB 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 + 2 MB128-bit wide M3 memory connected to six internal memory buses. The 2 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 MSC8157E SC3850 DSP core subsystem. For detailed programming and functional information, refer to the *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).
- 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).
- 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 3 MB 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 667 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 six 512 KB groups, each with its own bus controller. The first two groups of memory are located at addresses 0xC0000000 through 0xC00FFFFF in the MSC8157E memory map. The last four groups are located at addresses 0xC010000 through 0xC02FFFFF. To support decreased power consumption, power to the last four 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 four 512 KB memory groups, the remaining two 512 KB groups still have power to provide support for the hardware semaphores and the RapidIO mail boxes.

11.6 Internal Boot ROM

The MSC8157E 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.

12 DDR SDRAM Memory Controller

The fully programmable DDR SDRAM controller supports most JEDEC standard x8 and x16 DDR3 available 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.

Note: In this chapter, the word *bank* refers to a physical bank specified by a chip select; *logical bank* refers to one of the four or eight sub-banks in each SDRAM chip. A sub-bank is specified by the 2 or 3 bits on the bank address (MBA) pins during a memory access.

Figure 12-1 shows a high-level view of the DDR memory controller with its associated interfaces.

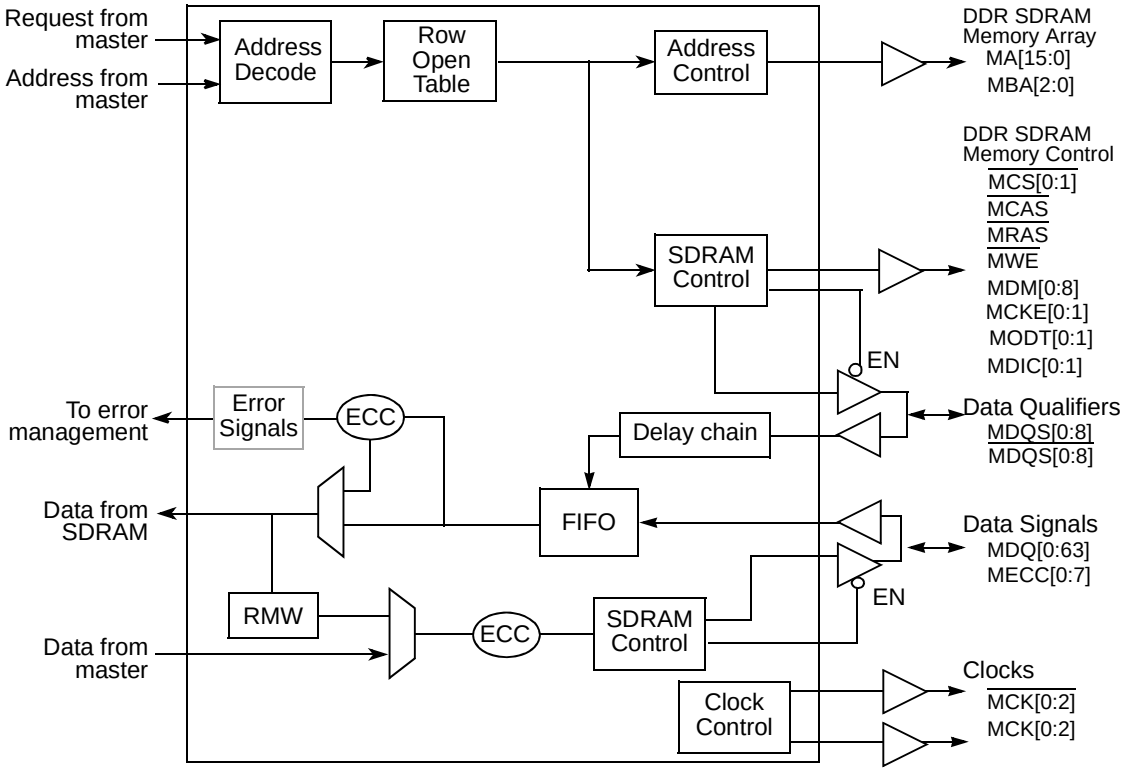


Figure 12-1. DDR SDRAM Memory Controller Simplified Block Diagram

12.1 DDR Memory Controller Features

The DDR memory controller includes these distinctive features:

- Support for DDR3 SDRAM
- 64-/72-bit SDRAM data bus, 32-/40-bit SDRAM for DDR3
- Programmable settings for meeting all SDRAM timing parameters
- The following SDRAM configurations are supported:
  - As many as two physical banks (chip selects), each bank independently addressable
  - 512-Mbit to 4-Gbit devices depending on internal device configuration with x8/x16 data ports (no direct x4 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
- Support for error injection

12.2 DDR Memory Controller Modes of Operation

The DDR memory controller supports the following modes:

- *Dynamic power management mode*. The DDR memory controller can reduce power consumption by negating the SDRAM CKE signal when no transactions are pending to the SDRAM.
- *Auto-precharge mode*. Clearing DDR_SDRAM_INTERVAL[BSTOPRE] causes the memory controller to issue an auto-precharge command with every read or write transaction. Auto-precharge mode can be enabled for separate chip selects by setting CS*n*_CONFIG[AP_*n*_EN].

12.3 DDR Controller Functional Description

The DDR SDRAM controller controls processor and I/O interactions with system memory. It provides support for JEDEC-compliant 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.

Figure 12-2 is a high-level block diagram of the DDR memory controller. Requests are received from the internal mastering device and the address is decoded to generate the physical bank, logical bank, row, and column addresses. The transaction is compared with values in the row open table to determine if the address maps to an open page. If the transaction does not map to an open page, an active command is issued.

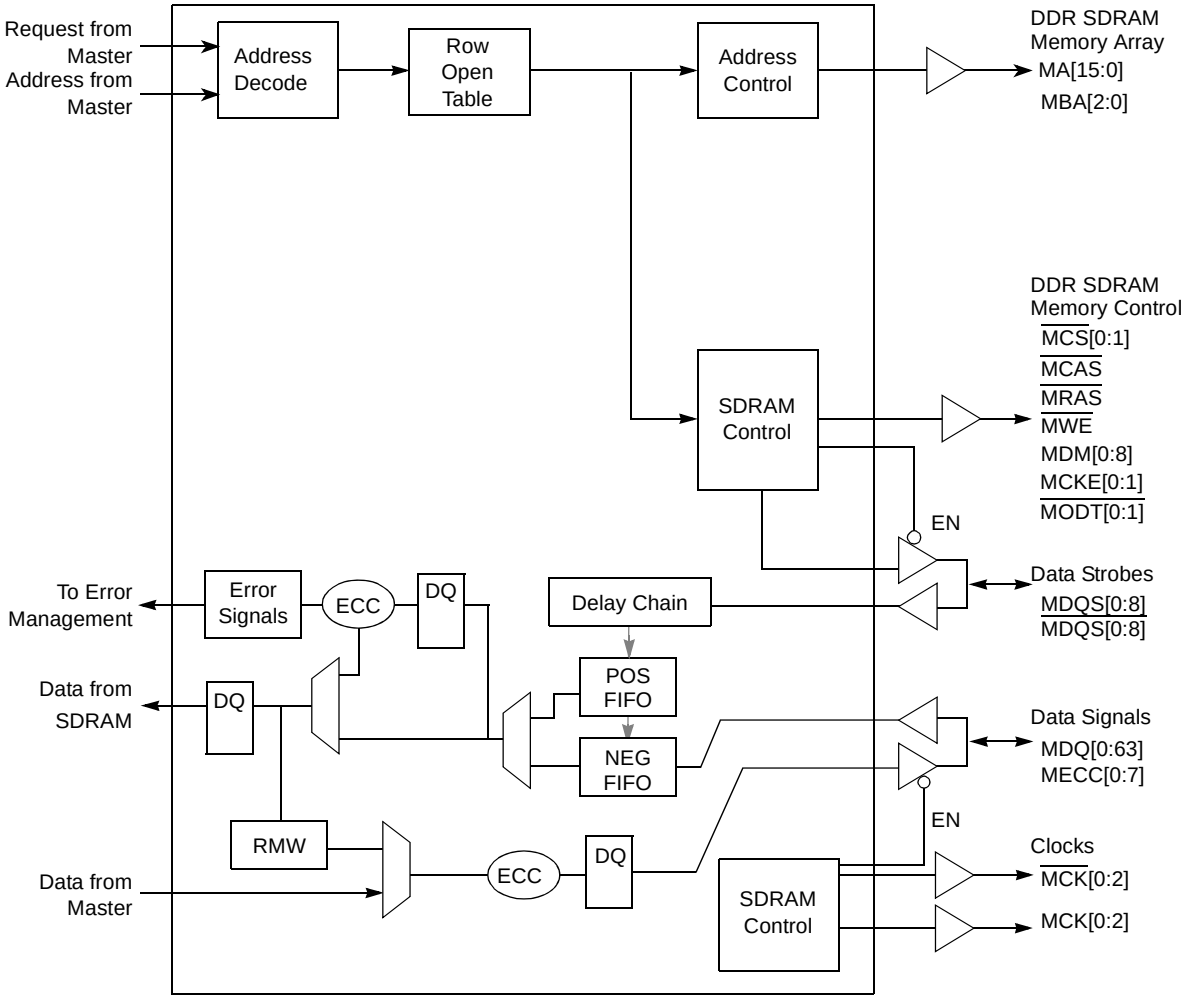


Figure 12-2. DDR Memory Controller Block Diagram

The memory interface supports as many as two physical banks of 64-/72-bit wide or 32-/40bit bit wide memory. Bank sizes up to 2 Gbytes are supported, providing up to a maximum of 2 Gbytes of DDR main memory. Programmable parameters allow for a variety of memory organizations and timings. Optional error checking and correcting (ECC) protection is provided for the DDR SDRAM data bus. Using ECC, the DDR memory controller detects and corrects all single-bit errors within the 64- or 32-bit data bus, detects all double-bit errors within the 64- or 32- bit data bus, and detects all errors within a nibble. The controller allows as many as 16 pages to be open simultaneously. The amount of time (in clock cycles) the pages remain open is programmable with DDR_SDRAM_INTERVAL[BSTOPRE].

Read and write accesses to memory are burst oriented; accesses start at a selected location and continue for a programmed number of higher locations (4 or 8) in a programmed sequence. Accesses to closed pages start with the registration of an ACTIVE command followed by a READ or WRITE. Accessing open pages does not require an ACTIVE command. The address bits registered coincident with the activate command specifies the logical bank and row to be accessed. The address coincident with the READ or WRITE command specify the logical bank and starting column for the burst access.

The data interface is source synchronous, meaning whatever sources the data also provides a clocking signal to synchronize data reception. These bidirectional data strobes (MDQS[0:8]) are inputs to the controller during reads and outputs during writes. The DDR SDRAM specification requires the data strobe signals to be centered within the data tenure during writes and to be offset by the controller to the center of the data tenure during reads. This delay is implemented in the controller for both reads and writes. When ECC is enabled, 1 clock cycle is added to the read path to check ECC and correct single-bit errors. ECC generation does not add a cycle to the write path.

The address and command interface is also source synchronous, although 1/8 cycle adjustments are provided for adjusting the clock alignment. **Figure 12-3** shows an example DDR SDRAM configuration with four logical banks.

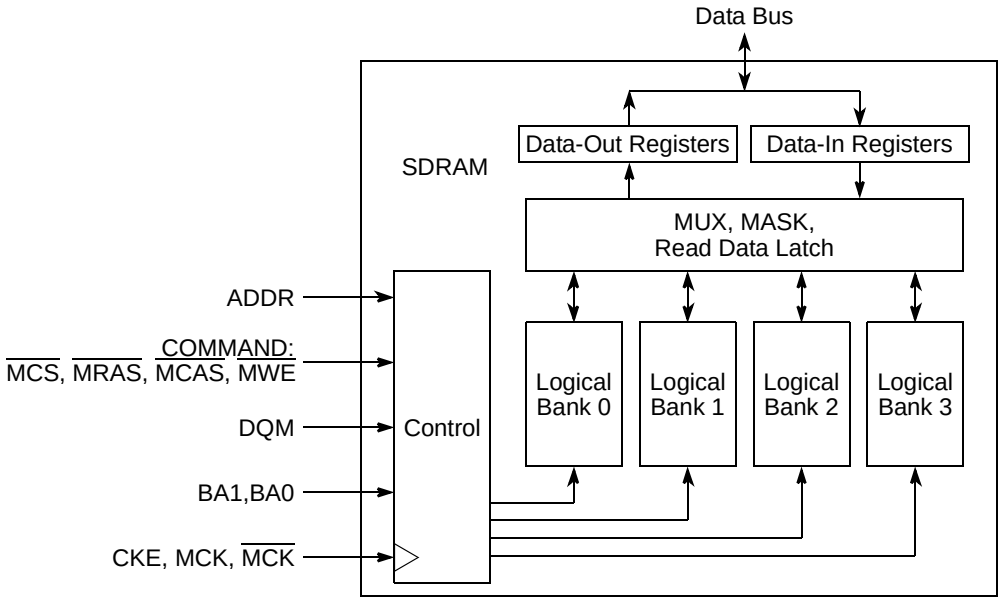


Figure 12-3. Typical Dual Data Rate SDRAM Internal Organization

Figure 12-4 shows some typical signal connections.

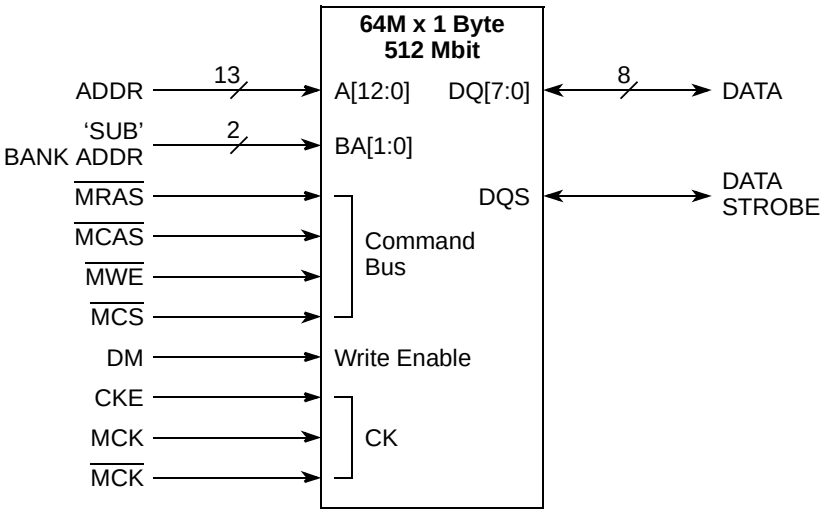
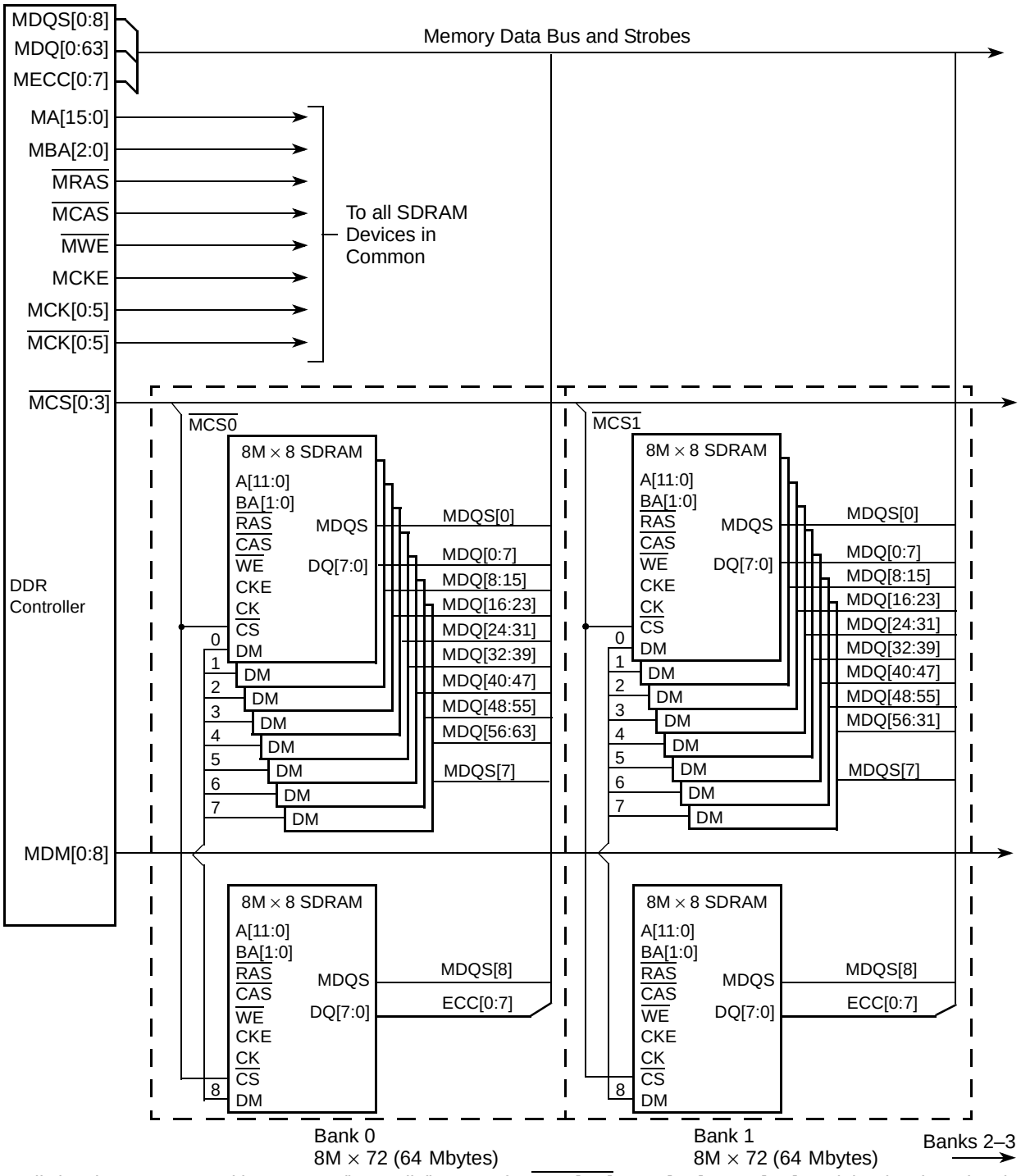


Figure 12-4. Typical DDR SDRAM Interface Signals

Figure 12-5 shows an example DDR SDRAM configuration with two physical banks each comprised of nine 8M × 8 DDR modules for a total of 128 Mbytes of system memory. One of the nine modules is used for the memory ECC checking function. Certain address and control lines may require buffering. Analysis of the device AC timing specifications, desired memory operating frequency, capacitive loads, and board routing loads can assist the system designer in deciding signal buffering requirements.

The DDR memory controller drives 16 address pins, but in this example the DDR SDRAM devices use only 12 bits. **Section 12.3.14**, *Error Management* explains how the DDR memory controller handles errors.



1. All signals are connected in common (in parallel) except for $\overline{\text{MCS}}[0:1]$, MCK[0:5], MDM[0:8], and the data bus signals.
2. Each of the $\overline{\text{MCS}}[0:1]$ signals correspond with a separate physical bank of memory.
3. Buffering may be needed if large memory arrays are used.
4. MCK[0:5] may be apportioned among all memory devices. Complementary bus is not shown.

Figure 12-5. Example 64-Mbyte DDR SDRAM Configuration

12.3.1 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 and three logical bank select signals support device densities from 512 Mbits to 4 Gbits. Two chip select ($\overline{CS}$) signals support up to one 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 512 Mbytes to 2 Gbytes. The physical banks can be constructed using x8 or x16 memory devices. The memory technologies supported are 512 Mbits, 1 Gbit, 2 Gbits, and 4 Gbits. Nine data qualifier (DQM) signals provide byte selection for memory accesses.

Note: An 8-bit DDR SDRAM device has a DQM signal and eight data signals (DQ[0:7]). A 16-bit DDR SDRAM device has two DQM signals associated with specific halves of the 16 data signals (DQ[0:7] and DQ[8:15]).

When ECC is enabled, all memory accesses are performed on double-word boundaries (that is, all DQM signals are set simultaneously). However, when ECC is disabled, the memory system uses the DQM signals for byte lane selection.

12-1 shows the DDR memory controller's relationships between data byte lane0–7, MDM[0:7], MDQS[0:7], and MDQ[0:63] when DDR SDRAM memories are used with x8 or x16 devices.

Table 12-1. Byte Lane to Data Relationship

Data Byte Lane	Data Bus Mask	Data Bus Strobe	Data Bus 64-Bit Mode
0 (MSB)	MDM[0]	MDQS[0]	MDQ[0:7]
1	MDM[1]	MDQS[1]	MDQ[8:15]
2	MDM[2]	MDQS[2]	MDQ[16:23]
3	MDM[3]	MDQS[3]	MDQ[24:31]
4	MDM[4]	MDQS[4]	MDQ[32:39]
5	MDM[5]	MDQS[5]	MDQ[40:47]
6	MDM[6]	MDQS[6]	MDQ[48:55]
7 (LSB)	MDM[7]	MDQS[7]	MDQ[56:63]

12.3.2 Supported DDR SDRAM Organizations

Although the DDR memory controller multiplexes row and column address bits onto 16 memory address signals and 3 logical bank select signals, a physical bank may be implemented with memory devices requiring fewer than 32 address bits. The physical bank may be configured to provide from 12 to 15 row address bits, plus 3 logical bank-select bits and from 8–11 column address bits. **Table 12-2** describes DDR SDRAM device configurations supported by the DDR memory controller.

Note: DDR SDRAM is limited to 30 total address bits.

Table 12-2. Supported DDR3 SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column x Sub-bank Bits	64-Bit Bank Size	Two Banks of Memory
1 Gbits	128 Mbits x 8	14 x 10 x 3	1 Gbyte	2 Gbytes
1 Gbits	64 Mbits x 16	13 x 10 x 3	512 Mbytes	1 Gbytes
2 Gbits	256Mbits x 8	15 x 10 x 3	2 Gbytes	—
2 Gbits	128Mbits x 16	14 x 10 x 3	1 Gbyte	2 Gbytes
4 Gbits	512Mbits x 8	16 x 10 x 3	—	—
4 Gbits	256Mbits x 16	15 x 10 x 3	2 Gbytes	—
Note: The MSC8157E controller supports a total of 2 Gbyte of DDR SDRAM using one or two banks of memory.				

If a transaction request is issued to the DDR memory controller and the address does not lie within any of the programmed address ranges for an enabled chip select, a memory select error is flagged. Errors are described in detail in **Section 12.3.14**, *Error Management*.

By using a memory-polling algorithm at power-on reset or by querying the JEDEC serial presence detect capability of memory modules, system firmware uses the memory-boundary registers to configure the DDR memory controller to map the size of each bank in memory. The memory controller uses its bank map to assert the appropriate $\overline{\text{MCS}}n$ signal for memory accesses according to the provided bank starting and ending addresses. The memory banks are not required to be mapped to a contiguous address space.

12.3.3 DDR SDRAM Address Multiplexing

Table 12-4 shows the address bit encodings for each DDR SDRAM configuration. The address presented at the memory controller signals MA[15:0] use MA[15] as the msb and MA[0] as the lsb. Also, MA[10] is used as the auto-precharge bit in DDR3 mode for reads and writes, so the column address can never use MA[10].

Table 12-3. DDR3 Address Multiplexing for 64-Bit Data Bus with Interleaving and Partial Array Self Refresh Disabled

Row x Col		msb	Address from Core Master																										lsb
		31	30	29	28	27	26	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	$\overline{\text{MRAS}}$		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0										
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 3	$\overline{\text{MRAS}}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0										
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{\text{MRAS}}$				12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0										
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	

Table 12-4. DDR3 Address Multiplexing for 32-Bit Data Bus with Interleaving and Partial Array Self Refresh Disabled

Row x Col		msb	Address from Core Master																										lsb
		31-30	29	28	27	26	25	24	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	$\overline{\text{MRAS}}$		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0										
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 3	$\overline{\text{MRAS}}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0										
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{\text{MRAS}}$				12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0										
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	

Chip select interleaving is supported for the memory controller, and is programmed in DDR_SDRAM_CFG[BA_INTLV_CTL]. Interleaving is supported between chip selects 0 and 1. When interleaving is enabled, the chip selects being interleaved must use the same size of memory. One extra bit in the address decode is used for the interleaving to determine which chip select to access.

Table 12-5 illustrates examples of address decode when interleaving between two chip selects.

Table 12-5. Example of Address Multiplexing for 32/64-Bit Data Bus Interleaving Between Two Banks with Partial Array Self Refresh Disabled

Row x Col		msb	Address from Core Master																										lsb
		31	30	29	28	27	26	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
14 x 10 x 3	$\overline{\text{MRAS}}$		13	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 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	

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 is swapped, and the sub-bank is decoded as the most significant portion of the DRAM address, as shown in **Table 12-6**. If chip select interleaving and PASR are enabled for a chip select, then the interleaved chip select bit is placed immediately to the left of the column decode, as shown in **Table 12-7**.

Table 12-6. DDR3 Address Multiplexing with Partial Array Self Refresh Enabled

Row x Col		msb	Address from Core Master																											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
16 x 10 x 3	$\overline{\text{MRAS}}$					15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
	MBA		2	1	0																										
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0
15 x 10 x 3	$\overline{\text{MRAS}}$						14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
	MBA			2	1	0																									
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0
14 x 10 x 3	$\overline{\text{MRAS}}$							13	12	11	10	9	8	7	6	5	4	3	2	1	0										
	MBA				2	1	0																								
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0
13 x 10 x 3	$\overline{\text{MRAS}}$								12	11	10	9	8	7	6	5	4	3	2	1	0										
	MBA					2	1	0																							
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0

Table 12-7. Example of Address Multiplexing for 64-bit Data Bus Interleaving Between Two Banks with Partial Array Self Refresh Enabled

Row x Col		msb		Address from Core Master																								lsb
		31	30	29	28	27	26	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
14 x 10 x3	$\overline{\text{MRAS}}$					13	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 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	

12.3.4 JEDEC Standard DDR SDRAM Interface Commands

This section describes the commands and timings the controller uses when operating in DDR3 mode. All read or write accesses to DDR SDRAM are performed by the DDR memory controller using JEDEC standard DDR SDRAM interface commands. The SDRAM device samples command and address inputs on rising edges of the memory clock; data is sampled using both the rising and falling edges of DQS. Data read from the DDR SDRAM is also sampled on both edges of DQS.

The following DDR SDRAM interface commands (summarized in **Table 12-8**) are provided by the DDR controller. All actions for these commands are described from the perspective of the SDRAM device.

- Row activate—Latches row address and initiates memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored by a precharge command before another row activate occurs.
- Precharge—Restores data from the sense amplifiers to the appropriate row. Also initializes the sense amplifiers in preparation for reading another row in the memory array, (performing another activate command). Precharge must occur after read or write, if the row address changes on the next open page mode access.
- Read—Latches column address and transfers data from the selected sense amplifier to the output buffer as determined by the column address. During each succeeding clock edge, additional data is driven without additional read commands. The amount of data transferred is determined by the burst size, which is set to 8 by the DDR controller.
- 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 8 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 being read, the row is automatically rewritten in the memory array. All logical banks must be in a precharged state before executing a refresh. The memory controller also supports posted refreshes, where several refreshes may be executed at once, and the refresh interval may be extended.
- Mode register set (for configuration)—Allows setting of DDR SDRAM options. These options are: $\overline{\text{MCAS}}$ latency, additive latency (for DDR3), write recovery (for DDR3), burst type, and burst length. $\overline{\text{MCAS}}$ latency may 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}, etc.). Burst type is always sequential. Although some SDRAMs provide burst lengths of 1, 2, 4, 8, and page size, this memory controller supports a burst length of 4. A burst length of 8 is supported for DDR3 memory only. The mode register set command is performed by the DDR memory controller during system initialization. Parameters such as mode register data, $\overline{\text{MCAS}}$ 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[MEM_EN] is set. If DDR_SDRAM_CFG[BI] is set to bypass the automatic initialization, then the MODE registers can be configured through software via use of the DDR_SDRAM_MD_CNTL register.
- Self refresh (for long periods of standby)—Used when the device is in standby for very long periods of time. Automatically generates internal refresh cycles to keep the data in all memory banks refreshed. Before execution of this command, the DDR controller will place all logical banks in a precharged state.

Table 12-8. DDR SDRAM Command Table

Operation	CKE Prev.	CKE Current	$\overline{\text{MCS}}$	$\overline{\text{MRAS}}$	$\overline{\text{MCAS}}$	$\overline{\text{MWE}}$	MBA	MA10	MA
Activate	H	H	L	L	H	H	Logical bank select	Row	Row
Precharge select logical bank	H	H	L	L	H	L	Logical bank select	L	X
Precharge all logical banks	H	H	L	L	H	L	X	H	X
Read	H	H	L	H	L	H	Logical bank select	L	Column
Read with auto-precharge	H	H	L	H	L	H	Logical bank select	H	Column
Write	H	H	L	H	L	L	Logical bank select	L	Column
Write with auto-precharge	H	H	L	H	L	L	Logical bank select	H	Column
Mode register set	H	H	L	L	L	L	Opcode	Opcode	Opcode and mode

Table 12-8. DDR SDRAM Command Table (Continued)

Operation	CKE Prev.	CKE Current	$\overline{\text{MCS}}$	$\overline{\text{MRAS}}$	$\overline{\text{MCAS}}$	$\overline{\text{MWE}}$	MBA	MA10	MA
Auto refresh	H	H	L	L	L	H	X	X	X
Self refresh	H	L	L	L	L	H	X	X	X

12.3.5 DDR SDRAM Interface Timing

The DDR memory controller supports eight-beat bursts to SDRAM. For single-beat reads, the DDR memory controller performs an eight-beat burst read, but ignores the last seven beats. Single-beat writes are performed by masking the last seven beats of the eight beat burst using the data mask MDM[0:8]. If ECC is disabled, writes smaller than 64 bits 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 eight beats are not terminated early even if some data is irrelevant.

To accommodate available memory technologies across a wide spectrum of operating frequencies, the DDR memory controller allows the setting of the intervals defined in **Table 12-9** with granularity of one memory clock cycle, except for CASLAT, which can be programmed with 1/2 clock granularity.

Table 12-9. DDR SDRAM Interface Timing Intervals

Timing Intervals	Definition
ACTTOACT	The number of clock cycles from a bank-activate command until another bank-activate command within a physical bank. This interval is listed in the AC specifications of the SDRAM as t_{RRD}.
ACTTOPRE	The number of clock cycles from an activate command until a precharge command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RAS}.
ACTTORW	The number of clock cycles from an activate command until a read or write command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RCD}.
BSTOPRE	The number of clock cycles to maintain a page open after an access. The page open duration counter is reloaded with BSTOPRE each time the page is accessed (including page hits). When the counter expires, the open page is closed with a SDRAM precharge bank command as soon as possible.
CASLAT	Used in conjunction with additive latency to obtain the READ latency. The number of clock cycles between the registration of a READ command by the SDRAM and the availability of the first piece of output data. If a READ command is registered at clock edge n, and the read latency is m clocks, the data is available nominally coincident with clock edge $n + m$.
PRETOACT	The number of clock cycles from a precharge command until an activate or a refresh command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RP}.
REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR_SDRAM_CFG_2[NUM_PR], some number of rows are refreshed in each SDRAM bank during each refresh cycle. The value of REFINT depends on the specific SDRAMs used and the frequency of the interface as t_{RP}.

Table 12-9. DDR SDRAM Interface Timing Intervals (Continued)

Timing Intervals	Definition
REFREC	The number of clock cycles from the refresh command until an activate command is allowed. This can be calculated by referring to the AC specification of the SDRAM device. The AC specification indicates a maximum refresh to activate interval in nanoseconds. This field is also combined with TIMING_CFG_3[EXT_REFREC]
WR_DATA_DELAY	Provides different options for the timing between a write command and the write data strobe. This allows write data to be sent later than the nominal time to meet the SDRAM timing requirement between the registration of a write command and the reception of a data strobe associated with the write command. The specification dictates that the data strobe may not be received earlier than 75% of a cycle, or later than 125% of a cycle, from the registration of a write command. This parameter is not defined in the SDRAM specification. It is implementation-specific, defined for the DDR memory controller in TIMING_CFG_2.
WRREC	The number of clock cycles from the last beat of a write until a precharge command is allowed. This interval, write recovery time, is listed in the AC specifications of the SDRAM as t_{WR}.
WRTORD	Last write pair to read command. Controls the number of clock cycles from the last write data pair to the subsequent read command to the same bank as t_{WTR}.

The value of the parameters listed in **Table 12-9** (in whole clock cycles) must be configured by boot code at system start-up (in the TIMING_CFG_0, TIMING_CFG_1, TIMING_CFG_2, and TIMING_CFG_3 registers as described in **Section 12.5.5**, *DDR SDRAM Timing Configuration Register 0 (TIMING_CFG_0)*, **Section 12.5.6**, *DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1)*, **Section 12.5.7**, *DDR SDRAM Timing Configuration Register 2 (TIMING_CFG_2)*, and **Section 12.5.4**, *DDR SDRAM Timing Configuration 3 Register (TIMING_CFG_3)*) and be kept in the DDR memory controller configuration register space.

The following figures show SDRAM timing for various types of accesses. System software is responsible (at reset) for optimally configuring SDRAM timing parameters. The programmable timing parameters apply to both read and write timing configuration. The configuration process must be completed and the DDR SDRAM initialized before any accesses to SDRAM are attempted.

Figure 12-6 through **Figure 12-8** show DDR SDRAM timing for various types of accesses; see **Figure 12-6** for a single-beat read operation, **Figure 12-7** for a single-beat write operation, and **Figure 12-8** for a burst-write operation. Note that all signal transitions occur on the rising edge of the memory bus clock and that single-beat read operations are identical to burst-reads. These figures assume the CLK_ADJUST is set to 1/2 DRAM cycle, an additive latency of 0 DRAM cycles is used, and the write latency is 1 DRAM cycle.

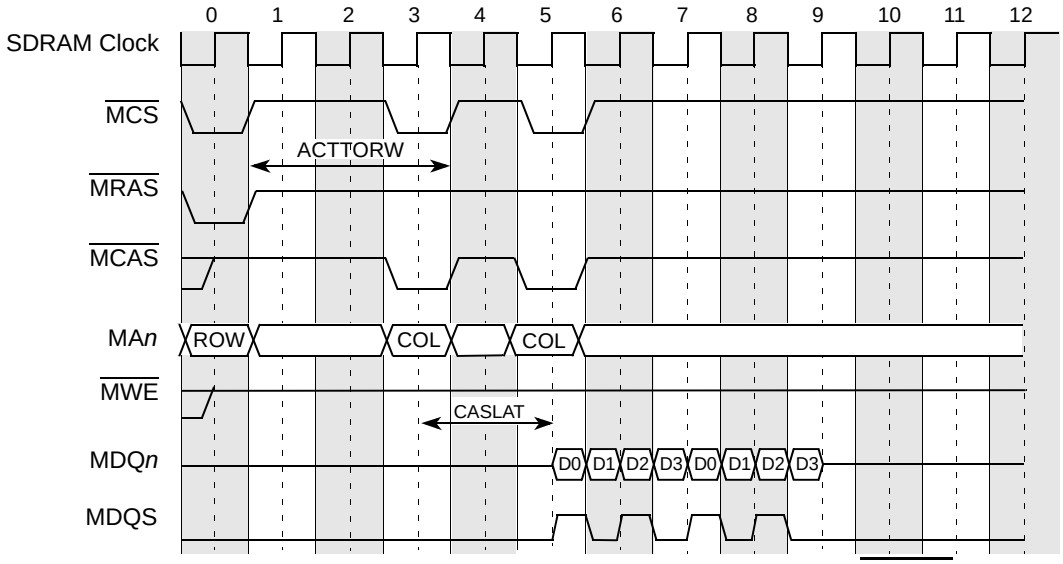


Figure 12-6. DDR SDRAM Burst Read Timing—ACTTORW = 3, MCAS Latency = 2

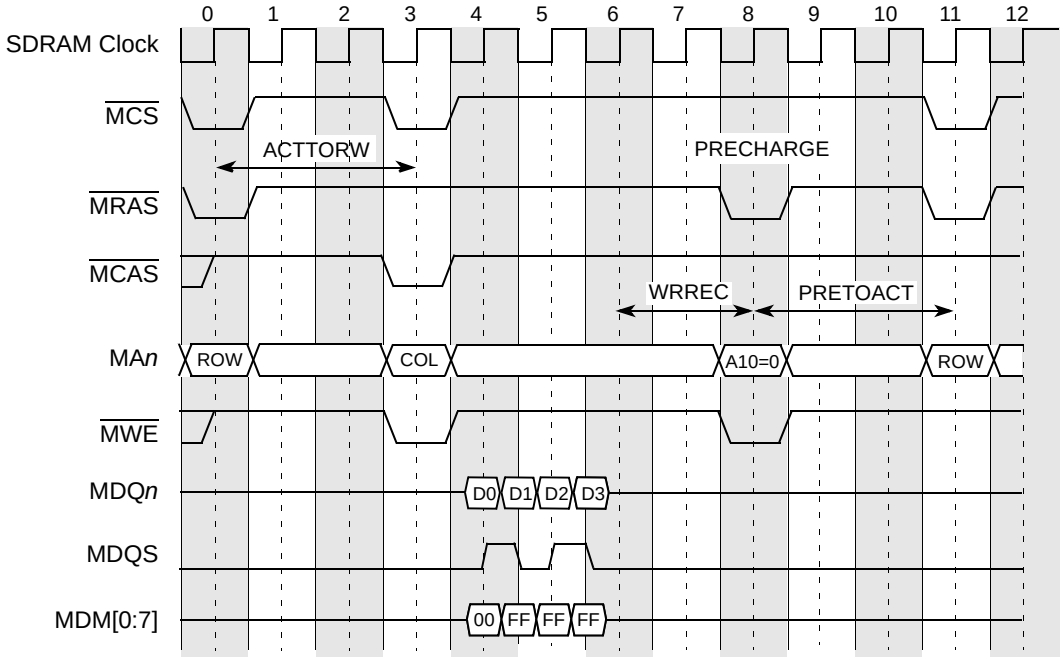


Figure 12-7. DDR SDRAM Single-Beat (64 bit) Write Timing—ACTTORW = 3

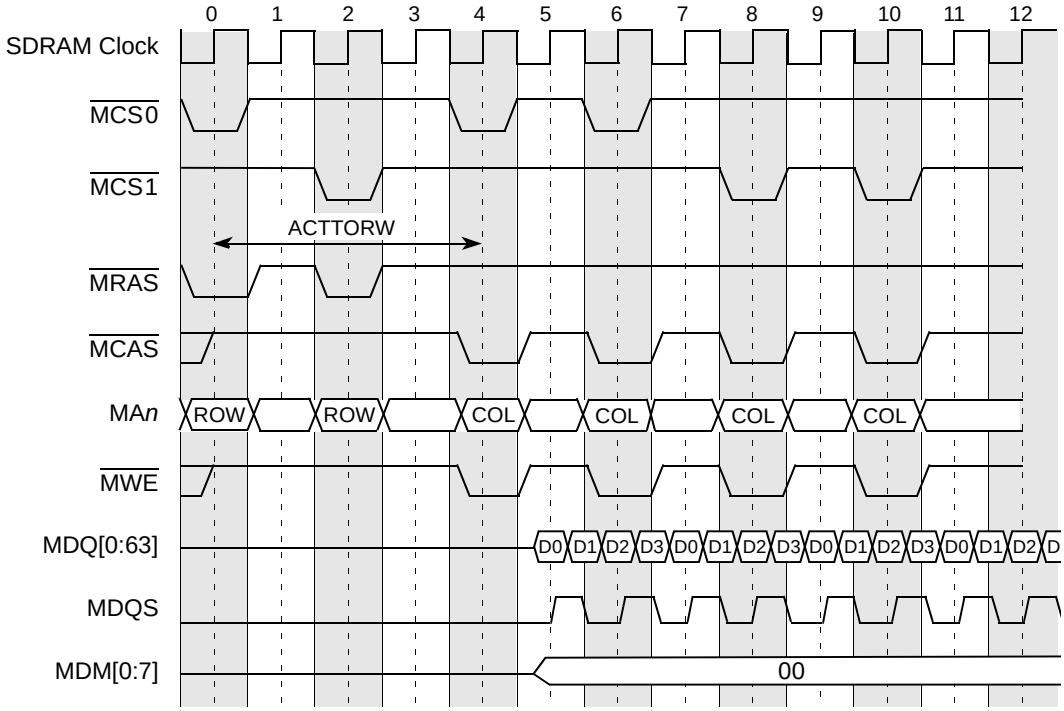


Figure 12-8. DDR SDRAM 4-Beat Burst Write Timing—ACTTORW = 4

12.3.6 Clock Distribution

Use the following recommendations for clock distribution:

- If running with many devices, zero-delay PLL clock buffers, JEDEC-JESD82 standard, should be used. These buffers were designed for DDR applications.
- A 72 bit × 2 Gbytes DDR bank has 9-byte-wide DDR chips, In this case, each MCK/$\overline{\text{MCK}}$ signal pair should drive exactly three devices.
- PCB traces for DDR clock signals should be short, all on the same layer, and of equal length and loading.
- DDR SDRAM manufacturers provide detailed information on PCB layout and termination issues.

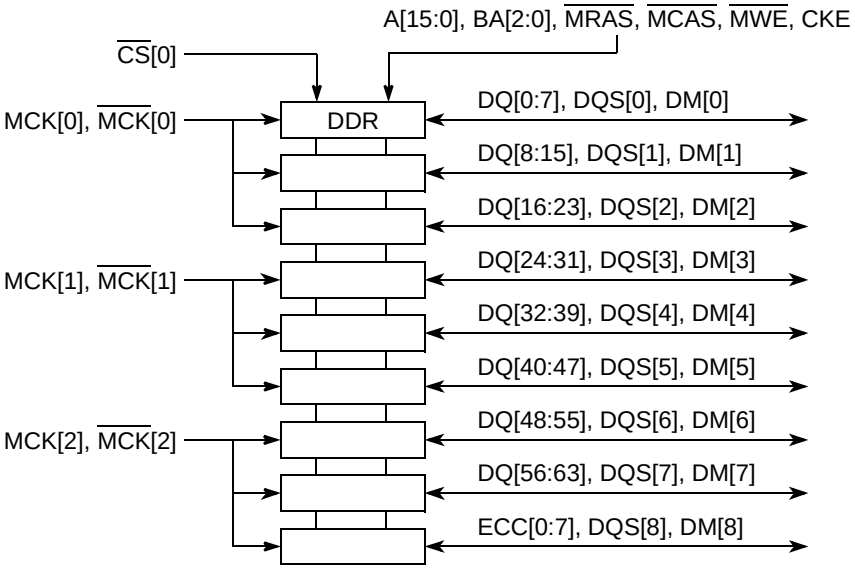


Figure 12-9. DDR SDRAM Clock Distribution Example for x8 DDR SDRAMs

12.3.7 DDR SDRAM Mode-Set Command Timing

The DDR memory controller transfers the mode register set commands to the SDRAM array, and it uses the setting of TIMING_CFG_0[MRS_CYC] for the Mode Register Set cycle time.

Figure 12-10 shows the timing of the mode-set command. The first transfer corresponds to the ESDMODE code; the second corresponds to SDMODE. The Mode Register Set cycle time is set to 2 DRAM cycles.

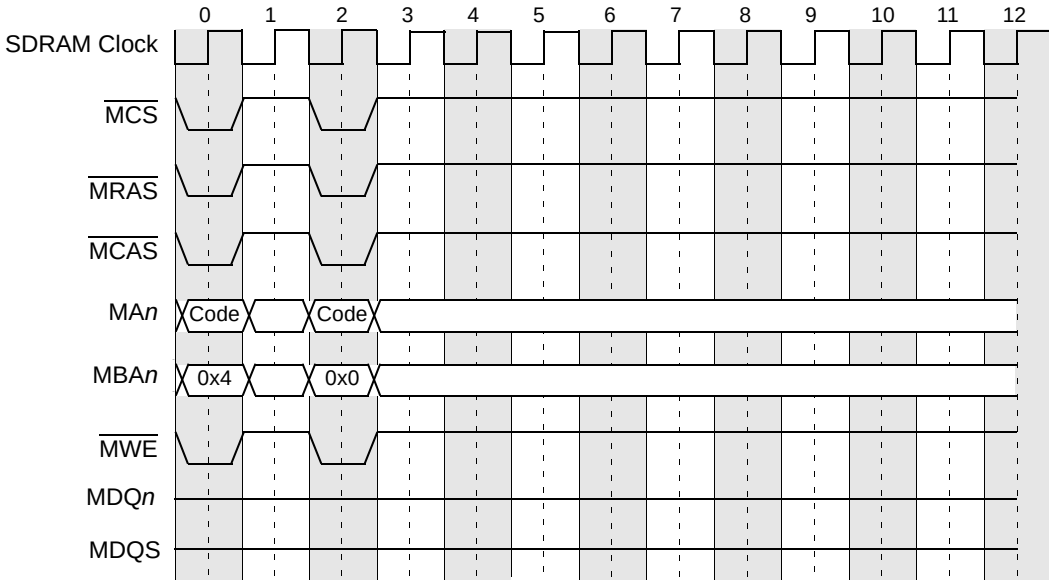


Figure 12-10. DDR SDRAM Mode-Set Command Timing

12.3.8 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 12-11 shows the registered DDR SDRAM DIMM single-beat write timing.

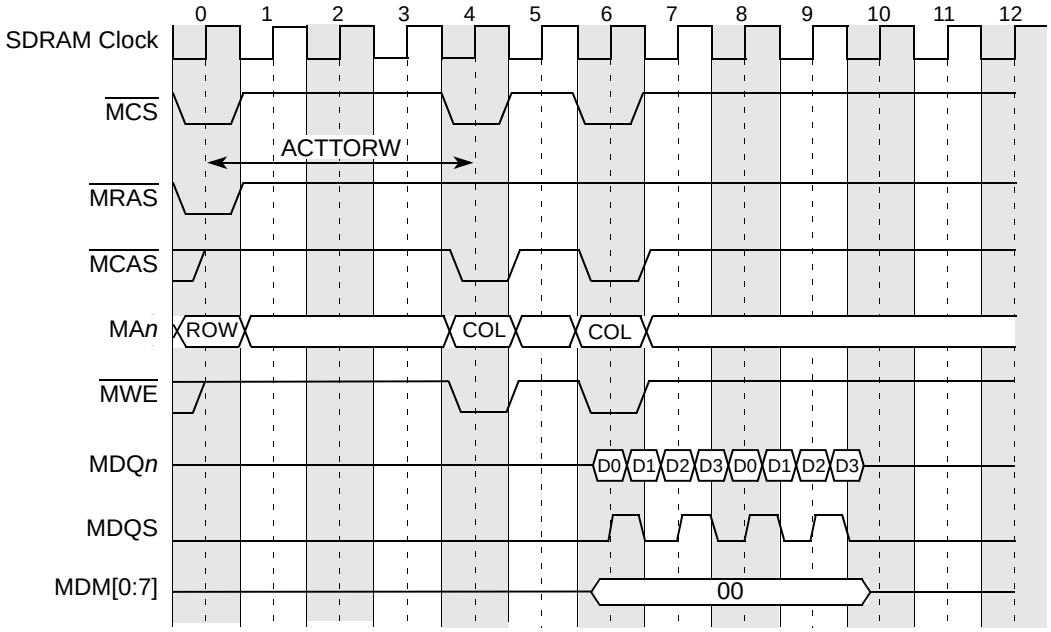


Figure 12-11. Registered DDR SDRAM DIMM Burst Write Timing

Use the following steps if RC10 must be modified to allow higher frequency operation for registered DIMMS (RDIMMs). These steps replace the Automatic DIMM Register Control Word update sequence enabled by DDR_SDRAM_CFG_2[RCW_EN]. The Automatic DIMM Register Control Word update sequence can be used if RC10 does not require an update.

1. Write all DDR registers, but leave DDR_SDRAM_CFG[MEM_EN] cleared. Note that address parity should also be enabled if writing to the RCWs.
2. Disable automatic CPO by ensuring that TIMING_CFG_2[CPO] is not set to all 1s.
3. Write 0x00000400 to the register at address 0xFFF20F08.
4. Disable ZQ calibration by clearing DDR_ZQ_CNTL[ZQ_EN].

5. Disable write levelling by clearing DDR_WRLVL_CNTL[WRLVL_EN].
6. Disable the automatic RCW sequence by clearing DDR_SDRAM_CFG_2[RCW_EN].
7. Ensure that DDR_SDRAM_CFG_2[D_INIT] is cleared and the memory test is disabled (DDR_MTCR[MT_EN] is cleared).
8. Write 0x00000015 to the register at address 0xFFF20F30.
9. Write 0x24000000 to the register at address 0xFFF20F54.
10. Disable refreshes by clearing DDR_SDRAM_INTERVAL[REFINT].
11. Set DDR_SDRAM_CFG[BI].
12. Enable the DDR controller by setting DDR_SDRAM_CFG[MEM_EN].
13. Poll until bit 1 of the register at address 0xFFF20F04 is asserted by hardware. This indicates that the controller is idle.
14. Use the DDR_SDRAM_MD_CNTL register to write RC10 for any DIMMs that are used. Note that if two registered DIMM modules are used, this requires two separate writes to the DDR_SDRAM_MD_CNTL register. Ensure that either a decoding of 0b100 or 0b101 is used for the CS_SEL field, and WRCW should be set by the controller. After each write to the DDR_SDRAM_MD_CNTL register, DDR_SDRAM_MD_CNTL[MD_EN] should be polled until it is cleared by hardware.
15. Wait for t_{STAB} as defined by the register specifications.
16. Clear DDR_SDRAM_CFG[MEM_EN].
17. Clear DDR_SDRAM_CFG[MEM_EN].
18. Restore all registers back to the original settings. This may also include enabling DDR_SDRAM_CFG_2[RCW_EN] if more RCWs will be modified. If this bit is set, ensure that DDR_SDRAM_RCW_2[RCW10] is programmed to provide the same value as that programmed via the DDR_SDRAM_MD_CNTL register in step 14.
19. Set DDR_SDRAM_CFG[MEM_EN].
20. The DDR controller should now be operational.

12.3.9 DDR SDRAM Write Timing Adjustments

The DDR memory controller facilitates system design flexibility by providing a write timing adjustment parameter, write data delay, (TIMING_CFG_2[WR_DATA_DELAY]) for data and DQS. The DDR SDRAM specification requires DQS be received no sooner than 75% of an SDRAM clock period—and no later than 125% of a clock period—from the capturing clock edge of the command/address at the SDRAM. The WR_DATA_DELAY parameter may be used to meet this timing requirement for a variety of system configurations, ranging from a system with one DIMM to a fully populated system with two DIMMs.
TIMING_CFG_2[WR_DATA_DELAY] specifies how much to delay the launching of DQS and

data from the first clock edge occurring one SDRAM clock cycle after the command is launched. The delay increment step sizes are in 1/4 SDRAM clock periods starting with the default value of 0. **Figure 12-12** shows the use of the WR_DATA_DELAY parameter.

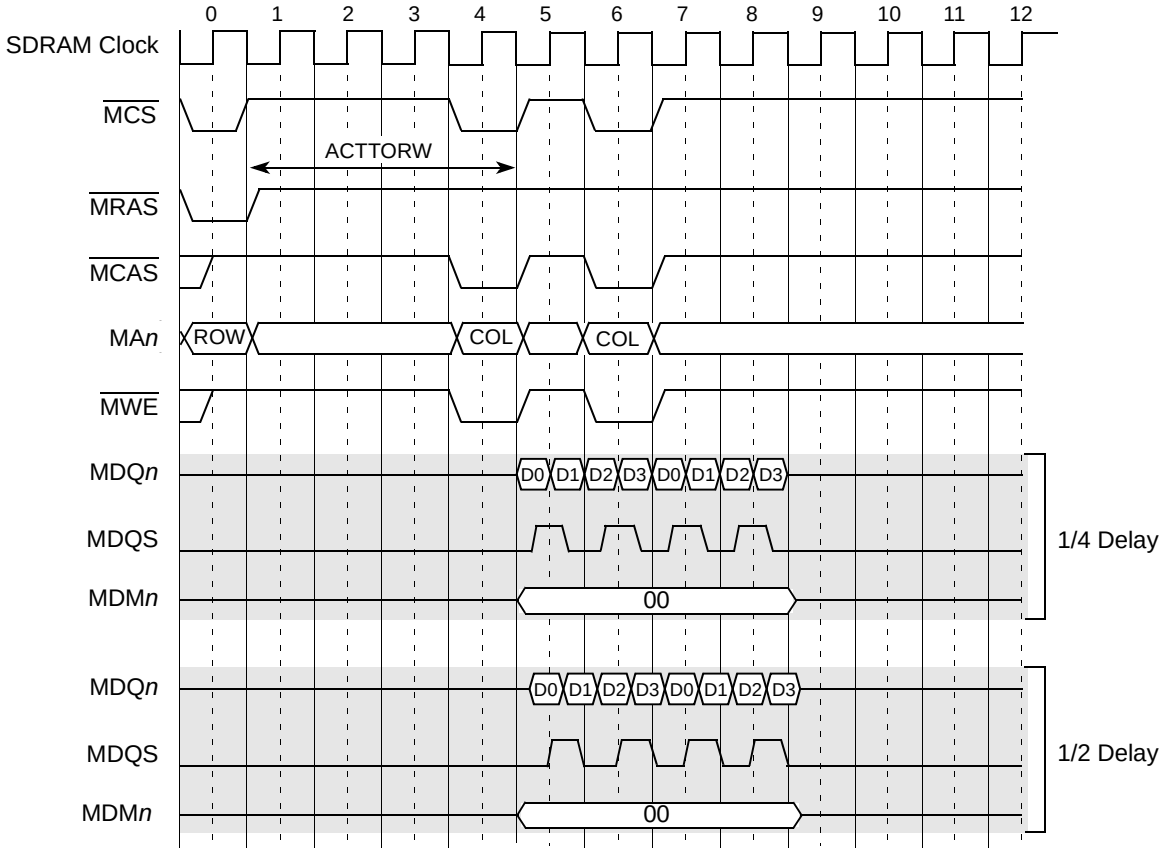


Figure 12-12. Write Timing Adjustments Example for Write Latency = 1

12.3.10 DDR SDRAM Refresh

The DDR memory controller supports auto-refresh and self-refresh. Auto refresh is used during normal operation and is controlled by the DDR_SDRAM_INTERVAL[REFINT] value; self-refresh is used only when the DDR memory controller is set to enter a sleep power management state. The REFINT value, which represents the number of memory bus clock cycles between refresh cycles, must allow for possible outstanding transactions to complete before a refresh request is sent to the memory after the REFINT value is reached. If a memory transaction is in progress when the refresh interval is reached, the refresh cycle waits for the transaction to complete. In the worst case, the refresh cycle must wait the number of bus clock cycles required by the longest programmed access. To ensure that the latency caused by a memory transaction does not violate the device refresh period, it is recommended that the programmed value of REFINT be less than that required by the SDRAM.

When a refresh cycle is required, the DDR memory controller does the following:

1. Completes all current memory requests.
2. Closes all open pages with a PRECHARGE-ALL command to each DDR SDRAM bank with an open page (as indicated by the row open table).
3. Issues one or more auto-refresh commands to each DDR SDRAM bank (as identified by its chip select) to refresh one row in each logical bank of the selected physical bank.

The auto-refresh commands are staggered across the two possible banks to reduce the system instantaneous power requirements. Two sets of auto refresh commands will be issued on consecutive cycles when the memory is fully populated with two DIMMs. The initial PRECHARGE-ALL commands are also staggered in two groups for convenience. It is important to note that when entering self-refresh mode, only one refresh command is issued simultaneously to all physical banks. For this entire refresh sequence, no cycle optimization occurs. After the refresh sequence completes, any pending memory request is initiated after an inactive period specified by TIMING_CFG_1 [REFREC] and TIMING_CFG_3[EXT_REFREC]. In addition, posted refreshes are supported to allow the refresh interval to be set to a larger value.

12.3.10.1 DDR SDRAM Refresh Timing

Refresh timing for the DDR SDRAM is controlled by the programmable timing parameter TIMING_CFG_1 [REFREC], which specifies the number of memory bus clock cycles from the refresh command until a logical bank activate command is allowed. The DDR memory controller implements bank staggering for refreshes, as shown in **Figure 12-13** (TIMING_CFG_1 [REFREC] = 10 in this example).

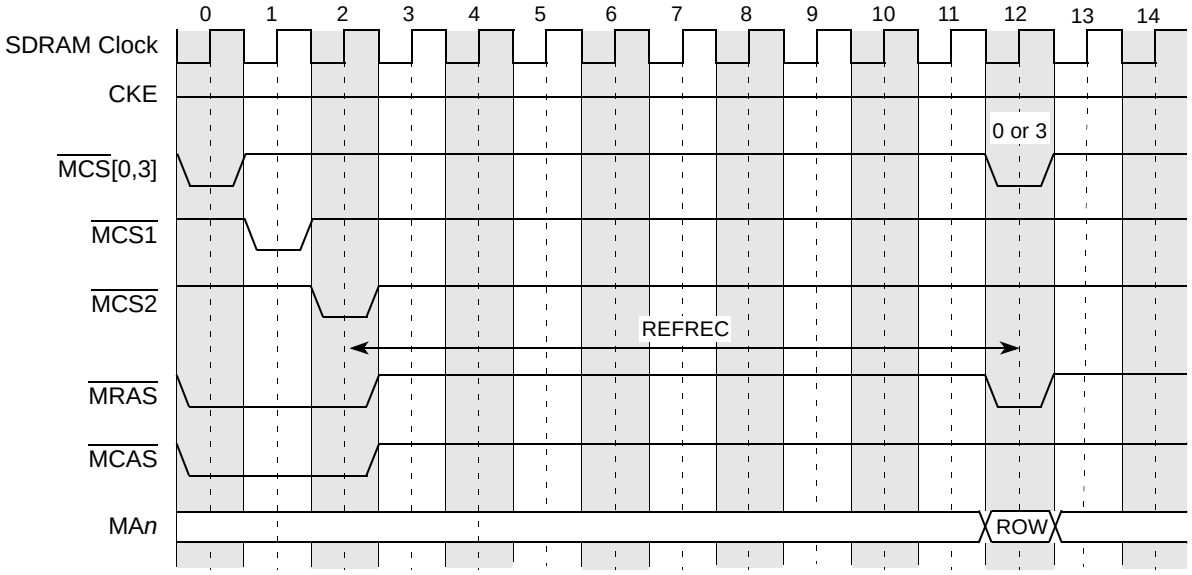


Figure 12-13. DDR SDRAM Bank Staggered Auto Refresh Timing

System software is responsible for optimal configuration of TIMING_CFG_1 [REFREC] and TIMING_CFG_3[EXT_REFREC] at reset. Configuration must be completed before DDR SDRAM accesses are attempted.

12.3.10.2 DDR SDRAM Refresh and Power-Saving Modes

In full-on mode, the DDR memory controller supplies the normal auto refresh to SDRAM. In sleep mode, the DDR memory controller can be configured to take advantage of self-refreshing SDRAMs or to provide no refresh support. Self-refresh support is enabled with the SREN memory control parameter.

Table 12-10 summarizes the refresh types available in each power-saving mode.

Table 12-10. DDR SDRAM Power-Saving Modes Refresh Configuration

Power Saving Mode	Refresh Type	SREN
Sleep	Self	1
	None	—

Note that in the absence of refresh support, system software must preserve DDR SDRAM data (such as by copying the data to disk) before entering the power-saving mode.

The dynamic power-saving mode uses the CKE DDR SDRAM pin to dynamically power down when there is no system memory activity. The CKE pin is negated when both of the following conditions are met:

- No memory refreshes are scheduled
- No memory accesses are scheduled

CKE is reasserted when a new access or refresh is scheduled or the dynamic power mode is disabled. This mode is controlled with DDR_SDRAM_CFG[DYN_PWR_MGMT].

Dynamic power management mode offers tight control of the memory system's power consumption by trading power for performance through the use of CKE. Powering up the DDR SDRAM when a new memory reference is scheduled causes an access latency penalty, depending on whether active or precharge powerdown is used, along with the settings of TIMING_CFG_0[ACT_PD_EXIT] and TIMING_CFG_0[PRE_PD_EXIT]. A penalty of 1 cycle is shown in **Figure 12-14**.

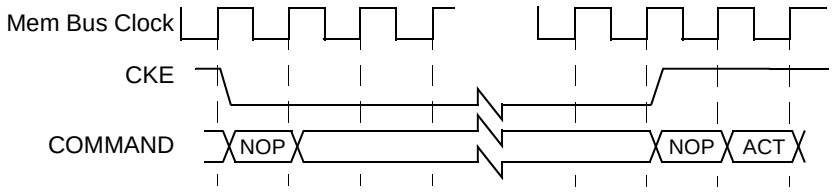


Figure 12-14. DDR SDRAM Power-Down Mode

12.3.10.3 Self-Refresh in Sleep Mode

The entry and exit timing for self-refreshing SDRAMs is shown in **Figure 12-15** and **Figure 12-16**.

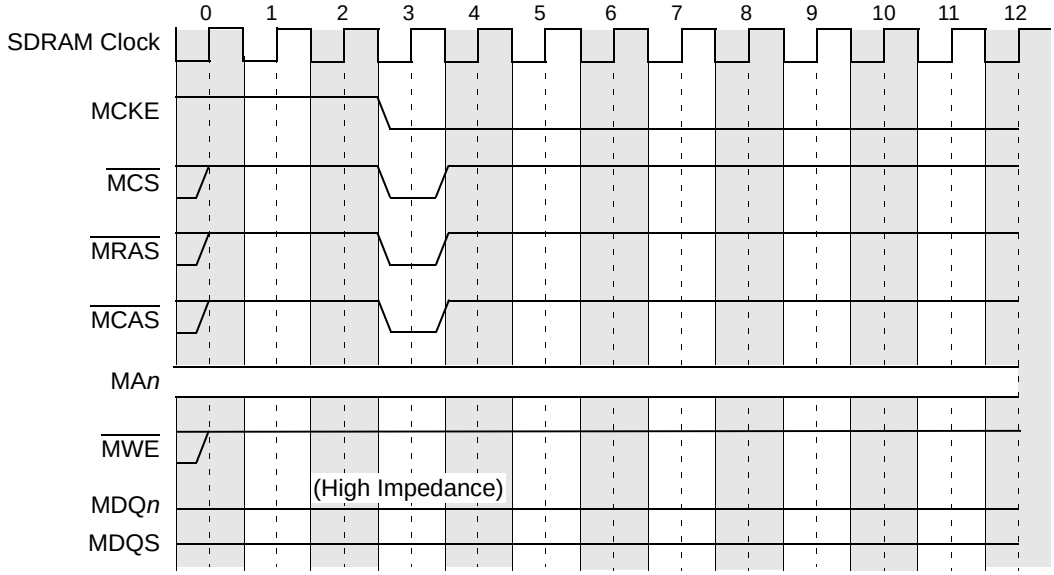


Figure 12-15. DDR SDRAM Self-Refresh Entry Timing

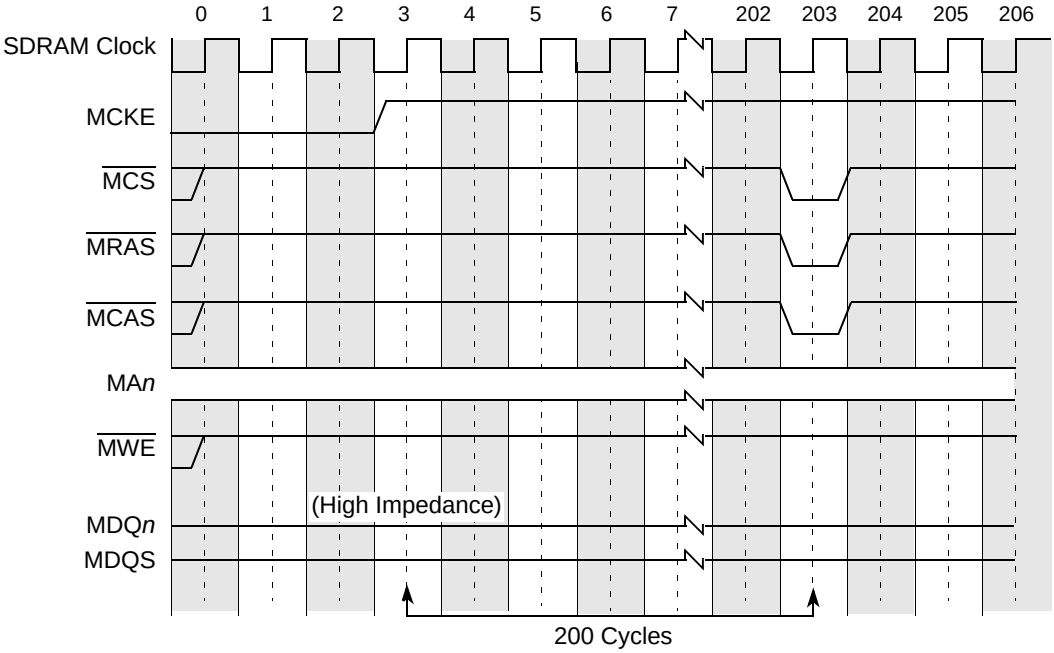


Figure 12-16. DDR SDRAM Self-Refresh Exit Timing

12.3.11 DDR Data Beat Ordering

Transfers to and from memory are always performed in eight-beat bursts (eight beats = 64 bytes when a 64-bit bus is used). For transfer sizes other than eight beats, the data transfers are still operated as eight-beat bursts. If ECC is enabled and either the access is not 64-bit aligned or the size is not a multiple of a 64 bits, a full read-modify-write is performed for a write to SDRAM. If ECC is disabled or both the access is 64-bit aligned with a size that is a multiple of a 64-bits, the data masks (MDM[0:8] (MDM[0:4] for 32-bit bus) can be used to prevent the writing of unwanted data to SDRAM. The DDR memory controller also uses data masks to prevent all unintended full 64 bits from writing to SDRAM. For example, if a write transaction is desired with a size of 64 bits (8 bytes), then the second to seventh beats of data are not written to DRAM.

All writes for DDR3 mode are aligned to beat 0 of the DRAM.

12.3.12 Page Mode and Logical Bank Retention

The DDR memory controller supports an open/closed page mode with an allowable open page for each logical bank of DRAM used. In closed page mode for DDR SDRAMs, the DDR memory controller uses the SDRAM auto-precharge feature, which allows the controller to indicate that the page must be automatically closed by the DDR SDRAM after the READ or WRITE access. This is performed by using MA[10] of the address during the COMMAND phase of the access to enable auto-precharge. Auto-precharge is non-persistent in that it is either enabled or disabled for each individual READ or WRITE command. It can, however, be enabled or disabled separately for each chip select.

When the DDR memory controller operates in open page mode, it retains the currently active SDRAM page by not issuing a precharge command. The page remains opens until one of the following conditions occurs:

- Refresh interval is met.
- The user-programmable DDR_SDRAM_INTERVAL[BSTOPRE] value is exceeded.
- There is a logical bank row collision with another transaction that must be issued.

Page mode can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, using page mode can save two to three clock cycles for subsequent burst accesses that hit in an active page. Also, better performance can be obtained by using more banks, especially in systems which use many different channels. Page mode is disabled by clearing DDR_SDRAM_INTERVAL[BSTOPRE] or setting CSn_CONFIG[AP_nEN].

12.3.13 Error Checking and Correcting (ECC)

The DDR memory controller supports error checking and correcting (ECC) for the data path between the core master and system memory. The memory detects all double-bit errors, detects

all multi-bit errors within a nibble, and corrects all single-bit errors. Other errors may be detected, but are not guaranteed to be corrected or detected. Double-bit errors are always reported when error reporting is enabled. When a single-bit error occurs, the single-bit error counter register is incremented, and its value compared to the single-bit error trigger register. An error is reported when these values are equal. The single-bit error registers can be programmed such that minor memory faults are corrected and ignored, but a catastrophic memory failure generates an interrupt.

For writes that are smaller than 64 bits, the DDR memory controller performs a double-word read from system memory of the address for the write (checking for errors), and merges the write data with the data read from memory. Then, a new ECC code is generated for the merged 64 bits. The data and ECC code is then written to memory. If a multi-bit error is detected on the read, the transaction completes the read-modify-write to keep the DDR memory controller from hanging. However, the corrupt data is masked on the write, so the original contents in SDRAM remain unchanged.

The syndrome encodings for the ECC code are shown in **Table 12-11** and **Table 12-12**.

Table 12-11. DDR SDRAM ECC Syndrome Encoding

Data Bit	Syndrome Bit								Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
0	•	•						•	32			•	•				•
1	•		•					•	33			•		•			•
2	•			•				•	34	•		•		•			
3	•				•			•	35		•	•		•			
4	•	•				•			36			•	•		•		
5	•		•			•			37			•		•	•		
6	•			•		•			38	•		•		•	•		•
7	•				•	•			39		•	•		•	•		•
8	•	•					•		40			•	•			•	
9	•		•				•		41			•		•		•	
10	•			•			•		42	•		•		•		•	•
11	•				•		•		43		•	•		•		•	•
12	•	•				•	•	•	44			•	•		•	•	•
13	•		•			•	•	•	45			•		•	•	•	•
14	•			•		•	•	•	46	•		•		•	•	•	
15	•				•	•	•	•	47		•	•		•	•	•	
16		•	•					•	48		•				•	•	
17		•		•				•	49			•			•	•	
18		•			•			•	50				•		•	•	
19	•	•			•				51	•					•	•	

Table 12-11. 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 12-12. 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								•

12.3.14 Error Management

The DDR memory controller detects four different kinds of errors: training, single-bit, multi-bit, and memory select errors. The following discussion assumes all the relevant error detection, correction, and reporting functions are enabled as described in **Section 12.5.43**, *DDR SDRAM Memory Error Detect Register (ERR_DETECT)*, **Section 12.5.44**, *DDR SDRAM Memory Error Disable Register (ERR_DISABLE)*, and **Section 12.5.45**, *DDR SDRAM Memory Error Interrupt Enable Register (ERR_INT_EN)*.

Single-bit errors are counted and reported based on the ERR_SBE value. When a single-bit error is detected, the DDR memory controller does the following:

- Corrects the data
- Increments the single-bit error counter ERR_SBE[SBEC]

- Generates a critical interrupt if the counter value ERR_SBE[SBEC] equals the programmable threshold ERR_SBE[SBET]
- Completes the transaction normally

If a multi-bit error is detected for a read, the DDR memory controller logs the error and generates the machine check or critical interrupt (if enabled, as described in **Section 12.5.44**, *DDR SDRAM Memory Error Disable Register (ERR_DISABLE)*). Another error the DDR memory controller detects is a memory select error, which causes the DDR memory controller to log the error and generate a critical interrupt (if enabled, as described in **Section 12.5.43**, *DDR SDRAM Memory Error Detect Register (ERR_DETECT)*). This error is detected if the address from the memory request does not fall into any of the enabled, programmed chip select address ranges. For all memory select errors, the DDR memory controller does not issue any transactions onto the pins after the first read has returned data strobes. If the DDR memory controller is not using sample points, then a dummy transaction is issued to DDR SDRAM with the first enabled chip select. In this case, the source port on the pins is forced to 0x1F to show the transaction is not real. **Table 12-13** shows the errors with their descriptions. The final error the memory controller detects is the automatic calibration error. This error is set if the memory controller detects an error during its training sequence.

Table 12-13. Memory Controller Errors

Category	Error	Descriptions	Action	Detect Register
Notification	Single-bit ECC threshold	The number of ECC errors has reached the threshold specified in the ERR_SBE.	The error is reported via machine check or critical interrupt if enabled.	The error control register only logs read versus write, not full type
Access Error	Multi-bit ECC error	A multi-bit ECC error is detected during a read, or read-modify-write memory operation.		
	Memory select error	Read, or write, address does not fall within the address range of any of the memory banks.		

12.4 Initialization/Application Information

System software must configure the DDR memory controller, using a memory polling algorithm at system start-up, to correctly map the size of each bank in memory. Then, the DDR memory controller uses its bank map to assert the appropriate $\overline{\text{MCS}n}$ signal for memory accesses according to the provided bank depths. System software must also configure the DDR memory controller at system start-up to appropriately multiplex the row and column address bits for each bank. Refer to row-address configuration in **Section 12.5.2**, *Chip-Select x Configuration Register (CSx_CONFIG)*. Address multiplexing occurs according to these configuration bits.

At system reset, initialization software (boot code) must set up the programmable parameters in the memory interface configuration registers. See **Section 12.5**, *Memory Controller Programming Model*, on page 12-33 for detailed descriptions of the configuration registers. These parameters are shown in **Table 12-14**.

Table 12-14. Memory Interface Configuration Register Initialization Parameters

Name	Description	Parameter	Section/Page
CSn_BNDS	Chip select memory bounds	SAn EAn	12.5.1/12-35
CSn_CONFIG	Chip select configuration	CS_n_EN AP_n_EN ODT_RD_CFG ODT_WR_CFG BA_BITS_CS_n ROW_BITS_CS_n COL_BITS_CS_n	12.5.2/12-36
CS*n*_CONFIG_2	Chip select configuration 2	PASR_CFG	12.5.3/12-38
TIMING_CFG_3	Extended timing parameters for fields in TIMING_CFG_1	EXT_REFREC EXT_ACTTOPRE EXT_CASLAT CNTL_ADJ	12.5.4/12-39
TIMING_CFG_0	Timing configuration	RWT WRT RRT WWT ACT_PD_EXIT PRE_PD_EXIT ODT_PD_EXIT MRS_CYC	12.5.5/12-42
TIMING_CFG_1	Timing configuration	PRETOACT ACTTOPRE ACTTORW CASLAT REFREC WRREC ACTTOACT WRTORD	12.5.6/12-45
TIMING_CFG_2	Timing configuration	ADD_LAT CPO WR_LAT RD_TO_PRE WR_DATA_DELAY CKE_PLS FOUR_ACT	12.5.7/12-48
DDR_SDRAM_CFG	Control configuration	SREN ECC_EN RD_EN SDRAM_TYPE DYN_PWR 32_BE 8_BE DBW NCAP 2T_EN 3T_EN BA_INTLV_CTL x32_EN HSE BI	12.5.8/12-51
DDR_SDRAM_CFG_2	Control configuration	SR_IE DLL_RST_DIS DQS_CFG ODT_CFG NUM_PR AP_EN D_INIT RCW_EN MD_EN	12.5.9/12-54
DDR_SDRAM_MODE	Mode configuration	ESDMODE SDMODE	12.5.10/12-57
DDR_SDRAM_MODE_2	Mode configuration	ESDMODE2 ESDMODE3	12.5.11/12-58
DDR_SDRAM_INTERVAL	Interval configuration	REFINT BSTOPRE	12.5.13/12-61
DDR_DATA_INIT	Data initialization configuration register	INIT_VALUE	12.5.14/12-62
DDR_SDRAM_CLK_CNTL	Clock adjust	CLK_ADJUST	12.5.15/12-63
DDR_INIT_ADDR	Initialization address	INIT_ADDR	12.5.16/12-64

Table 12-14. Memory Interface Configuration Register Initialization Parameters (Continued)

Name	Description	Parameter	Section/Page
DDR_INIT_EXT_ADDR	Extended initialization address	INIT_EXT_ADDR	12.5.17/12-65
TIMING_CFG_4	Timing configuration	RWT WRT RRT WWT DLL_LOCK	12.5.18/12-66
TIMING_CFG_5	Timing configuration	RODT_ON RODT_OFF WODT_ON WODT_OFF	12.5.19/12-68
DDR_ZQ_CNTL	ZQ calibration control	ZQ_EN ZQINIT ZQOPER ZQCS	12.5.20/12-70
DDR_WRLVL_CNTL	Write leveling control	WRLVL_EN WRLVL_MRD WRLVL_ODTEN WRLVL_DQSEN WRLVL_SMPL WRLVL_WLR WRLVL_START	12.5.21/12-72
DDR_WRLVL_CNTL_2	Write leveling control	WRLVL_START_1 WRLVL_START_2 WRLVL_START_3 WRLVL_START_4	12.5.22/12-75
DDR_WRLVL_CNTL_3	Write leveling control	WRLVL_START_5 WRLVL_START_6 WRLVL_START_7 WRLVL_START_8	12.5.23/12-78
DDR_SR_CNTR	Self refresh control	SR_IT	12.5.24/12-81
DDR_SDRAM_RCW_1	Register control words configuration	RCW0 RCW1 RCW2 RCW3 RCW4 RCW5 RCW6 RCW7	12.5.25/12-82
DDR_SDRAM_RCW_2	Register control words configuration	RCW8 RCW9 RCW10 RCW11 RCW12 RCW13 RCW14 RCW15	12.5.26/12-83

Table 12-14. Memory Interface Configuration Register Initialization Parameters (Continued)

Name	Description	Parameter	Section/Page
DDRCDR_1	Driver control	DHC_EN ODT DSO_C_EN DSO_D_EN DSO_CPZ DSO_CNZ DSO_DPZ DSO_DNZ	12.5.31/12-87
DDRCDR_2	Driver control	DSO_CLK_EN DSO_CLKPZ DSO_CLKNZ	12.5.32/12-91

12.4.1 Programming Summary

Depending on the memory type used, certain fields must be programmed differently. **Table 12-15** illustrates the differences in certain fields for different memory types. Note: This table does not list all fields that must be programmed.

Table 12-15. DDR3 Programming Summary

Parameter	Description	Summary	Section/page
AP*n*_EN	Chip Select *n* Auto Precharge Enable	Can be used to place chip select *n* in auto precharge mode	12.5.2/12-36
ODT_RD_CFG	Chip Select ODT Read Configuration	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, systems with only 1 chip select will typically not use ODT when issuing reads to the memory.	12.5.2/12-36
ODT_WR_CFG	Chip Select ODT Write Configuration	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, ODT will typically be set to assert for the chip select that is getting written to (value would be set to 001).	12.5.2/12-36
ODT_PD_EXIT	ODT Powerdown Exit	Should be set to 0001 for DDR3. The powerdown times (t_{XP} and t_{XPDLL}) required for DDR3 are controlled via TIMING_CFG_0[ACT_PD_EXIT] and TIMING_CFG_0[PRE_PD_EXIT].	12.5.5/12-42
PRETOACT	Precharge to Activate Timing	Should be set according to the specifications for the memory used (t_{RP})	12.5.6/12-45
ACTTOPRE	Activate to Precharge Timing	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used (t_{RAS})	12.5.6/12-45
ACTTORW	Activate to Read/Write Timing	Should be set according to the specifications for the memory used (t_{RCD})	12.5.6/12-45
CASLAT	CAS Latency	Should be set, along with the Extended CAS Latency, to the desired CAS latency	12.5.6/12-45
REFREC	Refresh Recovery	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used (T_{RFC})	12.5.6/12-45
WRREC	Write Recovery	Should be set according to the specifications for the memory used (t_{WR}).	12.5.6/12-45
ACTTOACT	Activate *A* to Activate *B*	Should be set according to the specifications for the memory used (t_{RRD})	12.5.6/12-45

Table 12-15. DDR3 Programming Summary (Continued)

Parameter	Description	Summary	Section/page
WRTORD	Write to Read Timing	Should be set according to the specifications for the memory used (t_{WTR})	12.5.6/12-45
ADD_LAT	Additive Latency	Should be set to the desired additive latency. This must be set to a value less than TIMING_CFG_1[ACTTORW]	12.5.7/12-48
WR_LAT	Write Latency	Should be set to the desired write latency. Note that DDR3 SDRAMs do not necessarily require the write latency to equal the CAS latency minus 1 cycle. The minimum WR_LAT that can be used in 1T timing mode is 5 cycles if DDR_RATE=0	12.5.7/12-48
RD_TO_PRE	Read to Precharge Timing	Should be set according to the specifications for the memory used (t_{RTP}). Time between read and precharge for non-zero value of additive latency (AL) is a minimum of AL + t_{RTP} cycles.	12.5.7/12-48
CKE_PLS	Minimum CKE Pulse Width	Should be set according to the specifications for the memory used (t_{CKE})	12.5.7/12-48
FOUR_ACT	Four Activate Window	Should be set according to the specifications for the memory used (t_{FAW}).	12.5.7/12-48
RD_EN	Registered DIMM Enable	If registered DIMMs are used, then this field should be set to 1	12.5.8/12-51
8_BE	8-beat burst enable	If a 64-bit bus is used, this should be set to 0. Otherwise, this should be set to 1. If this is set to 0, then other requirements in TIMING_CFG_4 will be needed to ensure t_{CCD} is met.	12.5.8/12-51
2T_EN	2T Timing Enable	In heavily loaded systems, this can be set to 1 to gain extra timing margin on the interface at the cost of address/command bandwidth.	12.5.8/12-51
DLL_RST_DIS	DLL Reset Disable	Should be set to 1	12.5.9/12-54
DQS_CFG	DQS Configuration	Should be set to 01	12.5.9/12-54
ODT_CFG	ODT Configuration	Can be set for termination at the IOs according to system topology. Typically, if ODT is enabled, then the internal IOs should be set up for termination only during reads to DRAM.	12.5.9/12-54
RWT	Read-to-write turnaround for same chip select (in TIMING_CFG_4)	This can be used to force a longer read-to-write turnaround time when accessing the same chip select. This is useful for burst chop mode, as there are some timing requirements to the same chip select that still must be met.	12.5.18/12-66
WRT	Write-to-read turnaround for same chip select (in TIMING_CFG_4)	This could be used to force a certain turnaround time between a write and read to the same chip select. This is useful for burst chop mode. However, it is expected that TIMING_CFG_1[WRTORD] will be programmed appropriately such that TIMING_CFG_4[WRT] can be set to 0000.	12.5.18/12-66
RRT	Read-to-read turnaround for same chip select (in TIMING_CFG_4)	Should typically be set to 0100 in burst chop mode (on-the-fly or fixed).	12.5.18/12-66
WWT	Write-to-write turnaround for same chip select (in TIMING_CFG_4)	Should typically be set to 0100 in burst chop mode (on-the-fly or fixed).	12.5.18/12-66

Table 12-15. DDR3 Programming Summary (Continued)

Parameter	Description	Summary	Section/page
ZQ_EN	ZQ Calibration Enable	Should be set to 1. The other fields in DDR_ZQ_CNTL should also be programmed appropriately based on the DRAM specifications.	12.5.20/12-70
WRLVL_EN	Write Leveling Enable	Can be set to 1 if write leveling is desired. Otherwise the value used in TIMING_CFG_2[WR_DATA_DELAY] will be used to shift all bytes during writes to DRAM. If write leveling will be used, all other fields in DDR_WRLVL_CNTL should be programmed appropriately based on the DRAM specifications.	12.5.21/12-72
BSTOPRE	Burst To Precharge Interval	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	12.5.13/12-61

12.4.2 DDR SDRAM Initialization Sequence

After configuration of all parameters is complete, system software must set DDR_SDRAM_CFG[MEM_EN] to enable the memory interface. Note that 200 µs (500 µs for DDR3) must elapse after DRAM clocks are stable (DDR_SDRAM_CLK_CNTL[CLK_ADJUST] is set and any chip select is enabled) before MEM_EN can be set, so a delay loop in the initialization code may be necessary if software is enabling the memory controller. If DDR_SDRAM_CFG[BI] is not set, the DDR memory controller will conduct an automatic initialization sequence to the memory, which will follow the memory specifications. If the bypass initialization mode is used, then software can initialize the memory through the DDR_SDRAM_MD_CNTL register.

Note: The DDR controller has an internal adjustment circuit use for automatic calibration. To prevent degradation of DDR transfer performance, disable this circuit before enabling the DDR memory using the following steps:

1. Write a value of 0x00000015 to the register at address 0xFFF20F30.
2. Write a value of 0x24000000 to the register at address 0xFFF20F54.

12.4.3 Self-Refresh Mode Usage

This section describes the options offered by this device to support battery-backed main memory.

12.4.3.1 Software Based Self-Refresh Scheme

The DDR controller also has a software-programmable bit, DDR_SDRAM_CFG_2[FRC_SR], that immediately puts main memory into self-refresh mode. See **Section 12.5.11, *DDR SDRAM Mode Configuration 2 Register (DDR_SDRAM_MODE_2)*** for a description of this register.

It is expected that a critical interrupt routine triggered by an external voltage sensing device will have time to set this bit.

12.4.3.2 Bypassing Re-initialization During Battery-Backed Operation

The DDR controller offers an initialization bypass feature (DDR_SDRAM_CFG[BI]), which system designers may use to prevent re-initialization of main memory during system power-on following an abnormal shutdown. See **Section 12.5.8**, *DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG),*" for information on this bit and **Section 12.5.16**, *DDR SDRAM Initialization Address Register (DDR_INIT_ADDR)* for a discussion about avoiding possible ECC errors in this mode.

Note that the DDR controller will automatically wait 200 DRAM cycles before issuing any command after the assertion of MCKE[0:1] when this mode is used.

12.5 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 (CSx_BNDS), **page 12-35**.
- Chip-Select Configuration Register (CSx_CONFIG), **page 12-36**.
- Chip-Select Configuration Register 2 (CSx_CONFIG_2), **page 12-38**
- DDR SDRAM Timing Configuration 3 Register (TIMING_CFG_3), **page 12-39**.
- DDR SDRAM Timing Configuration 0 Register (TIMING_CFG_0), **page 12-42**.
- DDR SDRAM Timing Configuration 1 Register (TIMING_CFG_1), **page 12-45**.
- DDR SDRAM Timing Configuration 2 Register (TIMING_CFG_2), **page 12-48**.
- DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG), **page 12-51**.
- DDR SDRAM Control Configuration 2 Register (DDR_SDRAM_CFG_2), **page 12-54**.
- DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE), **page 12-57**.
- DDR SDRAM Mode Configuration 2 Register (DDR_SDRAM_MODE_2), **page 12-58**.
- DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL), **page 12-58**.
- DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL), **page 12-61**.

- DDR SDRAM Data Initialization Register (DDR_DATA_INIT), **page 12-62**.
- DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL), **page 12-63**.
- DDR Initialization Address Register (DDR_INIT_ADDRESS), **page 12-64**.
- DDR Initialization Enable Register (DDR_INIT_EN), **page 12-65**
- DDR SDRAM Timing Configuration 4 Register (TIMING_CFG_4), **page 12-66**
- DDR SDRAM Timing Configuration 5 Register (TIMING_CFG_5), **page 12-68**
- DDR ZQ Calibration Control Register (DDR_ZQ_CNTL), **page 12-70**
- DDR Write Levelling Control Register (DDR_WRLVL_CNTL), **page 12-72**
- DDR Write Levelling Control 2 Register (DDR_WRLVL_CNTL_2), **page 12-75**
- DDR Write Levelling Control 3 Register (DDR_WRLVL_CNTL_3), **page 12-78**
- DDR SDRAM Self Refresh Counter Register (DDR_SR_CNTR), **page 12-81**
- DDR SDRAM Register Control Words 1 Register (DDR_SDRAM_RCW_1), **page 12-82**
- DDR SDRAM Register Control Words 2 Register (DDR_SDRAM_RCW_2), **page 12-83**
- DDR Debug Status Register 1 (DDRDSR_1), **page 12-86**
- DDR Debug Status Register 2 (DDRDSR_2), **page 12-87**
- DDR Control Driver Register 1 (DDRCDR_1), **page 12-87**
- DDR Control Driver Register 2 (DDRCDR_2), **page 12-91**
- DDR IP Block Revision 1 Register (DDR_IP_REV1), **page 12-92**.
- DDR IP Block Revision 2 Register (DDR_IP_REV2), **page 12-92**.
- Memory Data Path Error Injection Mask High Register (DDR_ERR_INJECT_HI), **page 12-95**.
- Memory Data Path Error Injection Mask Low Register (DDR_ERR_INJECT_LO), **page 12-96**.
- Memory Data Path Error Injection Mask ECC Register (DDR_ERR_INJECT), **page 12-97**
- Memory Data Path Read Capture High Register (CAPTURE_DATA_HI), **page 12-98**.
- Memory Data Path Read Capture Low Register (CAPTURE_DATA_LO), **page 12-98**.
- Memory Data Path Read Capture ECC Register (CAPTURE_ECC), **page 12-99**.
- Memory Error Detect Register (ERR_DETECT), **page 12-99**.
- Memory Error Disable Register (ERR_DISABLE), **page 12-101**.
- Memory Error Interrupt Enable Register (ERR_INT_EN), **page 12-102**.
- Memory Error Attributes Capture Register (CAPTURE_ATTRIBUTES), **page 12-103**.
- Memory Error Address Capture Register (CAPTURE_ADDRESS), **page 12-104**.
- Single-Bit ECC Memory Error Management Register (ERR_SBE), **page 12-105**.

Note: DDR controller 1 (M1 registers) uses base address: 0xFFF20000

12.5.1 Chip-Select x Bounds Register (CSx_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/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/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSx_BNDS define 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*.

If chip select interleaving is enabled, all fields in the lower interleaved chip select are used, and the other chip select 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 12-16. 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.
— 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.

12.5.2 Chip-Select x Configuration Register (CSx_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 registers enable the DDR chip selects x and set the number of row and column bits used for each chip select. Load these registers 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 is used and the other register fields are not used, 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 12-17. 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 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 12-17. 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 Configuration** 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 for Chip Select x** Specifies the number of logical bank bits MBA[2–0] for SDRAM on chip select x. See **Table 12-7** and **Table 12-8** 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 for Chip Select x** Specifies the number of row bits for SDRAM on chip select x. See **Table 12-7** and **Table 12-8** 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 for SDRAM on Chip Select x** Specifies the number of column bits for SDRAM on chip select x. See **Table 12-7** and **Table 12-8** for details.	000 8 column bits. 001 9 column bits. 010 10 column bits. 011 11 column bits. 100–111 Reserved.

12.5.3 Chip-Select x Configuration Register 2 (CSx_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/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

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 12-18** describes the CSx_CONFIG_2 fields.

Table 12-18. CSx_CONFIG_2 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 are placed on MA[2:0] during the write to the EMRS(2) register when the automatic hardware DRAM initialization is used (DDR_SDRAM_CFG[BI] is cleared when DDR_SDRAM_CFG[MEM_EN] is set). If this field is a non-zero value, then it overrides the least significant 3 bits in DDR_SDRAM_MODE_2[ESDMODE2] 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 12.5.11**),	000 001–111	Partial array self refresh is disabled Partial array self refresh is enabled per JEDEC specifications. Overriding the least significant 3 bits of EMRS or EMRS2 is only supported for DDR3 memory types.
— 23–0	0	Reserved. Write to zero for future compatibility.		

12.5.4 DDR SDRAM Timing Configuration 3 Register (TIMING_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/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/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TIMING_CFG_3 register sets the extended values for the refresh recovery time, activate-to-precharge, CAS latency, and control adjust fields, which are combined with the corresponding TIMING_CFG_1 fields to determine the total required time. **Table 12-19** describes the TIMING_CF3_3 fields.

Table 12-19. 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
— 23–21	0	Reserved. Write to zero for future compatibility.	

Table 12-19. TIMING_CFG_3 Bit Descriptions (Continued)

Bits	Reset	Description	Setting
EXT_ REFREC 20–16	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. t_{RFC} = {REFREC | + | EXT_REFREC}	00000 0 clock cycles. 00001 16 clock cycles. 00010 32 clock cycles. 00011 48 clock cycles. 00100 64 clock cycles. 00101 80 clock cycles. 00110 96 clock cycles. 00111 112 clock cycles 01000 128 clock cycles 01001 144 clock cycles 01010 160 clock cycles 01011 176 clock cycles 01100 192 clock cycles 01101 208 clock cycles 01110 224 clock cycles 01111 240 clock cycles 10000 256 clock cycles. 10001 272 clock cycles. 10010 288 clock cycles. 10011 304 clock cycles. 10100 320 clock cycles. 10101 336 clock cycles. 10110 352 clock cycles. 10111 368 clock cycles 11000 384 clock cycles 11001 400 clock cycles 11010 416 clock cycles 11011 432 clock cycles 11100 448 clock cycles 11101 464 clock cycles 11110 480 clock cycles 11111 496 clock cycles
— 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.	

Table 12-19. TIMING_CFG_3 Bit Descriptions (Continued)

Bits	Reset	Description	Setting
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] will be launched aligned with the other DRAM address and control signals. 001 MODT[0:1], $\overline{\text{MCS}}$[0:1], and MCKE[0:1] will be launched 1/2 platform cycle later than the other DRAM address and control signals. 010 MODT[0:1], $\overline{\text{MCS}}$[0:1], and MCKE[0:1] will be launched 1 platform cycle later than the other DRAM address and control signals. 011 MODT[0:1], $\overline{\text{MCS}}$[0:1], and MCKE[0:1] will be launched 3/2 platform cycles later than the other DRAM address and control signals. 100 MODT[0:1], $\overline{\text{MCS}}$[0:1], and MCKE[0:1] will be launched 2 platform cycles later than the other DRAM address and control signals. 101 MODT[0:1], $\overline{\text{MCS}}$[0:1], and MCKE[0:1] will be launched 5/2 platform cycles later than the other DRAM address and control signals. 110– 111 Reserved

12.5.5 DDR SDRAM Timing Configuration Register 0 (TIMING_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															
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**
	PRE_PD_EXIT		—		ODT_PD_EXIT				—				MRS_CYC			
Type	R/W															
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1

TIMING_CFG_0 register sets the number of clock cycles between various SDRAM control commands.

Table 12-20. 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 turn-around. For 0 clock cycles, 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. For 0 clock cycles, 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 will use a predefined value of 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.	00 0 clock cycles. 01 1 clock cycle. 10 2 clock cycles. 11 3 clock cycles.

Table 12-20. TIMING_CFG_0 Field Descriptions (Continued)

Bit	Reset	Description	Settings
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 will use a predefined value of 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.	00 0 clock cycles. 01 1 clock cycle. 10 2 clock cycles. 11 3 clock cycles.
ACT_PD_EXIT 23–20	0b0001	**Active Power-Down Exit Timing (t_{XP})** Specifies how many clock cycles to wait between exit from active power-down and issuing a command. The default is one clock cycle.	0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1100 8 clocks 1101 9 clocks 1110 10 clocks 1111 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
PRE_PD_EXIT 19–15	0b00010	**Precharge Power-Down Exit Timing (t_{xp})** Specifies how many clock cycles to wait after exiting precharge power down before issuing any command. **Note:** The decoding for this field requires the least significant bit to be set to add 16 clocks.	00000 Reserved 00001 16 clocks 00010 1 clock 00011 17 clocks 00100 2 clocks 00101 18 clocks 00110 3 clocks 00111 19 clocks 01000 4 clocks 01001 20 clocks 01010 5 clocks 01011 21 clocks 01100 6 clocks 01101 22 clocks 01110 7 clocks 01111 23 clocks 11000 8 clocks 10001 24 clocks 11010 9 clocks 10011 25 clocks 11100 10 clocks 10101 26 clocks 11110 11 clocks 10111 27 clocks 11000 12 clocks 11001 28 clocks 11010 13 clocks 11011 29 clocks 11100 14 clocks 11101 30 clocks 11110 15 clocks 11111 31 clocks

Table 12-20. TIMING_CFG_0 Field Descriptions (Continued)

Bit	Reset	Description	Settings
— 14–12	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:** 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.	
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.

12.5.6 DDR SDRAM Timing Configuration Register 1 (TIMING_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															
Reset	0	0	0	0	0	0	0	0	0	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															
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 12-21. 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:** 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.	*If TIMING_CFG_3 [EXT_ACTTOPRE] = 1:* 0000 0 clock cycles. 0001 1 clock cycles. 0010 2 clock cycles. 0011 3 clock cycles. *If TIMING_CFG_3 [EXT_ACTTOPRE] = 0:* 0000 16 clock cycles. 0001 17 clock cycles. 0010 18 clock cycles. 0011 19 clock cycles. *For any value of TIMING_CFG_3 [EXT_ACTTOPRE]:* 0100 4 clock cycles. 0101 5 clock cycles. 0110 6 clock cycles. 0111 7 clock cycles. ... 1111 15 clock cycles.

Table 12-21. 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 12.5.10**, *DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE)* on **page 12-57**. This field must be programmed at initialization for proper operation of the DDR Controller, as indicated in **Section 12.5.9**, *DDR SDRAM Control Configuration Register 2 (DDR_SDRAM_CFG_2)*.	0000 Reserved. 0001 1 clock cycle. 0010 1.5 clock cycles. 0011 2 clock cycles. 0100 2.5 clock cycles 0101 3 clock cycles. 0110 3.5 clock cycles 0111 4 clock cycles. 1000 4.5 clock cycles 1001 5 clock cycles. 1010 5.5 clock cycles 1011 6 clock cycles. 1100 6.5 clock cycles 1101 7 clock cycles. 1110 7.5 clock cycles 1111 8 clock cycles.
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. t_{RFC} = {REFREC | + | EXT_REFREC} **Note:** The minimum value for REFREC is 8 clock cycles. This required value can be calculated by referring to the AC specification of the SDRAM device. The AC specification indicates a maximum refresh to activate interval in ns.	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. 1000 16 clock cycles. 1001 17 clock cycles. 1010 18 clock cycles. 1011 19 clock cycles. 1100 20 clock cycles. 1101 21 clock cycles. 1110 22 clock cycles. 1111 23 clock cycles.

Table 12-21. TIMING_CFG_1 Field Descriptions (Continued)

Bits	Reset	Description	Settings
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.	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.
ACTTOACT 7–4	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). The default is one clock cycle. This interval is calculated from the AC specifications of the SDRAM.	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 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
WRTORD 3–0	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.	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 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

12.5.7 DDR SDRAM Timing Configuration Register 2 (TIMING_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				—		RD_TO_PRE
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**
	RD_TO_PRE			WR_DATA_ DELAY				CKE_PLS			FOUR_ACT					
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TIMING_CFG_2 register sets the clock delay to data for writes and should be defined according to the system timing. **Table 12-23** describes the TIMING_CFG_2 fields.

Table 12-22. 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].	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 Reserved.

Table 12-22. 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. **Note:** 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)
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. **Note:** Total write latency for DDR3 is equal to WR_LAT + ADD_LAT. If a write latency of 1 is required, then the additive latency must also be configured to at least 1 cycle.	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.
— 18–17	0	Reserved. Write to zero for future compatibility.	

Table 12-22. TIMING_CFG_2 Field Descriptions (Continued)

Bit	Reset	Description	Settings
RD_TO_PRE 16–13	0	**Read to Precharge (t_{RTP})** Specifies the number of clock cycles between the read command and the precharge command. .	0000 Reserved. 0001 1 clock cycle. 0010 2 clock cycles. 0011 3 clock cycles. 0100 4 clock cycles. 0101 5 clock cycle. 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 cycle. 1110 14 clock cycles. 1111 15 clock cycles.
WR_ DATA_ DELAY 12–9	0	**Write Command to Write Data Strobe Delay** Controls the amount of delay applied to the data and data strobes for writes. See **Section 12.3.9**, *DDR SDRAM Write Timing Adjustments* for details. 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 requirements, do no use these values.	0000 0 clock delay 0001 2 clock delay 0010 1/4 clock delay 0011 9/4 clock delay 0100 1/2 clock delay 0101 5/2 clock delay 0110 3/4 clock delay 0111 Reserved 1000 1 clock delay 1001 Reserved 1010 5/4 clock delay 1011 Reserved 1100 3/2 clock delay 1101 Reserved. 1110 7/4 clock delay 1111 Reserved
CKE_PLS 8–6	0	**Minimum CKE Pulse Width (t_{CKE})** Specifies the minimum clock cycles t_{CKE}.	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.
FOUR_ACT 5–0	0	**Window for Four Activates (t_{FAW}).** Specifies the t_{FAW} timing. **Note:** This is applied to DDR3 with eight logical banks only.	000000 Reserved. 000001 1 clock cycle. 000010 2 clock cycles. 000011 3 clock cycles. 000100 4 clock cycles. ... 011110 30 clock cycles. 011111 31 clock cycles. 100000 32 clock cycles. 100001– 111111 Reserved.

12.5.8 DDR SDRAM Control Configuration Register (DDR_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	DBW		8_BE	NCAP	3T_EN
Type	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															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_SDRAM_CFG register enables the interface logic and specifies certain operating features such as self refreshing, error checking and correcting, registered DIMMs, and dynamic power management.

Table 12-23. 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** 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. 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.	011 Reserved 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 12-23. DDR_SDRAM_CFG Field Descriptions (Continued)

Bits	Reset	Description	Settings
DBW 20–19	0	**DRAM Data Bus Width** Selects bus size.	00 64-bit bus is used. 01 32-bit bus is used. 10 Reserved. 11 Reserved.
8_BE 18	0	**8-Beat Burst Enable** **Note:** DDR3 (SDRAM_TYPE = 111) must use 8-beat bursts when using 32-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. **Note:** This field cannot be set if DDR_SDRAM_CFG[2T_EN] is also set. **Note:** This field may not be used with 4-beat bursts.	0 1T timing is enabled if 2T_EN is cleared. The DRAM command/address are held for only 1 cycle on the DRAM bus. 1 3T timing is enabled. The DRAM command/address are held for 3 full cycles on the DRAM bus for every DRAM transaction. However, the chip select is only held for the third cycle
2T_EN 15	0	**2T Timing Enable** Enables/disabled 2T timing. **Note:** This field cannot be set if DDR_SDRAM_CFG[3T_EN] is also set. **Note:** RD_EN and 2T_EN cannot be set simultaneously	0 1T timing is used if 3T_EN is cleared (0). The DRAM command/address are held for only 1 cycle on the DRAM bus. 1 2T timing is enabled. The DRAM command/address are held for 2 full cycles on the DRAM bus for every DRAM transaction. The chip select is only held for the second cycle.
BA_INTLV_CTL 14	0	**Bank (CS) Interleaving Control** Specifies whether to use bank interleaving.	0 No external banks are interleaved. 1 External memory banks 0 and 1 are interleaved.
— 13–4	0	Reserved. Write to zero for future compatibility.	
HSE 3	0	**Global Half-Strength Override** Specifies whether the I/O drivers are configured to full strength or half strength. The half strength impedance is used by the MDIC, address/command, data, and clock, but only if automatic hardware calibration is disabled and the corresponding group software override is disabled in the DDR control driver registers **Section 12.5.31**, *DDR Control Driver Register 1 (DDRCDR_1)* and **Section 12.5.32**, *DDR Control Driver Register 2 (DDRCDR_2)*. Clear this bit if using automatic hardware calibration.	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 12-23. 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 bypassing initialization and forcing the MODE REGISTER SET commands through software. This bit should be used carefully. Using this MEM_HALT 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 must initialize memory through the DDR_SDRAM_MD_CTR 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:** The DDR controller does not issue a DDL 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 you must force the controller 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 12-64**.	0 DDR controller cycles through the initialization routine based on the value of SDRAM_TYPE. 1 Initialization routine is bypassed.

12.5.9 DDR SDRAM Control Configuration Register 2 (DDR_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															
Reset	0	0	0	0	0	0	0	0	0	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				—			UNQ_MRS_EN	—		AP_EN	D_INIT	—	RCW_EN	CD_DIS	MD_EB
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_SDRAM_CFG_2 register provides control configuration for the DDR controller in addition to that provided by DDR_SDRAM_CFG.

Table 12-24. 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**	00 Reserved. 01 Differential DQS signals used. 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 12-47** and **Table 12-48** for the definition of the impedance value that will be 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.	

Table 12-24. DDR_SDRAM_CFG_2 Field Descriptions (Continued)

Bit	Reset	Description	
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.	0000 Reserved. 0001 1 refresh at a time. 0010 2 refreshes at a time. 0011 3 refreshes at a time. ... 1000 8 refreshes at a time. 1001 - 1111 Reserved.
— 11–9	0	Reserved. Write to zero for future compatibility.	
UNQ_MRS_EN 8	0	**Unique MRS Enable.** Determines if the DDR_SDRAM_MODE_{3,4} registers will be used when initializing the memories for chip selects 1. These can be used to provide unique values to the Mode Registers of the DRAM to allow different termination values for each rank.	0 Unique MRS disabled DDR_SDRAM_MODE and DDR_SDRAM_MODE_2 are used for all chip selects 1 Unique MRS Enabled DDR_SDRAM_MODE, DDR_SDRAM_MODE_2 are used for chip select 0. FFR_SDRAM_MODE_3 and DDR_SDRAM_MODE_4 are used for chip select 1.
— 7–6	0	Reserved. Write to zero for future compatibility.	
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[0–3], MODT[0–3], or $\overline{\text{MCS}}$[0–3] 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 will automatically initialize DRAM after it is enabled. This bit will be automatically cleared by hardware once the initialization is completed. This data initialization bit should only be set when the controller is idle. The value in DDR_DATA_INIT register will be used to initialize memory	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.	

Table 12-24. DDR_SDRAM_CFG_2 Field Descriptions (Continued)

Bit	Reset	Description	
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.
CD_DIS 1	0	**Corrupted Data Disable** If this bit is set, then the corrupted data feature is disabled. When the corrupted data feature is enabled, the DDR controller inverts the generated ECC code for any beat of data that is known to have corrupted data. When a read to the corrupted data is generated later, the ERR_DETECT[CDE] error bit is set and generates an error interrupt if error reporting is enabled.	0 Corrupted data is enabled. 1 Corrupted data is disabled.
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 (MBA0 versus MBA1; MA3 versus MA4; MA5 versus MA6; MA7 versus MA8).	0 Mirrored DIMMs are not used. 1 Mirrored DIMMs are used.

12.5.10 DDR SDRAM Mode Configuration Register (DDR_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 register sets the values loaded into the DDR mode registers.

Table 12-25. DDR_SDRAM_MODE Bit Descriptions

Bit	Refresh	Description
ESDMODE 31–16 MR1	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. The write leveling enable bit should be cleared by software in this field
SDMODE 15–0 MR0	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.

12.5.11 DDR SDRAM Mode Configuration 2 Register (DDR_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 register sets the values loaded into the DDR extended mode 2 and 3 registers.

Table 12-26. DDR_SDRAM_MODE_2 Bit Descriptions

Bit	Reset	Description
ESDMODE2 31–16 MR2	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 MR3	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.

12.5.12 DDR SDRAM Mode Control Register (DDR_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		WRCW	—		
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**
	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 register allows software to initiate the following tasks:

- Issue a MODE REGISTER SET command to a particular chip select
- Issue an immediate REFRESH to a particular chip select
- Issue an immediate PRECHARGE or PRECHARGE ALL command to a particular chip select
- Force the CKE signals to a specific value

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 guaranteed and can cause malfunction of the DDR SDRAM. MD_EN, SET_REF, and SET_PRE are mutually exclusive; only one of these fields should be set at a time.

Table 12-27. DDR_SDRAM_MD_CNTL Bit Descriptions

Bit	Reset	Description	Settings
MD_EN 31	0	**Mode Enable** 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. **Note:** MD_SEL and CS_SEL_MD_EN are mutually exclusive and cannot be set at the same time.	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.
CS_SEL 30–28	0	**Select for Chip Select** Specifies the chip select to drive active due to any command forced by software in DDR_SDRAM_MD_CNTL.	000 Chip select 0 is active. 001 Chip select 1 is active. 010 Reserved. 011 Reserved. 100 Chip select 0 and chip select 1 are active. 101 Reserved. 110 Reserved. 111 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 to the memory bank address pins (MBA*n*) of the DDR controller.	000 MR Mode register 001 EMR Extended Mode Register 010 EMR2 Extended Mode Register 2 011 EMR3 Extended Mode Register 3

Table 12-27. DDR_SDRAM_MD_CNTL Bit Descriptions (Continued)

Bit	Reset	Description	Settings
SET_REF 23	0	**Set Refresh** Forces a refresh to the chip select specified by DDR_SDRAM_MD_CNTL. Software sets this bit and hardware clears it when the command is issued. **Note:** MD_EN, 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.
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. Software sets this bit, and hardware clears it when the command is issued. **Note:** MD_EN, 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 these 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 will be used, since write leveling is run automatically before DDR_SDRAM_MD_CNTL can be used to force RCW writes.	0 Indicates that a register control word write 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 bit 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 12-28 shows how DDR_SDRAM_MD_CNTL fields should be set for each of the tasks described in **Table 12-27**.

Table 12-28. 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 12-27.**	—	Selects logical bank	—
MD_VALUE	Value written to mode register	—	Only MD_VALUE10 is significant. See **Table 12-27.**	—
CKE_CNTL	0	0	0	See **Table 12-27.**

12.5.13 DDR SDRAM Interval Configuration Register (DDR_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 register 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 open after it is accessed.

Table 12-29. DDR_SDRAM_INTERVAL Bit Descriptions

Bit	Reset	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 12-29. DDR_SDRAM_INTERVAL Bit Descriptions (Continued)

Bit	Reset	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. 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.

12.5.14 DDR SDRAM Data Initialization Register (DDR_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 register provides the value to initialize memory if DDR_SDRAM_CFG_2[D_INIT] is set.

Table 12-30. 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.

12.5.15 DDR SDRAM Clock Control Configuration Register (DDR_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/W															
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/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_SDRAM_CLK_CNTL register provides a 1/8 cycle clock adjustment.

Table 12-31. DDR_SDRAM_CLK_CNTL Bit Descriptions

Bit	Reset	Description	Settings
— 31–27	0	Reserved. Write to zero for future compatibility.	
CLK_ADJUST 26–23	0100	**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.	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. 1001–1111 Reserved.
— 22–0	0	Reserved. Write to zero for future compatibility.	

12.5.16 DDR SDRAM Initialization Address Register (DDR_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 register 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 $\overline{\text{HRESET}}$ is asserted after the DRAM enters Self-Refresh mode, memory is not initialized. Therefore, this address should be written using an 8- or 32-bit transaction to avoid possible ECC errors when this address is accessed later.

Table 12-32. DDR_INIT_ADDR Bit Descriptions

Bit	Reset	Description	Settings
INIT_ADDR 31–0	0	**Initialization Address** Provides the address used for the data strobe to data skew adjustment and automatic CAS to preamble calibration after setting DDR_SDRAM_CFG[MEM_EN]. This address is written during the initialization sequence.	

12.5.17 DDR Initialization Enable Register (DDR_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															
Reset	0	0	0	0	0	0	0	0	0	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

The DDR SDRAM register initialization enable register selects the initialization address to use for the DDR_INIT_ADDR for the data-strobe-to-data-skew adjustment and automatic $\overline{\text{CAS}}$-to-preamble calibration after power-on reset. **Table 12-40** describes the DDR_INIT_EN fields.

Table 12-33. DDR_INIT_EN Field Descriptions

Bit	Reset	Description	Settings
UIA 31	0	**Use Initialization Address** Indicates the source of the initialization address.	0 Use the default address for training sequence as calculated by the controller. This will be 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.	

12.5.18 DDR SDRAM Timing Configuration 4 Register (TIMING_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 12-34** describes the TIMING_CFG_4 fields.

Table 12-34. 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 will be added between a read to write turnaround for transactions to the same chip select. If a value of 0000 is chosen, then the DDR controller will use the value used for transactions to different chip selects, as defined in TIMING_CFG_0[RWT]. This field can be used to improve performance when operating in burst-chop mode by forcing transactions to the same chip select to use extra cycles, while transaction to different chip selects can utilize the tri-state time on the DRAM interface. Regardless of the value that is set in this field, the value defined by TIMING_CFG_0[RWT] will also be met before issuing a write command.	0000 Default 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 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 will be added between a write to read turnaround for transactions to the same chip select. If a value of 0000 is chosen, then the DDR controller will use the value used for transactions to different chip selects, as defined in TIMING_CFG_0[WRT]. This field can be used to improve performance when operating in burst-chop mode by forcing transactions to the same chip select to use extra cycles, while transaction to different chip selects can utilize the tri-state time on the DRAM interface. Regardless of the value that is set in this field, the value defined by TIMING_CFG_0[WRT] will also be met before issuing a read command.	0000 Default 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks

Table 12-34. 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 will be added between reads to the same chip select. If a value of 0000 is chosen, then 2 cycles will be required between read commands to the same chip select if 4-beat bursts are used (4 cycles will be required if 8-beat bursts are used). Note that DDR3 does not support 4-beat bursts. However, this field may be used to add extra cycles when burst-chop mode is used, and the DDR controller must wait 4 cycles for read-to-read transactions to the same chip select.	0000 BL/2 clocks 0001 BL/2 + 1 clock 0010 BL/2 + 2 clocks 0011 BL/2 + 3 clocks 0100 BL/2 + 4 clocks 0101 BL/2 + 5 clocks 0110 BL/2 + 6 clocks 0111 BL/2 + 7 clocks 1000 BL/2 + 8 clocks 1001 BL/2 + 9 clocks 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 will be added between writes to the same chip select. If a value of 0000 is chosen, then 2 cycles will be required between write commands to the same chip select if 4-beat bursts are used (4 cycles will be required if 8-beat bursts are used). Note that DDR3 does not support 4-beat bursts. However, this field may be used to add extra cycles when burst-chop mode is used, and the DDR controller must wait 4 cycles for write-to-write transactions to the same chip select.	0000 BL/2 clocks 0001 BL/2 + 1 clock 0010 BL/2 + 2 clocks 0011 BL/2 + 3 clocks 0100 BL/2 + 4 clocks 0101 BL/2 + 5 clocks 0110 BL/2 + 6 clocks 0111 BL/2 + 7 clocks 1000 BL/2 + 8 clocks 1001 BL/2 + 9 clocks 1010 BL/2 + 10 clocks 1011 BL/2 + 11 clocks 1100 BL/2 + 12 clocks 1101 BL/2 + 13 clocks 1110 BL/2 + 14 clocks 1111 BL/2 + 15 clocks
— 25–2	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 will take for the DRAMs DLL to lock at power-on reset and after exiting self refresh. The controller will wait the specified number of cycles before issuing any commands after exiting POR or self refresh.	00 200 clocks 01 512 clocks 10 Reserved 11 Reserved

12.5.19 DDR SDRAM Timing Configuration 5 Register (TIMING_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 12-35** describes the TIMING_CFG_5 fields.

Table 12-35. 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 will pass from when a read command is placed on the DRAM bus until the assertion of the relevant ODT signal(s). The default case (00000) provides a decode of RL - 3 cycles to support legacy of past products. RL is the read latency, derived from CAS latency + additive latency. If 2T timing is used, an extra cycle will automatically be added to the value selected in this field. Recommended value should be defined by RL - WL (Read Latency - Write Latency)	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. 01111 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
— 23	0	Reserved. Write to zero for future compatibility.	

Table 12-35. TIMING_CFG_5 Field Descriptions (Continued)

Bits	Reset	Description	Settings
RODT_OFF 22–20	0	**Read to ODT Off** Specifies the number of cycles that the relevant ODT signal(s) will remain asserted for each read transaction. The default case (000) will leave the ODT signal(s) asserted for 3 DRAM cycles. Recommended value is 4 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 will pass from when a write command is placed on the DRAM bus until the assertion of the relevant ODT signal(s). The default case (00000) provides a decode of WL - 3 cycles to support legacy of past products. WL is the write latency, derived from Write Latency + Additive Latency. If 2T timing is used, an extra cycle will automatically be added to the value selected in this field. Recommended value is 0 cycles.	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. 01111 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.	
WODT_OFF 10–8	0	**Write to ODT Off** Specifies the number of cycles that the relevant ODT signal(s) will remain asserted for each write transaction. The default case (000) will leave the ODT signal(s) asserted for 3 DRAM cycles. Recommended value is 4 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.	

12.5.20 DDR ZQ Calibration Control Register (DDR_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 will be 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 12-36** describes the DDR_ZQ_CNTL fields.

Table 12-36. 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.	

Table 12-36. DDR_ZQ_CNTL Field Descriptions (Continued)

Bit	Reset	Description	Settings
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 will be calibrated separately, and this time must elapse after the ZQCS command is issued for each chip select before a separate command may be issued.	0000 1 clocks 0001 2 clocks 0010 4 clocks 0011 8 clocks 0100 16 clocks 0101 32 clocks 0110 64 clocks 0111 128 clocks 1000 256 clocks 1001 512 clocks All other values reserved.
— 7–0	0	Reserved. Write to zero for future compatibility.	

12.5.21 DDR Write Leveling Control Register (DDR_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 Levelling Control register provides controls for write levelling, as it is supported for DDR3 memory devices. **Table 12-37** describes the DDR_WRLVL_CNTL fields.

Table 12-37. DDR_WRLVL_CNTL Field Descriptions

Bits	Reset	Description	Settings
WRLVL_EN 31	0	**Write Leveling Enable** This bit determines if write leveling will be used. If this bit is set, then the DDR controller will perform write leveling immediately after initializing the DRAM. This bit should only be set if DDR3 memory is used (DDR_SDRAM_CFG[SDRAM_TYPE] = 3'b111).	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 Write Leveling Mode Is Programmed (t_{WL_MRD})** Determines how many cycles to wait after write leveling 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 Write Leveling Mode Is Programmed** Determines how many cycles to wait after write leveling mode is programmed until ODT is asserted. should be programmed to a value greater than t_{MOD} 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 12-37. DDR_WRLVL_CNTL Field Descriptions (Continued)

Bits	Reset	Description	Settings
WRLVL_ DQSEN 18–16	0	**DQS/DQS Delay after Write Leveling Mode Is Programmed (t_{WL_DQSEN})** Determines how many cycles to wait after write leveling 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 write leveling mode. This field should be programmed at least 6 cycles higher than t_{WLO} to allow enough time for propagation delay and sampling of the prime data bits. This field is only relevant when DDR_WRLVL_CNTL[WRLVL_EN] is set.	0000 32 clocks 0001 1 clocks 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 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. The recommended value is 64 clocks	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 12-37. 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

12.5.22 DDR Write Leveling Control 2 Register (DDR_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 Levelling Control 2 register provides controls for write levelling, as it is supported for DDR3 memory devices. This register specifically defines the starting points for the individual data strobes. **Table 12-38** describes the DDR_WRLVL_CNTL_2 fields.

Table 12-38. 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 12-38. 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 12-38. 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

12.5.23 DDR Write Leveling Control 3 Register (DDR_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 Levelling Control 3 register provides controls for write levelling, as it is supported for DDR3 memory devices. This register specifically defines the starting points for the individual data strobes. **Table 12-38** describes the DDR_WRLVL_CNTL_3 fields.

Table 12-39. 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 12-39. 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 12-39. 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

12.5.24 DDR Self Refresh Counter Register (DDR_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/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

DDR_SR_CNTR Register can be programmed to force the DDR controller to enter self refresh after a predefined period of idle time.

Table 12-40. 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 will enter self refresh. Anytime a transaction is issued to the DDR controller, it will reset its internal counter. When a new transaction is received by the DDR controller, it will exit self refresh and reset its internal counter. If this field is zero, then the described power savings feature will be disabled. In addition, if a non-zero value is programmed into this field, then the DDR controller will exit self refresh anytime a transaction is issued to the DDR controller, regardless of the reason self refresh was initially entered.	0000 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.		

12.5.25 DDR SDRAM Register Control Words 1 Register (DDR_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 register should be programmed with the intended values of the register control words if DDR_SDRAM_CFG2[RCW_EN] is set. Each 4-bit field represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during register control word writes.

Table 12-41. DDR_SDRAM_RCW_1 Bit Descriptions

Bit	Reset	Description
RCW0 31–28	0	**Register Control Word 0** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 0.
RCW1 27–24	0	**Register Control Word 1** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 1.
RCW2 23–20	0	**Register Control Word 2** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 2.
RCW3 19–16	0	**Register Control Word 3** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 3.
RCW4 15–12	0	**Register Control Word 4** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 4.
RCW5 11–8	0	**Register Control Word 5** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 5.
RCW6 7–4	0	**Register Control Word 6** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 6.
RCW7 3–0	0	**Register Control Word 7** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 7.

12.5.26 DDR SDRAM Register Control Words 2 Register (DDR_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 register should be programmed with the intended values of the register control words if DDR_SDRAM_CFG2[RCW_EN] is set. Each 4-bit field represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during register control word writes.

Table 12-42. DDR_SDRAM_RCW_1 Bit Descriptions

Bit	Reset	Description
RCW8 31–28	0	**Register Control Word 8** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 8.
RCW9 27–24	0	**Register Control Word 9** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 9.
RCW10 23–20	0	**Register Control Word 10** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 10.
RCW11 19–16	0	**Register Control Word 11** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 11.
RCW12 15–12	0	**Register Control Word 12** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 12.
RCW13 11–8	0	**Register Control Word 13** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 13.
RCW14 7–4	0	**Register Control Word 14** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 14.
RCW15 3–0	0	**Register Control Word 15** Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 15.

12.5.27 DDR SDRAM Mode 3 Configuration Register (DDR_SDRAM_MODE_3)

DDR_SDRAM_MODE_3DDR SDRAM Mode 3 Configuration Register Offset 0x0200

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_3 Register sets the values loaded into the DDR mode registers. This register is used specifically for chip select 1 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set.

Table 12-43. DDR_SDRAM_MODE 3 Bit Descriptions

Bit	Refresh	Description
ESDMODE 31–16 MR1	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_3 bit 16. The MSB of the SDRAM extended mode register value must be stored at DDR_SDRAM_MODE_3 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. The write leveling enable bit should be cleared by software in this field
SDMODE 15–0 MR0	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_3 bit 0. The MSB of the SDRAM mode register value must be stored at DDR_SDRAM_MODE_3 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.

12.5.28 DDR SDRAM Mode Configuration 4 Register (DDR_SDRAM_MODE_4)

DDR_SDRAM_MODE_4 DDR SDRAM Mode Configuration 4 Register — Offset 0x0204

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_SDDR_SDRAM_MODE_4 register sets the values loaded into the DDR extended mode 2 and 3 registers. This register is used specifically for chip select 1 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set.

Table 12-44. DDR_SDRAM_MODE_4 Bit Descriptions

Bit	Reset	Description
ESDMODE2 31–16 MR2	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_4 bit 16. The MSB of the SDRAM extended mode 2 register value must be stored at DDR_SDRAM_MODE_4 bit 31.
ESDMODE3 15–0 MR3	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_4 bit 0. The MSB of the SDRAM extended mode 3 register value must be stored at DDR_SDRAM_MODE_4 bit 15.

12.5.29 DDR Debug Status Register 1 (DDRDSR_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 register contains the DDR driver compensation input value and the current settings of the P and N FET impedance for MDICn, command/control, and data.

Table 12-45. 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. Write to zero for future compatibility.
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**

12.5.30 DDR Debug Status Register 2 (DDRDSR_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								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

DDRDSR_2 register contains the current settings of the P and N FET impedance for the DDR drivers for clocks.

Table 12-46. DDRDSR_2 Bit Descriptions

Bit	Reset	Description	Settings
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. Write to zero for future compatibility.	

12.5.31 DDR Control Driver Register 1 (DDRCDR_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 register sets the driver hardware compensation enable, the DDR MDIC driver P/N impedance, ODT termination value for I/Os, 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.

The fields in DDRCDR_1, except DDRCDR_1[ODT], are used to enable driver calibration with the MDIC pins. This can be used to calibrate the DDR drivers to 36 Ω; however, this should only

be used for full-strength driver applications. If half-strength drivers are used, disable this calibration. Hardware DDR driver calibration is enabled by setting DDRCDR_1[DHC_EN].

Note: All driver calibration (hardware and software) must be done before the DDR controller is enabled (that is, before setting DDR_SDRAM_CFG[MEM_EN]).

Table 12-47. 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 for P-Impedance Output Enable**	0 Output override disabled. 1 Output override enabled.
DSO_MDIC_NZ_OE 20	0	**Driver Software Override for N-Impedance Output Enable**	0 Output override disabled. 1 Output override 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 Ω 001 55 Ω 010 60 Ω 011 50 Ω 100 150 Ω 101 43 Ω 110 120 Ω 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**	

Software can be used to calibrate the drivers instead of the automatic hardware calibration. If software calibration is required, use the following steps:

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 DDR3 (1.5 V) are:

- 0000 lowest strength/highest impedance
- 0001
- 0011
- 0010
- 0110
- 0111 half strength
- 0101
- 0100
- 1100
- 1101
- 1111
- 1110
- 1010
- 1011
- 1001
- 1000 highest strength/lowest impedance

12.5.32 DDR Control Driver Register 2 (DDRCDR_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															
Reset	0	0	0	0	0	0	0	0	0	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/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRCDR_2 register sets the driver software override enable for clocks, and the DDR clocks driver P/N impedance.

Table 12-48. 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	**Driver Software Override Value for Clocks P-Impedance Override**	
DSO_CLKNZ 23–20	0	**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 Ω 001 55 Ω 010 60 Ω 011 50 Ω 100 150 Ω 101 43 Ω 110 120 Ω 111 Reserved

12.5.33 DDR SDRAM IP Block Revision 1 Register (DDR_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	1	0	0

DDR_IP_REV1 register provides read-only fields with the IP block ID, along with major and minor revision information.

Table 12-49. DDR_IP_REV1 Bit Descriptions

Bit	Reset	Description
IP_ID 31–16	0x0002	**IP Block ID** For the DDR controller, this value is 0x0002.
IP_MJ 15–8	0x04	**Major Revision** Current setting is 0x04
IP_MN 7–0	0x04	**Minor Revision** Current setting is 0x04

12.5.34 DDR SDRAM IP Block Revision 2 Register (DDR_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	x	x	x	x	x	x	x	x
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	x	x	x	x	x	x	x	x

DDR_IP_REV2 register provides read-only fields with the IP block integration and configuration options.

Table 12-50. DDR_IP_REV2 Bit Descriptions

Bit	Reset	Description
— 31–24	0	Reserved. Write to zero for future compatibility.
IP_INT 23–16	0	**IP Block Integration Options**

Table 12-50. DDR_IP_REV2 Bit Descriptions (Continued)

Bit	Reset	Description
— 15–8	0	Reserved. Write to zero for future compatibility.
IP_CFG 7–0	0	**IP Block Configuration Options**

12.5.35 DDR Memory Test Control Register (DDR_MTCR)

DDR_MTCR DDR Memory Test Control Register Offset 0x0D00

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MT_EN	—					MT_TYP		—				MT_TRNARND			
Type	R/W															
Reset	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—															MT_STAT
Type	R/W															
Reset	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x

DDR_MTCR provides enable and control for an automatic memory test.

Table 12-51. DDR_MTCR Bit Descriptions

Bit	Reset	Description	Settings
MT_EN 31	0	**Memory Test Enable** This bit can be set by software to enable memory testing. Based on the value of MT_TYP, the controller either performs only writes, only reads, or reads and writes. The controller issues transactions throughout memory, as defined by the CSx_CONFIG and CSx_BNDS registers. The memory controller checks the data during the test. In addition, if ECC is enabled, the software should read the ERR_DETECT register to ensure there are no ECC errors. If an ECC error is detected, the error capture registers hold the address, data, and attributes captured for the first failure. If no ECC error occurs and the test fails for a data miscompare, the registers hold the data for the first miscompare. Transactions with ECC errors take precedent over data miscompare errors. Hardware clears the MT_EN bit after the test completes. This bit can be set before or after the controller is enabled.	0 Memory test disabled. 1 Memory test enabled.
— 30–26	0	Reserved. Write to zero for future compatibility.	
MT_TYP 25–24	0	**Memory Test Type** This field determines the type of test: write only, read only, or read/write. Do not use the read-only test unless memory is already initialized.	00 Read/write 01 Write only. 10 Read only 11 Reserved.
— 23–20	0	Reserved. Write to zero for future compatibility.	

Table 12-51. DDR_MTCR Bit Descriptions (Continued)

Bit	Reset	Description	Settings
MT_TRNARND 19–16	0	**Memory Test Turnaround** This field determines how many writes to issue during the memory test before the reads to the same addresses are issued. This can be used both to allow longer streams of reads/writes and to test the write-to-read and read-to-write turnarounds in a stressed setting. The field is only used if MT_TYP = 00.	0000 Entire memory is written before read transactions are issued. 0001 Total read/write streams = 1 transaction each. 0010 Total read/write streams = 2 transactions each. 0011 Total read/write streams = 4 transactions each.
— 15–1	0	Reserved. Write to zero for future compatibility.	
MT_STAT 0	0	**Memory Test Status** After hardware clears MT_EN, this bit sets is a failure occurred. If the failure occurred during the memory test (data miscompare), this bit sets at the same time MT_EN is cleared. Software can clear this bit after it is set.	0 No failure detected. 1 Data miscompare detected.

12.5.36 DDR Data Memory Test Pattern x Register (DDR_MTPx)

DDR_MTP0 DDR Data Memory Test Pattern 0 Register Offset 0x0D20
DDR_MTP1 DDR Data Memory Test Pattern 1 Register Offset 0x0D24
DDR_MTP2 DDR Data Memory Test Pattern 2 Register Offset 0x0D28
DDR_MTP3 DDR Data Memory Test Pattern 3 Register Offset 0x0D2C
DDR_MTP4 DDR Data Memory Test Pattern 4 Register Offset 0x0D30
DDR_MTP5 DDR Data Memory Test Pattern 5 Register Offset 0x0D34
DDR_MTP6 DDR Data Memory Test Pattern 6 Register Offset 0x0D38
DDR_MTP7 DDR Data Memory Test Pattern 7 Register Offset 0x0D3C
DDR_MTP8 DDR Data Memory Test Pattern 8 Register Offset 0x0D40
DDR_MTP9 DDR Data Memory Test Pattern 9 Register Offset 0x0D44

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

The DDR memory test pattern registers provide the data patterns to be written during the specified set of 32-bits of 40 each 40-byte memory test pattern (numbered 0 through 9). These values are used when DDR_MTCR[MT_EN] is set to enable the memory read/write tests.

Table 12-52. DDR_MTPx Bit Descriptions

Bit	Reset	Description
DDR_PATT 31–0	0	Memory Test Pattern This 32-bit test pattern is used during the memory test when enabled. The memory test reads/writes a programmable 40-byte pattern, using the ten DDR_MTPx registers.

12.5.37 DDR SDRAM Memory Data Path Error Injection Mask High Register (DATA_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 12-53. 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 high 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.

12.5.38 DDR SDRAM Memory Data Path Error Injection Mask Low Register (DATA_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 12-54. 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 low 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.

12.5.39 DDR SDRAM Memory Data Path Error Injection Mask ECC Register (ERR_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/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**
	—						EMB	EIEN	EEIM							
Type	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.

Table 12-55. 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 will be cleared by hardware after a single address parity error has been injected.	0 Address parity error injection disabled. 1 Address parity error injection enabled.
— 15–10	0	Reserved. Write to zero for future compatibility.	
EMB 9	0	**ECC Mirror Byte** Enables/disables mirror byte functionality.	0 Mirror byte functionality disabled. 1 Mirror the most significant data path byte onto the ECC byte
EIEN 8	0	**Error Injection Enable** Enables/disables injection of errors. Note that error injection should not be enabled until the memory controller has been enabled via DDR_SDRAM_CFG[MEM_EN]	0 Error injection disabled. 1 Error injection enabled. This applies to the data mask bits and to the ECC mask bits.
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.	

12.5.40 DDR SDRAM Memory Data Path Read Capture Data High Register (CAPTURE_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 register stores the high 32 bits of the read data path during error capture.

Table 12-56. 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.

12.5.41 DDR SDRAM Memory Data Path Read Capture Data Low Register (CAPTURE_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 register stores the low 32 bits of the read data path during error capture.

Table 12-57. 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.

12.5.42 DDR SDRAM Memory Data Path Read Capture ECC Register (CAPTURE_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
	—															
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**
	ECE															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAPTURE_ECC register stores the ECC syndrome bits on the data bus when an error is detected.

Table 12-58. CAPTURE_ECC Bit Descriptions

Bit	Reset	Description
— 31–16	0	Reserved. Write to zero for future compatibility.
ECE 15–0	0	**Error Capture ECC** Captures the ECC bits on the data path when errors are detected. The high order byte represents the 8-bit ECC code for the first 32 bits and the low order byte represents the 8-bit ECC code for the second 32 bits.

12.5.43 DDR SDRAM Memory Error Detect Register (ERR_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/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**
	—							APE	ACE	—		CDE	MBE	SBE	—	MSE
Type	R/W							W1C	W1C	R/W		W1C	W1C		R/W	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 register 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 12-59. 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 not detected. 1 Multiple memory errors 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 Error not detected. 1 Error detected.
— 6–5	0	Reserved. Write to zero for future compatibility.	
CDE 4	0	**Corrupted Data Error** This bit is set if the actual ECC value is inverted compared to the expected value.during a read command. The memory controller inverts the ECC code if the DDR_SDRAM_CFG2[CD_DIS] bit is cleared when corrupted data is written to memory. The ERR_DETECT[MBE] bit is also set when a corrupted data error is detected. It is possible for a 2-bit data error to cause the ECC code to become inverted, which would either mask a corrupted data error or cause a normal multi-bit error to also appear as a corrupted data error.	0 Corrupted data error not detected. 1 Corrupted data error detected.
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 greater than the threshold set in ERR_SBE[SBET]. This bit is cleared by software writing a 1 to it.	0 The number of errors has not crossed the threshold. 1 The number of errors has crossed 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.

12.5.44 DDR SDRAM Memory Error Disable Register (ERR_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/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**
	—							APED	ACED	—		CDED	MBED	SBED	—	MSED
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERR_DISABLE register selectively disables the DDR controller error detection circuitry. Disabled errors are not detected or reported.

Table 12-60. 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** Enables/disables address parity errors.	0 Address parity errors are enabled. 1 Address parity errors are disabled.
ACED 7	0	**Automatic Calibration Error Disable** Enables/disables automatic calibration errors detection	0 Calibration errors enabled. 1 Calibration errors disabled.
— 6–5	0	Reserved. Write to zero for future compatibility.	
CDED 4	0	**Corrupted Data Error Disable** Enables/disables corrupted data error detection.	0 Corrupted data error checking is enabled. 1 Corrupted data error checking is disabled.
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.

12.5.45 DDR SDRAM Memory Error Interrupt Enable Register (ERR_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/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**
	—							APEE	ACEE	—		CDEE	MBEE	SBEE	—	MSEE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERR_INT_EN register 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 12-61. 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.
— 6–5	0	Reserved. Write to zero for future compatibility.	
CDEE 4	0	**Corrupted Data Error Interrupt Enable** Specifies whether corrupted data errors generate interrupts.	0 Corrupted data errors cannot generate interrupts. 1 Corrupted data errors generate interrupts.
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.

12.5.46 DDR SDRAM Memory Error Attributes Capture Register (CAPTURE_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 register sets attributes for errors including type, size, source, and so on.

Table 12-62. 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 8 beats. 001 1 beat. 010 2 beats. 011 3 beats. 100 4 beats. 101 5 beats. 110 6 beats. 111 7 beats.
— 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

12.5.47 DDR SDRAM Memory Error Address Capture Register (CAPTURE_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 register holds the 32-bit of the transaction address when a DDR ECC error is detected.

Table 12-63. 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.	

12.5.48 DDR SDRAM Single-Bit ECC Memory Error Management Register (ERR_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 register stores the threshold value for reporting single-bit errors and the number of single-bit errors counted since the last error report. When the counter field reaches the threshold, it wraps back to the reset value (0). If necessary, software must clear the counter after it has managed the error.

Table 12-64. 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 cross the SBET value. SBEC is automatically cleared when the threshold value is reached.

13 Interrupt Handling

This chapter describes interrupt handling at the device level. All interrupts that are connected to the different processing cores in the system (SC3850 DSP cores and QUICC Engine and MAPLE RISC cores) and/or to different I/O pins (external events) are included in this chapter along with the device level interrupt programming model and functionality.

Note: Detailed explanations and descriptions of the different interrupts are not provided in this chapter. Refer to the chapter for the specific module that is the interrupt source. For example, for details about CPRI interrupts, refer to **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*.

The MSC8157E interrupt system is optimized for a multi-processing environment, routing each interrupt source to each of the SC3850 cores with programmable masking by core, which allows the following:

- Flexible resource allocation as well as a symmetrical or a non-symmetrical application architecture.
- Core-to-core full interrupt mesh.
- External host-to-core signalling mechanism through virtual interrupt generation.

In addition to the SC3850 core interrupt capability, the MSC8157E also includes the following:

- Routes CPRI interrupts to the MAPLE-B block with programmable masking of each interrupt source for the MAPLE block.
- Routes SEC and Serial RapidIO interrupts directly to the QUICC Engine processors with programmable masking of the each interrupt source for the QUICC Engine processors.

The MSC8157E supports both internal and external interrupt sources as well as allowing for the generation of an interrupt to external devices.

There are five device level interrupt handlers in the MSC8157E:

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 SC3850 cores.
3. *Embedded programmable interrupt controller (EPIC)*. Concentrates all the interrupt directed at the associated core and dispatches the highest priority interrupt to the SC3850 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. 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.
4. *QUICC Engine interrupt controller*. The QUICC Engine module includes an interrupt controller that handles interrupts within the module for the dual-RISC processor system. The user can configure this controller, using general configuration registers, to receive either the SEC primary interrupt or the Serial RapidIO eMSG Queue Manager Receive/Transmit interrupts. The Serial RapidIO eMSG Queue Manager Receive/Transmit interrupts are doorbells, and used for messaging filtering or offloading the core. The SEC interrupt is also used for offloading work from the core. For details about that interrupt controller and how to program it, refer to **Chapter 19**, *QUICC Engine Subsystem* and the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM).
5. *MAPLE-B2 interrupt controller*. The MAPLE-B2 includes an interrupt controller that handles interrupts within the module for the quad-RISC processor system. The interrupts from the CPRI controllers that indicate start of frame are connected to the Chip Rate Processing Element Down Link module for synchronization between the CPRI and CRPE modules. For details about that interrupt controller and how to program it, refer to **Chapter 26**, *Multi Accelerator Platform Engine, Baseband 2 (MAPLE-B2)*.

Part of the device interrupts are directed to the performance monitor module for counting and used in debug scenarios. For details about these use cases and connectivity, refer to **Chapter 25**, *Debugging, Profiling, and Performance Monitoring*.

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 SC3850 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 SC3850
VIRQ_1	Connected to Virtual Interrupt 1 at SC3850
VIRQ_2	Connected to Virtual Interrupt 2 at SC3850
VIRQ_3	Connected to Virtual Interrupt 3 at SC3850
VIRQ_4	Connected to Virtual Interrupt 4 at SC3850
VIRQ_5	Connected to Virtual Interrupt 5 at SC3850
VIRQ_6	Connected to Virtual Interrupt 6 at SC3850
VIRQ_7	Connected to Virtual Interrupt 7 at SC3850
VIRQ_8	Connected to Virtual Interrupt 8 at SC3850
VIRQ_9	Connected to Virtual Interrupt 9 at SC3850
VIRQ_10	Connected to Virtual Interrupt 10 at SC3850
VIRQ_11	Connected to Virtual Interrupt 11 at SC3850
VIRQ_12	Connected to Virtual Interrupt 12 at SC3850
VIRQ_13	Connected to Virtual Interrupt 13 at SC3850
VIRQ_14	Connected to Virtual Interrupt 14 at SC3850
VIRQ_15	Connected to Virtual Interrupt 15 at SC3850
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 **13.2.3**, *External Interrupts*)
VIRQ_25	Use to generate $\overline{\text{NMI\_OUT}}$ (see **13.2.3**, *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 MSC8157E before they reach the SC3850 EPICs. These services include:

- Generating ORed interrupt signals towards the SC3850 cores (see **Section 13.2.1**).
- Providing an interrupt enable bit for each interrupt source for each SC3850 core (see **Section 13.5.2**, *General Interrupt Configuration*, on page 13-29).
- Providing a status bit for each interrupt source. These bits are shared for all the SC3850 cores (see **Section 13.5.2**, *General Interrupt Configuration*, on page 13-29).

The general configuration block also performs services for interrupts generated by the CPRI controllers before they reach the MAPLE-B2 engine. These services include:

- Providing an interrupt enable bit for each interrupt source.
- Providing a status bit for each interrupt source.

The general configuration block also performs services for interrupts generated by the Serial RapidIO controllers and SEC before they reach the QUICC Engine processors. These services include:

- Generating an ORed interrupt signal towards the QUICC Engine dual-RISC processor system.
- Providing an interrupt enable bit for each interrupt source.
- Providing a status bit for each interrupt source.

13.2.1 Interrupt Groups Toward the SC3850 Cores

The general configuration block generates 11 interrupts based on the groups of ORed interrupts described in **Table 13-2**.

Table 13-2. SC3850 Core General Configuration Block Interrupt Sources

CPRI Rx Ctl	CPRI TX Ctl	CPRI Rx Frame Timing	CPRI Tx Frame Timing	Debug	SRIO eMSG Buffer Mgr	SRIO eMSG Rx Queue Mgr	SRIO eMSG Tx Queue Mgr	General	MAPLE-B2	Watch Dog Timer
CPRI1	CPRI1	CPRI1	CPRI1	CLASS0 overrun	BM0	QM0	QM0	CPRI	MAPLE general error	Watch Dog Timer 0
CPRI2	CPRI2	CPRI2	CPRI2	CLASS0 watch point	BM1	QM1	QM1	SRIO/ eMSG		Watch Dog Timer 1
CPRI3	CPRI3	CPRI3	CPRI3	CLASS0 error	BM2	QM2	QM2	BM		Watch Dog Timer 2
CPRI4	CPRI4	CPRI4	CPRI4	CLASS1 overrun	BM3	QM3	QM3	QM		Watch Dog Timer 3
CPRI5	CPRI5	CPRI5	CPRI5	CLASS1 watch point	BM4	QM4	QM4	eMSG multi-bit error		Watch Dog Timer 4
CPRI6	CPRI6	CPRI6	CPRI6	CLASS1 error	BM5	QM5	QM5	SRIO1 multi-bit error		Watch Dog Timer 5
				Perf. Mon. all	BM6	QM6	QM6	SRIO2 multi-bit error		Watch Dog Timer 6
				Core0–1 MEX addr error	BM7	QM7	QM7	PCI Exp multi-bit error		Watch Dog Timer 7
				Core2–3 MEX addr error				PCI Exp internal multi-bit error		
				Core4–5 MEX addr error				QUICC Engine IMEM multi-bit error		
								DMA		
								OCN0 to MBus		
								OCN1 to MBus		
								MAPLE multi-bit error		
								DDR		
								OCN0 DMA multi-bit error		
								OCN1 DMA double error		

13.2.2 Interrupt Groups Toward QUICC Engine Processors

The general configuration block generates 4 interrupts based on the groups of ORed interrupts described in **Table 13-3**. These interrupts are multiplexed with the SEC primary interrupt. The multiplexer is controlled by General Configuration Registers CPCE1R to CPCE4R (see **Section 8.2.16**, *QUICC Engine First External Request Multiplex Register (CPCE1R)*, on page 8-27.and the following three subsections).

Table 13-3. QUICC Engine General Configuration Block Interrupt Sources

SRIO eMSG Rx Interrupt Source #1	SRIO eMSG Rx Interrupt Source #2	SRIO eMSG Tx Interrupt Source #1	SRIO eMSG Tx Interrupt Source #2
QM0	QM0	QM0	QM0
QM1	QM1	QM1	QM1
QM2	QM2	QM2	QM2
QM3	QM3	QM3	QM3
QM4	QM4	QM4	QM4
QM5	QM5	QM5	QM5
QM6	QM6	QM6	QM6
QM7	QM7	QM7	QM7

Note: The provided redundancy allows the user to configure which Queue Manager Interrupt to receive. For example, the general SRIO eMSG Rx Interrupt Source #1 can be configured to pass on only the interrupt for Queue Manager 2, while SRIO eMSG Rx Interrupt Source #2 can be configured to pass on only the interrupt for Queue Manager 4.

13.2.3 External Interrupts

The MSC8157E allows a number of external interrupt inputs to be multiplexed with the GPIO signals to enable external devices to interrupt the cores (see **Chapter 21**, *GPIO*). There are also dedicated external interrupt pins.

Note: All external $\overline{\text{IRQ}}$ signals are multiplexed options of the associated GPIO ports.

Table 13-4 summarizes all the external interrupt inputs in the MSC8157E.

Table 13-4. MSC8157E External Interrupt Pins

Name	GPIO	Direction
$\overline{\text{NMI}}$	N/A	In
$\overline{\text{NMI\_OUT}}/\overline{\text{CP\_RX\_INT}}$	N/A	Out
$\overline{\text{INT\_OUT}}/\overline{\text{CP\_TX\_INT}}$	N/A	Out
$\overline{\text{IRQ0}}$	GPIO0	In
$\overline{\text{IRQ1}}$	GPIO1	In

Table 13-4. MSC8157E External Interrupt Pins (Continued)

Name	GPIO	Direction
$\overline{\text{IRQ2}}$	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. $\overline{\text{CP\_RX\_INT}}$ is a CPRI Receive Interrupt asserted by the CPRI controller that is constantly ORed with $\overline{\text{NMI\_OUT}}$. $\overline{\text{CP\_TX\_INT}}$ is a CPRI Transmit Interrupt asserted by the CPRI controller that is constantly ORed with $\overline{\text{INT\_OUT}}$. See **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*for details.

13.2.4 Interrupt Groups Directed Toward MAPLE-B2

The general configuration block generates three interrupts based on the CPRI interrupts 1 to 3. Each interrupt is sampled and can be masked separately through GIR8 and MAPLE_EXT_REQ_EN_1 (see **Chapter 8**, *General Configuration Registers*). There are three interrupts that connect directly to the MAPLE CRPE, as follows:

- CPRI1 indicates a sub-slot (256 chips).
- CPRI2 indicates a frame.
- CPRI3 indicates a chunk (16 chips).

The use of these interrupts is described in detail in **Chapter 26**, *Multi Accelerator Platform Engine, Baseband 2 (MAPLE-B2)*. These are the same three interrupts that connect directly between the CPRI and the SC3850 cores, as described in **Section 13.3**, *Interrupt Mapping*, on page 13-9.

13.2.5 Interrupt Handling

The MSC8157E 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 DMA channel 0 EOB 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, a serial RapidIO eMSG Buffer Manager Interrupt).

Figure 13-1 outlines the flow for handling the various types of interrupts.

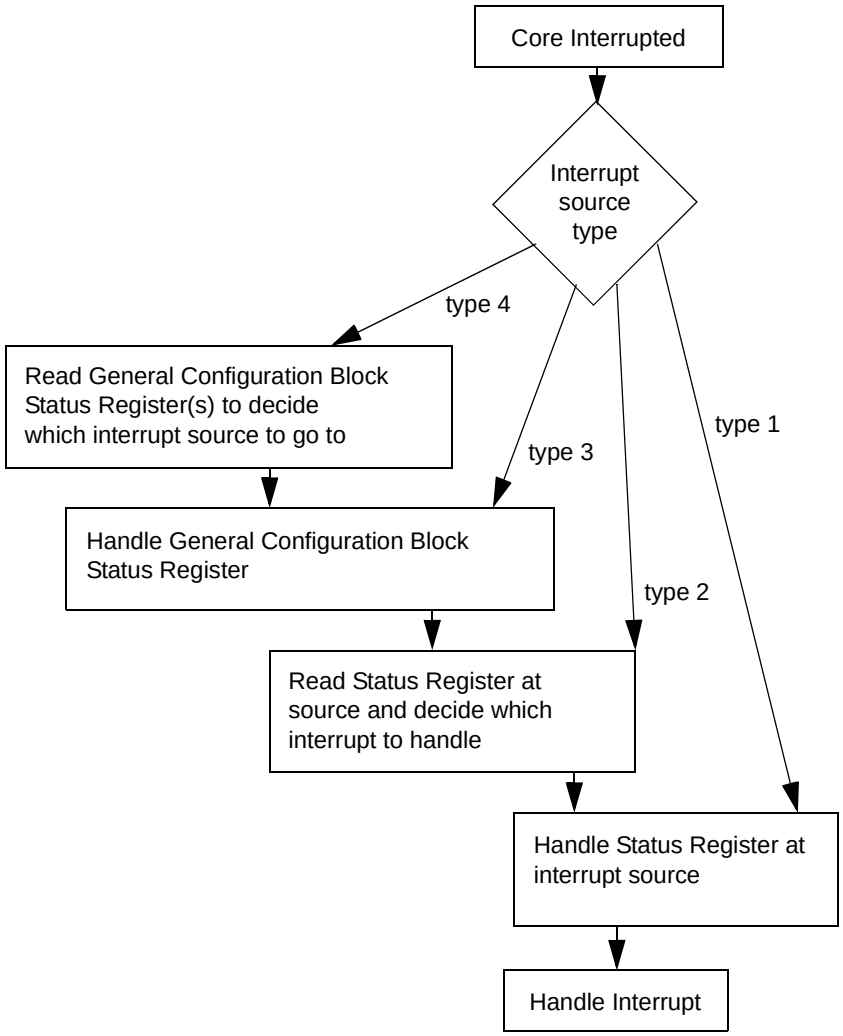


Figure 13-1. Interrupt Handling Flow

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 multi-bit error
- IMEM multi-bit 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 multi-bit error and IMEM multi-bit 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 13-5**. Details about the QUICC Engine interrupt controller and how to use this interrupts is provided in **Section 19.5** 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 MSC8157E device. The MSC8157E does not implement all of these possible sources.

Table 13-5 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 13-5. MSC8157E 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 13-5. MSC8157E 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	+	–
MAPLE-B2			
MAPLE BD 16	50	+	+
MAPLE BD 17	51	+	+
MAPLE BD 18	52	+	+
MAPLE BD 19	53	+	+
MAPLE BD 20	54	+	+
MAPLE BD 21	55	+	+
MAPLE BD 22	56	+	+
MAPLE BD 23	57	+	+
MAPLE BD 24	58	+	+
MAPLE BD 25	59	+	+
MAPLE BD 26	60	+	+
MAPLE BD 27	61	+	+
MAPLE BD 28	62	+	+
MAPLE BD 29	63	+	+
MAPLE BD 30	64	+	+
MAPLE BD 31	65	+	+
CPRI			
CPRI Interrupt 1	66	+	+
CPRI Interrupt 2	67	+	+
CPRI Interrupt 3	68	+	+
CPRI Interrupt 4	69	+	+
CPRI Interrupt 5	70	+	+
CPRI Interrupt 6	71	+	+
CPRI Interrupt 7	72	+	+
CPRI Interrupt 8	73	+	+
CPRI Interrupt 9	74	+	+
CPRI Interrupt 10	75	+	+
CPRI Interrupt 11	76	+	+
CPRI Interrupt 12	77	+	+

Table 13-5. MSC8157E Interrupt Table (Continued)

Interrupt Description	IRQ index	Level	Edge
Timer 32b 0			
Timer 32b 0 Channel 0	78	+	—
Timer 32b 0 Channel 1	79	+	—
Timer 32b 0 Channel 2	80	+	—
Timer 32b 0 Channel 3	81	+	—
Timer 32b 1			
Timer 32b 1 Channel 0	82	+	—
Timer 32b 1 Channel 1	83	+	—
Timer 32b 1 Channel 2	84	+	—
Timer 32b 1 Channel 3	85	+	—
General Configuration			
ORed Serial RapidIO eMSG Queue Manager Receive Interrupt	86	+	—
ORed Serial RapidIO eMSG Queue Manager Transmit Interrupt	87	+	—
ORed Serial RapidIO eMSG Buffer Manager Interrupt	88	+	—
ORed CPRI Receive Control Interrupt	89	+	—
ORed CPRI Transmit Control Interrupt	90	+	—
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	+	—
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	+	—

Table 13-5. MSC8157E Interrupt Table (Continued)

Interrupt Description	IRQ index	Level	Edge
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 7	132	+	—
QUICC Engine interrupt output 6	133	+	—
QUICC Engine interrupt output 5	134	+	—
QUICC Engine interrupt output 4	135	+	—
QUICC Engine interrupt output 3	136	+	—
QUICC Engine interrupt output 2	137	+	—
QUICC Engine interrupt output 1	138	+	—
QUICC Engine interrupt output 0	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	+	—

Table 13-5. MSC8157E Interrupt Table (Continued)

Interrupt Description	IRQ index	Level	Edge
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	+	—
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	—	+

Table 13-5. MSC8157E Interrupt Table (Continued)

Interrupt Description	IRQ index	Level	Edge
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	+	—
Channel 2 Interrupt	205	+	—
Channel 3 Interrupt	206	+	—
I^2C			
I^2C all	208	+	—
MAPLE-B2			
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	+	+

Table 13-5. MSC8157E Interrupt Table (Continued)

Interrupt Description	IRQ index	Level	Edge
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	+	+
$\overline{\text{NMI}}$ (see note)	242	+	+
General Configuration Block			
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	+	—
General Configuration			
ORed CPRI Receive Frame Timing Interrupt	253	+	—
ORed CPRI Transmit Frame Timing Interrupt	254	+	—

Note: For $\overline{\text{NMI}}$s and $\overline{\text{IRQ}}$s, 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 13-6. 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
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

Table 13-6. 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_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
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

Table 13-6. 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
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	From Core Subsystem 4	50	0x32	42	0x2A	1184	0x4A0	1	16 B
IRQ43	From Core Subsystem 4	51	0x33	43	0x2B	1200	0x4B0	1	16 B
IRQ44	From Core Subsystem 5	52	0x34	44	0x2C	1216	0x4C0	1	16 B
IRQ45	From Core Subsystem 5	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	MAPLE BD 16	58	0x3A	50	0x32	1312	0x520	1	16 B
IRQ51	MAPLE BD 17	59	0x3B	51	0x33	1328	0x530	1	16 B
IRQ52	MAPLE BD 18	60	0x3C	52	0x34	1344	0x540	1	16 B
IRQ53	MAPLE BD 19	61	0x3D	53	0x35	1360	0x550	1	16 B
IRQ54	MAPLE BD 20	62	0x3E	54	0x36	1376	0x560	1	16 B
IRQ55	MAPLE BD 21	63	0x3F	55	0x37	1392	0x570	1	16 B
IRQ56	MAPLE BD 22	64	0x40	56	0x38	1408	0x580	1	16 B
IRQ57	MAPLE BD 23	65	0x41	57	0x39	1424	0x590	1	16 B
IRQ58	MAPLE BD 24	66	0x42	58	0x3A	1440	0x5A0	1	16 B
IRQ59	MAPLE BD 25	67	0x43	59	0x3B	1456	0x5B0	1	16 B
IRQ60	MAPLE BD 26	68	0x44	60	0x3C	1472	0x5C0	1	16 B

Table 13-6. 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
IRQ61	MAPLE BD 27	69	0x45	61	0x3D	1488	0x5D0	1	16 B
IRQ62	MAPLE BD 28	70	0x46	62	0x3E	1504	0x5E0	1	16 B
IRQ63	MAPLE BD 29	71	0x47	63	0x3F	1520	0x5F0	1	16 B
IRQ64	MAPLE BD 30	72	0x48	64	0x40	1536	0x600	1	16 B
IRQ65	MAPLE BD 31	73	0x49	65	0x41	1552	0x610	1	16 B
IRQ66	CPRI Interrupt 1	74	0x4A	66	0x42	1568	0x620	1	16 B
IRQ67	CPRI Interrupt 2	75	0x4B	67	0x43	1584	0x630	1	16 B
IRQ68	CPRI Interrupt 3	76	0x4C	68	0x44	1600	0x640	1	16 B
IRQ69	CPRI Interrupt 4	77	0x4D	69	0x45	1616	0x650	1	16 B
IRQ70	CPRI Interrupt 5	78	0x4E	70	0x46	1632	0x660	1	16 B
IRQ71	CPRI Interrupt 6	79	0x4F	71	0x47	1648	0x670	1	16 B
IRQ72	CPRI Interrupt 7	80	0x50	72	0x48	1664	0x680	1	16 B
IRQ73	CPRI Interrupt 8	81	0x51	73	0x49	1680	0x690	1	16 B
IRQ74	CPRI Interrupt 9	82	0x52	74	0x4A	1696	0x6A0	1	16 B
IRQ75	CPRI Interrupt 10	83	0x53	75	0x4B	1712	0x6B0	1	16 B
IRQ76	CPRI Interrupt 11	84	0x54	76	0x4C	1728	0x6C0	1	16 B
IRQ77	CPRI Interrupt 12	85	0x55	77	0x4D	1744	0x6D0	1	16 B
IRQ78	Timer 32b 0 Channel 0	86	0x56	78	0x4E	1760	0x6E0	1	16 B
IRQ79	Timer 32b 0 Channel 1	87	0x57	79	0x4F	1776	0x6F0	1	16 B
IRQ80	Timer 32b 0 Channel 2	88	0x58	80	0x50	1792	0x700	1	16 B
IRQ81	Timer 32b 0 Channel 3	89	0x59	81	0x51	1808	0x710	1	16 B
IRQ82	Timer 32b 1 Channel 0	90	0x5A	82	0x52	1824	0x720	1	16 B
IRQ83	Timer 32b 1 Channel 1	91	0x5B	83	0x53	1840	0x730	1	16 B
IRQ84	Timer 32b 1 Channel 2	92	0x5C	84	0x54	1856	0x740	1	16 B
IRQ85	Timer 32b 1 Channel 3	93	0x5D	85	0x55	1872	0x750	1	16 B
IRQ86	ORed Serial RapidIO eMSG Queue Manager Receive Interrupt	94	0x5E	86	0x56	1888	0x760	1	16 B
IRQ87	ORed Serial RapidIO eMSG Queue Manager Transmit Interrupt	95	0x5F	87	0x57	1904	0x770	1	16 B
IRQ88	ORed Serial RapidIO eMSG Buffer Manager Interrupt	96	0x60	88	0x58	1920	0x780	1	16 B
IRQ89	ORed CPRI Receive Control Interrupt	97	0x61	89	0x59	1936	0x790	1	16 B

Table 13-6. 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
IRQ90	ORed CPRI Transmit Control Interrupt	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
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

Table 13-6. 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
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 7	140	0x8C	132	0x84	2624	0xA40	1	16 B
IRQ133	QUICC Engine interrupt output 6	141	0x8D	133	0x85	2640	0xA50	1	16 B
IRQ134	QUICC Engine interrupt output 5	142	0x8E	134	0x86	2656	0xA60	1	16 B
IRQ135	QUICC Engine interrupt output 4	143	0x8F	135	0x87	2672	0xA70	1	16 B
IRQ136	QUICC Engine interrupt output 3	144	0x90	136	0x88	2688	0xA80	1	16 B
IRQ137	QUICC Engine interrupt output 2	145	0x91	137	0x89	2704	0xA90	1	16 B
IRQ138	QUICC Engine interrupt output 1	146	0x92	138	0x8A	2720	0xAA0	1	16 B
IRQ139	QUICC Engine interrupt output 0	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

Table 13-6. 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
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
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

Table 13-6. 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
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
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

Table 13-6. 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
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
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	—	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

Table 13-6. 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
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	ORed CPRI Receive Frame Timing Interrupt	261	0x105	253	0xFD	4072	0xFE8	0.5	8 B
IRQ254	ORed CPRI Transmit Frame Timing Interrupt	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 MSC8157E 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 13-7** describes the interrupt names connecting the DSP core subsystems:

Table 13-7. 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 MSC8157E 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 13-8. 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 13-9. 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 $\overline{\text{NMI\_OUT}}$/$\overline{\text{CP\_RX\_INT}}$.	0 Interrupt not asserted 1 Interrupt asserted
VS24 24	**Virtual Interrupt 24 Status** Reflects the status of $\overline{\text{INT\_OUT}}$/$\overline{\text{CP\_TX\_INT}}$.	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

Table 13-9. VISR Bit Descriptions (Continued)

Name	Description	Settings
VS16 16	**Virtual Interrupt 16 Status** Reflects the status of VNMI_0.	0 Interrupt not asserted 1 Interrupt asserted
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*. The registers that concentrate and route rare and debug interrupts to the SC3850 cores are:

- General Interrupt Register 1 (GIR1), see **page 8-41**
- General Interrupt Enable Registers 1 for Cores 0–5 (GIER1_[0–5]), see **page 8-41**
- General Interrupt Register 3 (GIR3), see **page 8-42**
- General Interrupt Enable Registers 3 for Cores 0–5 (GIER3_[0–5]), see **page 8-44**
- General Interrupt Register 5 (GIR5), see **page 8-46**
- General Interrupt Enable Registers 5 for Cores 0–5 (GIER5_[0–5]), see **page 8-48**
- General Interrupt Register 6 (GIR6), see **page 8-71**
- General Interrupt Enable Registers 6 for Cores 0–5 (GIER6_[0–5], see **page 8-74**
- General Interrupt Register 7 (GIR7), see **page 8-77**
- General Interrupt Enable Registers 7 for Cores 0–5 (GIER7_[0–5], see **page 8-79**

The registers that route CPRI interrupts to the MAPLE-B2 are:

- General Interrupt Register 8 (GIR8), see **page 8-91**
- CPRI Interrupt to MAPLE External Request Enable (MAPLE_EXT_REQ_EN_1), see **page 8-95**

The registers that concentrate, multiplex, and route CPRI interrupts to the QUICC Engine module are:

- General Interrupt Register 7 (GIR7), see **page 8-77**
- eMSG to QUICC Engine External Request Enable (CPCEER), see **page 8-84**
- QUICC Engine First External Request Multiplex Register (CPCE1R), see **page 8-27**
- QUICC Engine Second External Request Multiplex Register (CPCE2R), see **page 8-28**
- QUICC Engine Third External Request Multiplex Register (CPCE3R), see **page 8-29**
- QUICC Engine Fourth External Request Multiplex Register (CPCE4R), see **page 8-30**

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 *SC3850 DSP Subsystem Reference Manual*, **Appendix C**: *Error Handling*.

14 Direct Memory Access (DMA) Controller

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 DMAC block diagram.

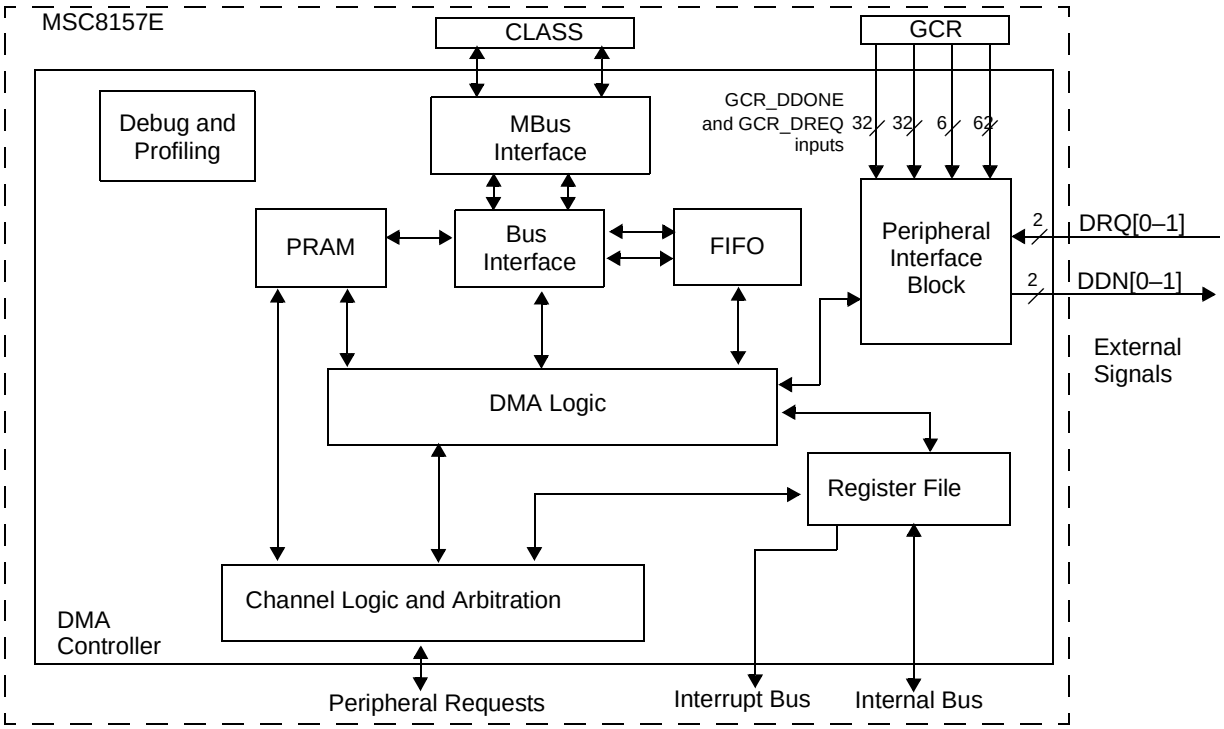


Figure 14-1. DMA Controller Block Diagram

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 DMAC supports memory to memory data transfers.
- *Debug mode*. The DMAC enters the debug by external debug request. Once the DMAC enters the debug mode, it holds its requests to the MBus ports. The internal logic in the DMAC 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 14-28*BD_ATTR Field Descriptions,*** on page 14>-48). **Figure 14-2** shows an example simple buffer.

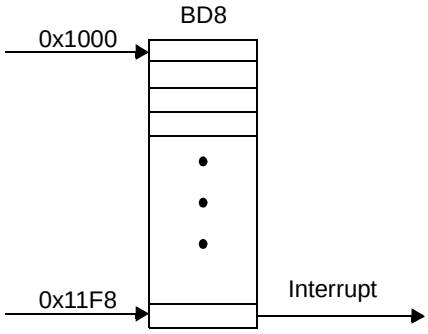


Figure 14-2. One-Dimensional Simple Buffer

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.

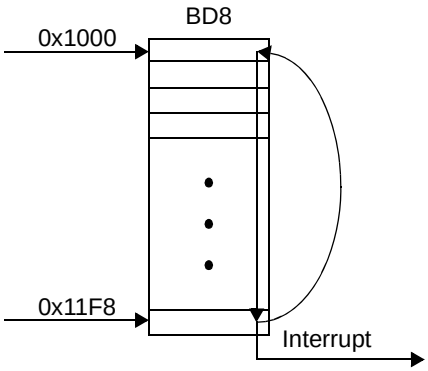


Figure 14-3. One-Dimensional Cyclic Buffer

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.

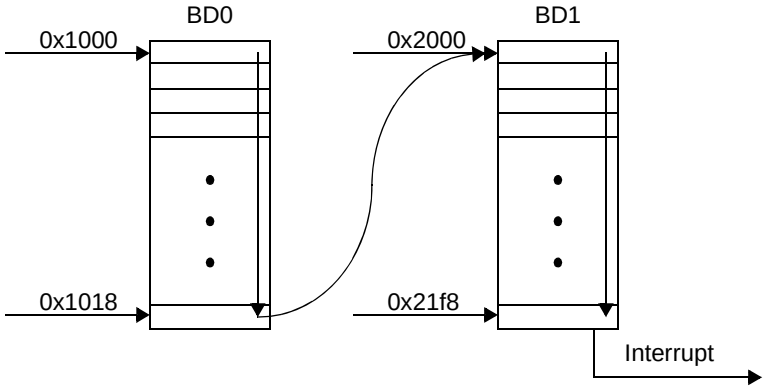


Figure 14-4. One-Dimensional Chained Buffer

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.

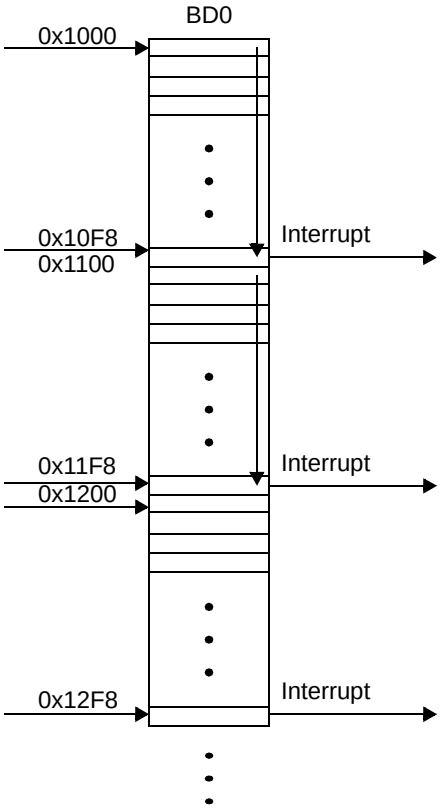


Figure 14-5. One-Dimensional Incremental Buffer

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**.

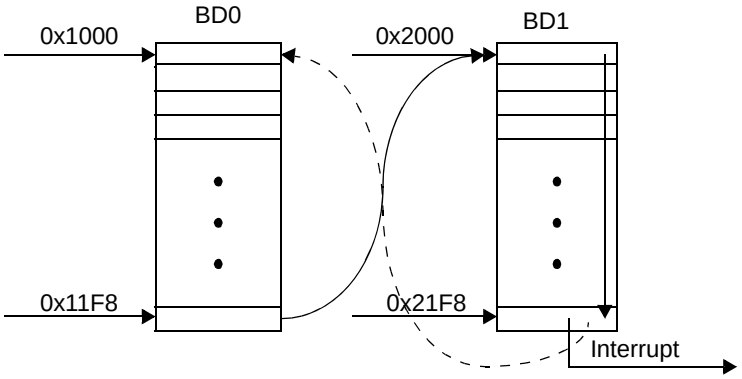


Figure 14-6. Dual Cyclic Buffers

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.

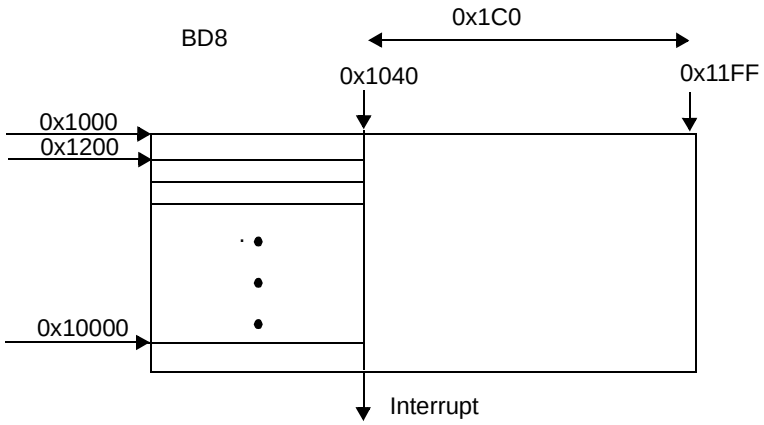


Figure 14-7. Two-Dimensional Simple Buffer

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.

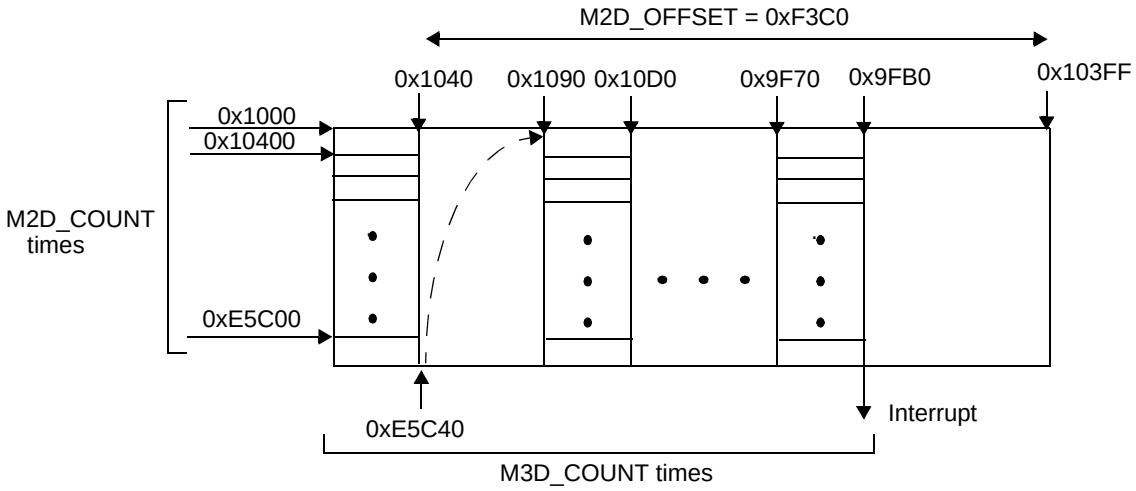


Figure 14-8. Three-Dimensional Simple Buffer

Table 14-7 shows the configuration of a simple buffer designated as channel BD8. A 0x40000 ($0x100 \times 10 \times 40$) 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 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.

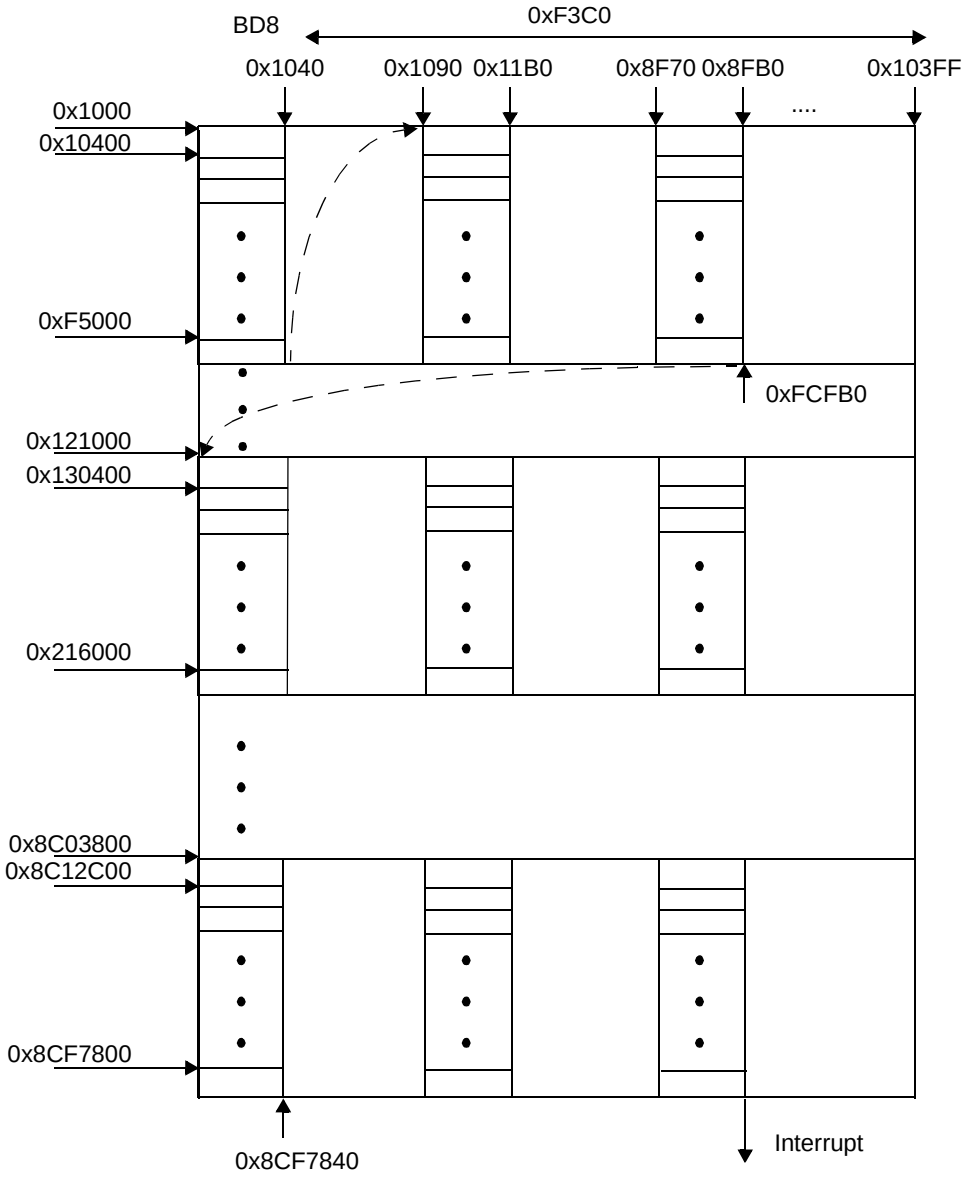


Figure 14-9. Four-Dimensional Simple Buffer

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.

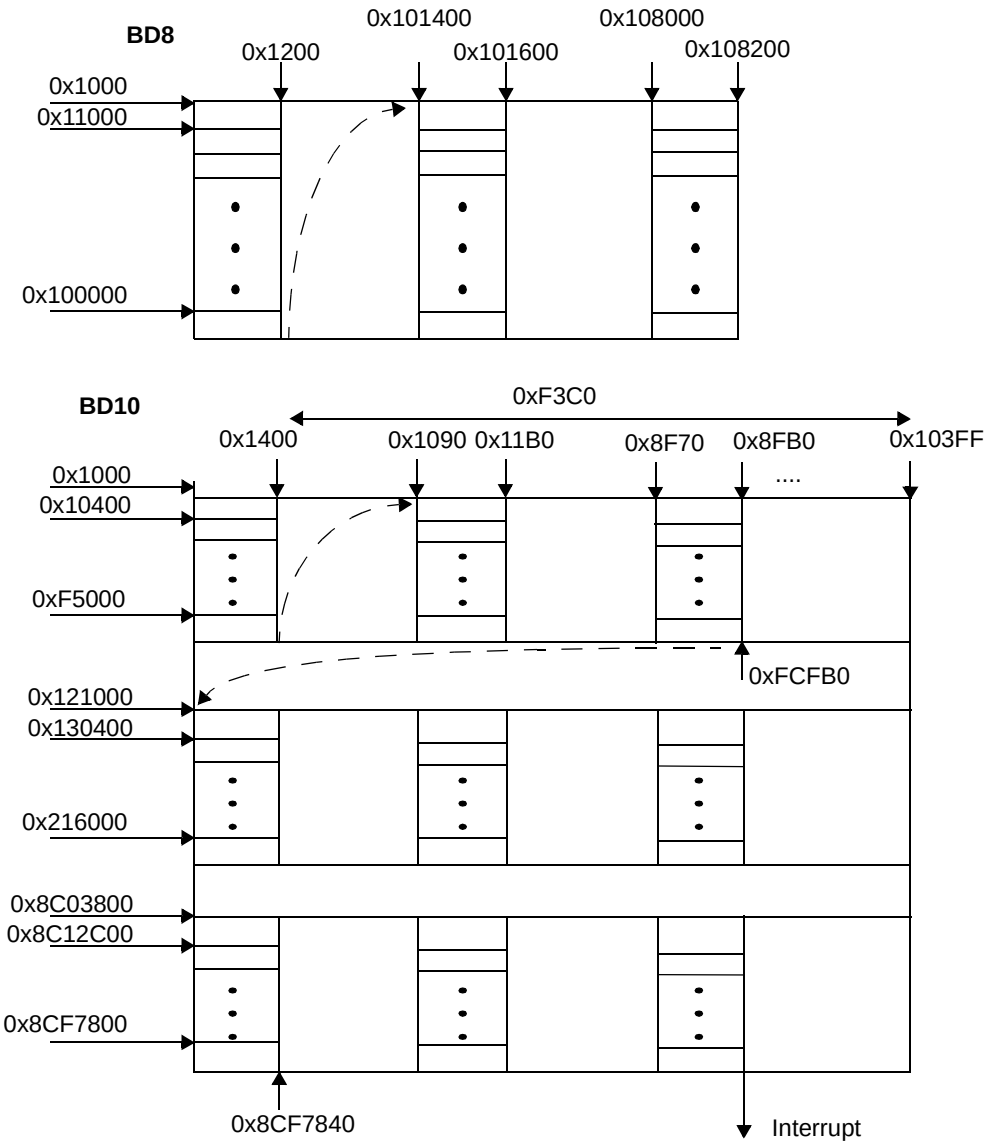


Figure 14-10. Multi-Dimensional Chained Buffer

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.

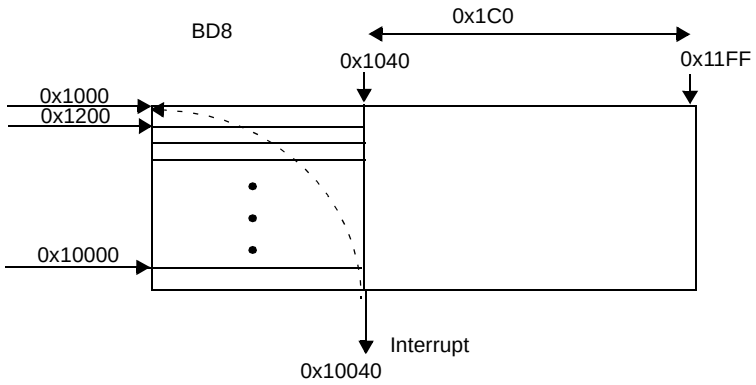


Figure 14-11. Two-Dimensional Cyclic Buffer

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.

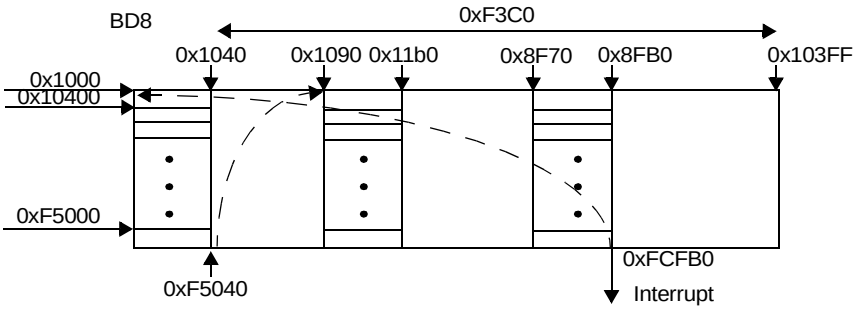


Figure 14-12. Three-Dimensional Cyclic Buffer

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 three clock sources: 2 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 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.

14.5.1 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.

14.5.2 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*.

14.5.3 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.

14.5.3.1 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.

14.5.3.2 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).

14.5.3.3 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.

14.5.4 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 21**, *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 MSC8157E 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.

14.6 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 Round-Robin Priority Group Update Register (DMARRPGUR), **page 14-43**
  - DMA Channel Active Status Register (DMACHASTR), **page 14-44**
  - DMA Channel Freeze Status Register (DMACHFSTR), **page 14-44**
- DMA Channel Buffer Descriptors
  - Buffer Attributes (BD_ATTR), **page 14-48**
  - Multi-Dimensional Buffer Attributes (BD_MD_ATTR), **page 14-51**

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 14.5.4**, *Using the DMA Peripheral Interface Block*, on page 14-26 for details.

14.6.1 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 14.6.21**, *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.

14.6.2 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 14.6.21**, *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 14-43**) **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 14.6.21**. **Written by:** User, DMA controller	

14.6.3 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.

14.6.4 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.

14.6.5 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.

14.6.6 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.

14.6.7 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

14.6.8 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.	

14.6.9 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.	

14.6.10 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.

14.6.11 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.

14.6.12 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.

14.6.13 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 MSC8157E, only unidirectional destination channel interrupts are implemented as described in **Section 14.4.1**.

14.6.14 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	

14.6.15 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.

14.6.16 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 14-27** and **Table 14-29**.	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.

14.6.17 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.

14.6.18 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 14-26. 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	

14.6.19 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.

14.6.20 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

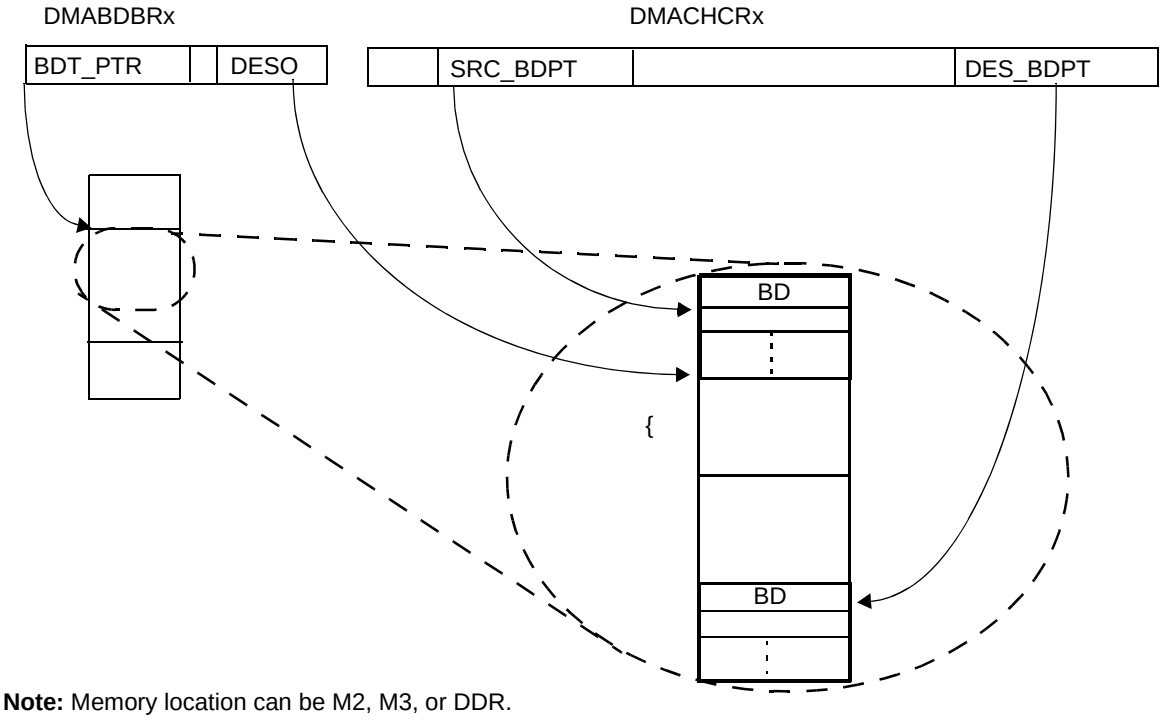
Each bit in the DMACHFSTR corresponds to the freeze status of the associated channel. If set, a bit associated with a channel indicates that the channel is still frozen.

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.

14.6.21 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.

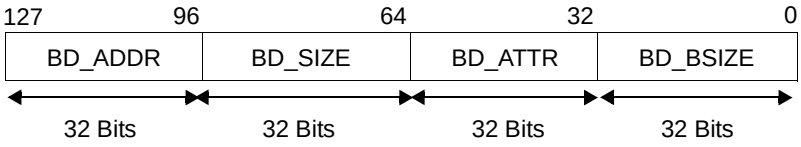


Figure 14-14. DMA Channel BD One-Dimensional Line

Figure 14-15 shows the structure of a multi-dimensional BD, which is a 256-bit entry.

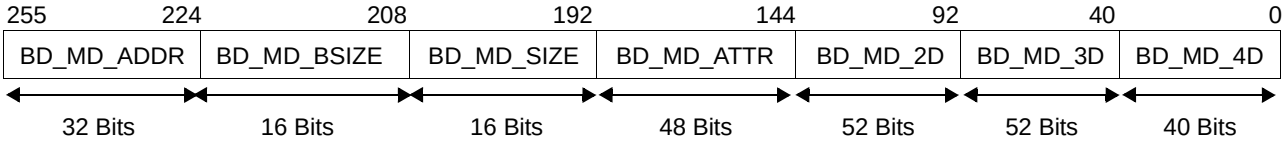


Figure 14-15. DMA Channel Multi-Dimensional Buffer Descriptor

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.

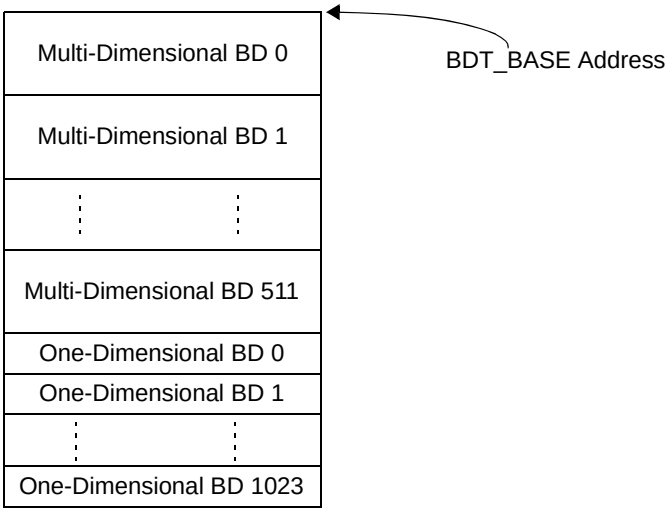


Figure 14-16. Example BD Table with a Mixed Dimensional Structure

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 14-27** lists the channel parameters for a one-dimensional BD.

Table 14-27. 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 14-28**.
BD_BSIZE 31–0	**Buffer Base Size** Holds the base size of the buffer.

14.6.21.1 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 14-28. 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 14-3.	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 14-28. 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. 1100 2048 bytes. 1101 4096 bytes. 111x 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.

Table 14-28. BD_ATTR Field Descriptions (Continued)

Bits	Description	Settings
— 4	Reserved. Write to zero for future compatibility.	
BTSZ 3–0	**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.	0000 64 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 101x reserved 11xx reserved

Table 14-29 shows the DMA channel parameters for a multi-dimensional buffer.

Table 14-29. 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 14-3.
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 14-30**.
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 14-31*BD_MD_2D Field Descriptions,*** on page 14>-54.

Table 14-29. Multi-Dimensional BD Field Descriptions (Continued)

Bits	Description
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 14-32*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 14-33*BD_MD_4D Field Descriptions,*** on page 14>-55.

14.6.21.2 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 14-30. 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 14-3. **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.

Table 14-30. BD_MD_ATTR Field Descriptions (Continued)

Bits	Description	Settings
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.	
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. 1100 2048 bytes. 1101 4096 bytes. 111x Reserved.
— 23	Reserved. Write to zero for future compatibility.	

Table 14-30. BD_MD_ATTR Field Descriptions (Continued)

Bits	Description	Settings
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	Reserved. Write to zero for future compatibility.	
BTSZ 19–16	**Basic Transfer Size** The basic transfer size issued by the request. If BTSZ is greater than TSZ, the DMA controller uses the TSZ value.	0000 64 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 101x reserved 11xx reserved
— 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.

Table 14-30. BD_MD_ATTR Field Descriptions (Continued)

Bits	Description	Settings
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.
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 14-31. 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 14-32. 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 14-33. 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.

15 High Speed Serial Interface (HSSI) Subsystem

The High Speed Serial Interface (HSSI) is a 10-port serial communications subsystem that supports multiplexing of the following serial interfaces:

- Two x1/x2/x4 Serial RapidIO ports
- One x1/x2/x4 PCI Express port
- Two SGMII ports
- Six x1 CPRI ports

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

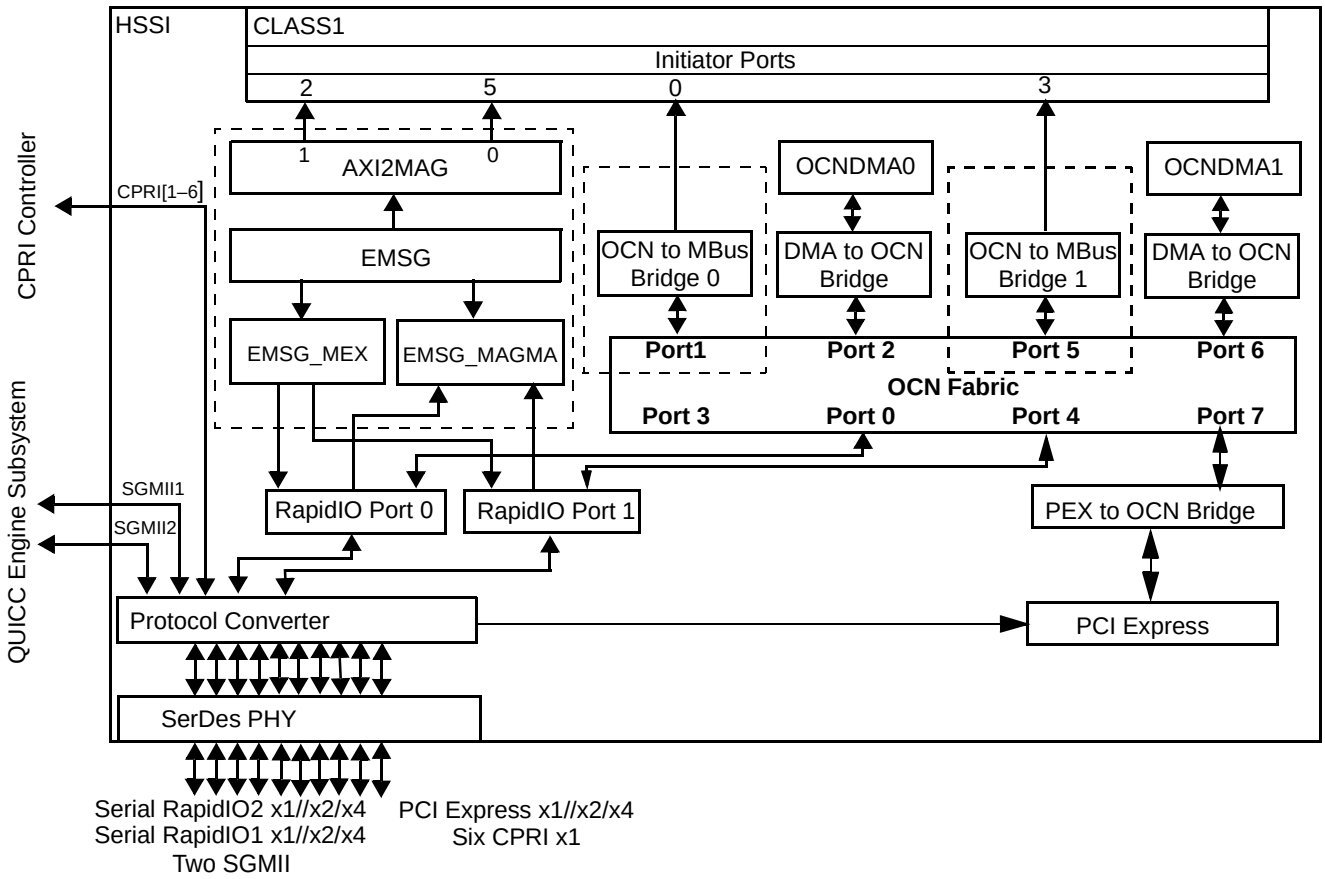
- One CLASS1 non-blocking fabric to interface between the HSSI and the system CLASS module.
- One 8-port OCN fabric with two dedicated DMA controllers and two OCN to MBus bridges to provide high-speed memory access with no core intervention for the two Serial RapidIO port controllers and the PCI Express controller. The bridges connect to the CLASS 1 module.
- Two Serial RapidIO port controllers (port 1 and 0) with enhanced messaging unit (eMSG). The RapidIO complex includes an AXI bridge to MBus to connect to the CLASS1 fabric and an eMSG MBus expander and normalizer to connect to the two Serial RapidIO port controllers. The RapidIO complex and eMSG details are provided in **Chapter 16**, *Serial RapidIO Controller and Enhanced Message Complex*.
- One PCI Express controller with a bridge to the OCN fabric. The PCI Express controller details are provided in **Chapter 17**, *PCI Express Controller*
- Six CPRI controllers, described in detail in **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*.
- One Protocol Converter module to link the two Serial RapidIO ports, one PCI Express port, six CPRI ports and the two SGMII ports to the SerDes PHY.
- One 10-channel SerDes PHY that multiplexes the RapidIO, PCI Express, CPRI and SGMII signals for external connection.

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 includes the interface component programming model, with the exception of the SerDes multiplexing programming, which is done using the SP bits in the low half of the reset configuration word (RCW), as described in **Chapter 5**, *Reset*. The communication controllers supported and eMSG unit within the block are described in **Chapter 16**, *Serial RapidIO Controller and Enhanced Message Complex***.** While the SGMII lines are multiplexed through the SerDes interfaces, they are functionally part of the QUICC Engine subsystem, as described in **Chapter 19**, *QUICC Engine Subsystem*. While the CPRI signals and multiplexed through the SerDes interfaces, the CPRI controller lies outside of HSSI complex, as described in **Chapter 18**, *Common Public Radio Interface (CPRI) Complex*.

15.1 HSSI Subsystem Block Diagram

Figure 15-1 shows a block diagram of the HSSI.



Note: The actual signal multiplexed for each channel in the PHY is determined by the SerDes configuration field contents in the lower 32 bits of the reset configuration word, which are recorded in RCWLR[SP]. See **Chapter 5**, *Reset* for details.

Figure 15-1. HSSI Block Diagram

15.2 CLASS1

Based on the same design as the system CLASS fabric, CLASS1 uses a fully pipelined low latency design and is the interconnect between the HSSI complex and the system CLASS. Like the CLASS, CLASS1 is a non-blocking, full-fabric interconnect that allows any initiator to access any target in parallel with another initiator-target couple. CLASS1 uses per-target prioritized round-robin arbitration, highly optimized to the target characteristics. CLASS1 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 Six CLASS1 initiators are:

- OCN-to-MBus bridge 0 (initiator port 0)
- OCN-to-MBus bridge 1 (initiator port 3)
- AXI-to-MBus bridge 0 (initiator port 5)
- AXI-to-MBus bridge 1 (initiator port 2)
- CPRI MBus port 0 (initiator port 4)
- CPRI MBus port 1 (initiator port 1)

The two CLASS1 targets are:

- HSSI port 0 (target port 1). Connects to CLASS0 initiator port 8.
- HSSI port 1 (target port 2). Connects to CLASS0 initiator port 12.

The CLASS initiators and targets are shown in **Figure 15-2**. The arrows indicate the address direction from initiator to target.

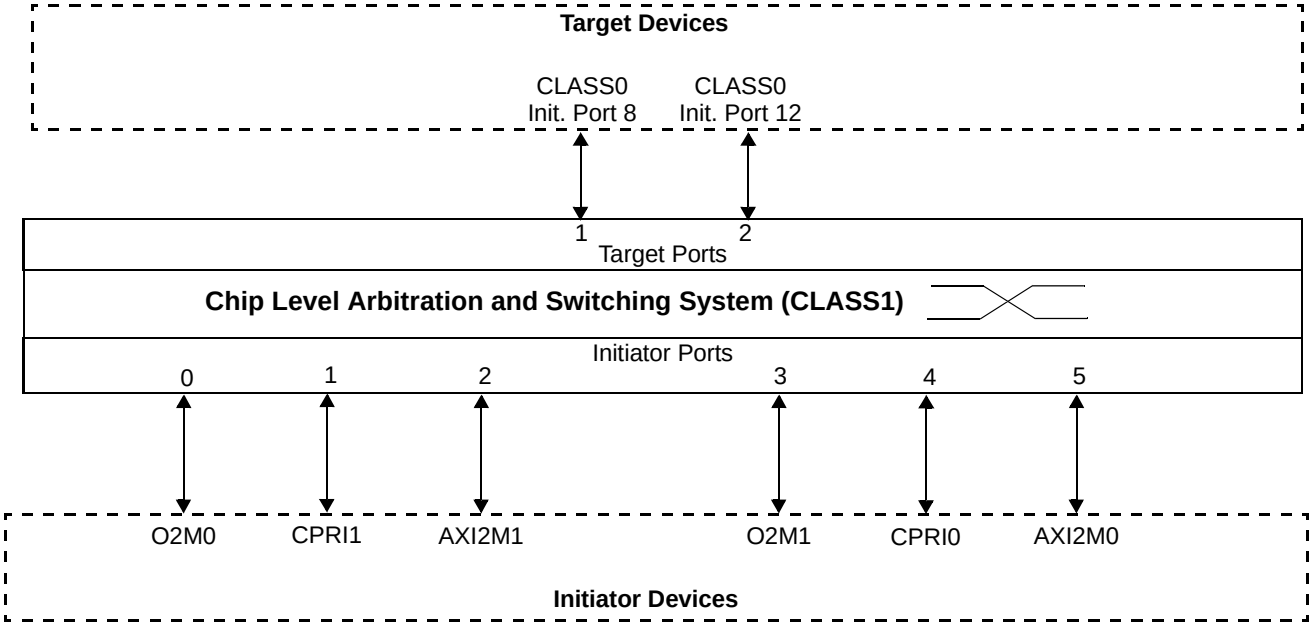


Figure 15-2. CLASS1 Initiators and Targets in the MSC8157E Device

15.2.1 Functional Description

CLASS1 is a non blocking interconnect between 6 initiators and 2 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. Each CLASS module 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 details about the expander module, see **Section 15.2.1.1**. 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 15.2.1.2**. 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 details about normalizer module see **Section 15.2.1.2.3**.

15.2.1.1 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.

15.2.1.2 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.

15.2.1.2.1 CLASS Arbiter

The CLASS arbiter performs weighted arbitration algorithm for requesters 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.

The arbiter supports the following arbitration schemes and functions:

- 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.
- 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. In CLASS1, this mode is activated/deactivated by the appropriate bit in the C1ACR (see **15.10.29**, *CLASS1 Arbitration Control Register (C1ACR)*).
- Priority Masking. In CLASS1, when C1ACR[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.
- Auto Priority Upgrade. In CLASS1, this mode is activated by setting the C1PACRx[AUE] bit (see **15.10.3**, *CLASS1 Priority Auto Upgrade Control Registers (C1PACRx)*). When active, a pending request has its priority upgraded to the next higher priority after a specified number of cycles specified by C1PAVRx[AUV] (see **15.10.2**, *CLASS1 Priority Auto Upgrade Value Registers (C1PAVRx)*). 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.

15.2.1.2.2 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.

15.2.1.2.3 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.

15.2.1.3 CLASS1 Control Interface (C1CI)

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 15.10** for programming details.

15.2.2 CLASS Error Interrupts

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 C1ISR 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 C1IER 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.

15.2.3 CLASS Debug Profiling Unit

CLASS1 supports debug and profiling measurements by the CLASS1 debug and profiling unit (C1DPU).

15.2.3.1 Profiling

The user can configure the desired measurement in the C1IPCRs and C1TPCRs.

Note: For each CLASS1 module, only one PMM field among all C1IPCRx and C1TPCRx can be greater than 0 during profiling.

You can activate the C1DPU by:

- Writing a 1 to the C1PCR[PE] bit.
- Configuring a watch point event in C1PCR[WPEC] field.

The CDPU is deactivated by:

- Writing a 0 to the C1PCR[PE] bit.
- Configuring a watch point event in C1PCR[WPEC] field.
- Reaching a time-out in the C1PTOR when the C1PCR[TOE] bit is set.
- C1PRCR overflow.

After the desired profiling mode has been chosen, activate the C1DPU to perform the measurement. At the beginning of every measurement, the CLASS1 Profiling Reference Counter (C1PRCR) starts counting the clock cycles. Read the C1PISR[OVE] bit to verify that the measurement is complete and that the profiling counter values are valid. If the C1PISR[OVE] is clear, read the profiling counters C1PRCR and C1PGCR and analyze the results.

15.2.3.2 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 CLASS1 module, only one WPEN field can be set among all C1IWPCRx and C1TWPCRx 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 C1PCR.
2. Clear C1IPCRx and C1TPCR.
3. For the time-out mechanism, program C1PTOR and set the C1PCR[TOE] bit.
4. Define the transaction to be monitored by writing the desired configuration to C1WPCR, C1WPACR, C1WPEACR, and C1WPAMR.
5. Enable the watch point units through C1IWPCRx and C1TWPCR.
6. Set the C1WPRCR[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 C1PISR[OVE] bit.
   - In time-out mode, read C1PRCR.
   - If C1PISR[OVE] is set or if C1PRCR is equal to C1PTOR, the results are not valid.
   - Read C1PGCRx to get the number of watch point events during the measurement.

15.2.3.3 Event Selection

Events are selected using a combination of the CLASS watch point and profiling registers. **Table 15-1** 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 15.10** for the register details.

Table 15-1. C1PGCRx Events Selection

Measurement Mode	Configuration Settings for Each Mode				Events Measured			
	C1WPCR [CE]	C1TPCR [TT]	C1TPCR [PMM]	C1IPCRx [PMM]	C1PGCR0	C1PGCR1	C1PGCR2	C1PGCR3
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 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 Target Switch (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 15-1. C1PGCRx Events Selection (Continued)

Measurement Mode	Configuration Settings for Each Mode				Events Measured			
	C1WPCR [CE]	C1TPCR [TT]	C1TPCR [PMM]	C1IPCRx [PMM]	C1PGCR0	C1PGCR1	C1PGCR2	C1PGCR3
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 15-1. C1PGCRx Events Selection (Continued)

Measurement Mode	Configuration Settings for Each Mode				Events Measured			
	C1WPCR [CE]	C1TPCR [TT]	C1TPCR [PMM]	C1IPCRx [PMM]	C1PGCR0	C1PGCR1	C1PGCR2	C1PGCR3
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)							

15.2.3.4 Debug and Profiling Events

The CLASS generates two event interrupts:

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

15.2.4 CLASS Reset

The CLASS implements two kinds of reset:

- Synchronous hard reset. This reset brings all states machines to idle state and sets all CLASS registers to the reset values.
- Synchronous 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 **15.10**, *HSSI Programming Model*.

15.2.5 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.

15.3 OCN Fabric

The On-Chip Network (OCN) fabric is a non-blocking high speed interconnect used for embedded system devices. The MSC8157E DSP HSSI uses an 8-port OCN to connect between the Serial RapidIO Controllers, PCI Express Controller, the two OCN-to-MBus bridges (O2M[0–1]) that connect to the CLASS1 module, and the two supporting dedicated DMA controllers. The OCN requires no programming and provides a seamless interface for the HSSI.

15.4 OCN-to-MBus (O2M) Bridges

The O2M bridges provide an interface between the OCN fabric and the CLASS1 module that connects to the system CLASS. The bridges support all mandatory MBus and OCN interface protocol procedures. The bridges provide initiator interfaces to access the target CLASS ports, formats OCN packets, negotiates with the OCN arbiter, and transmits/receives associated transactions. The O2M bridges are connected to CLASS1 initiator ports 0 and 3, each 128 bits wide and connect to OCN ports 1 and 5.

Note: When using the same OCN-to-MBus (O2M) for Read and Write transactions, the write transactions may write incorrect data for specific access sequences. To preclude this scenario, use one bridge for Write transactions and the other bridge for read transactions.

15.5 DMA Controllers

The MSC8157E includes two dedicated DMA controllers that transfer blocks of data between the serial RapidIO controller/PEX Controller and the local address space independent from the DSP cores. **Figure 15-3** shows the block diagram of each dedicated DMA controller.

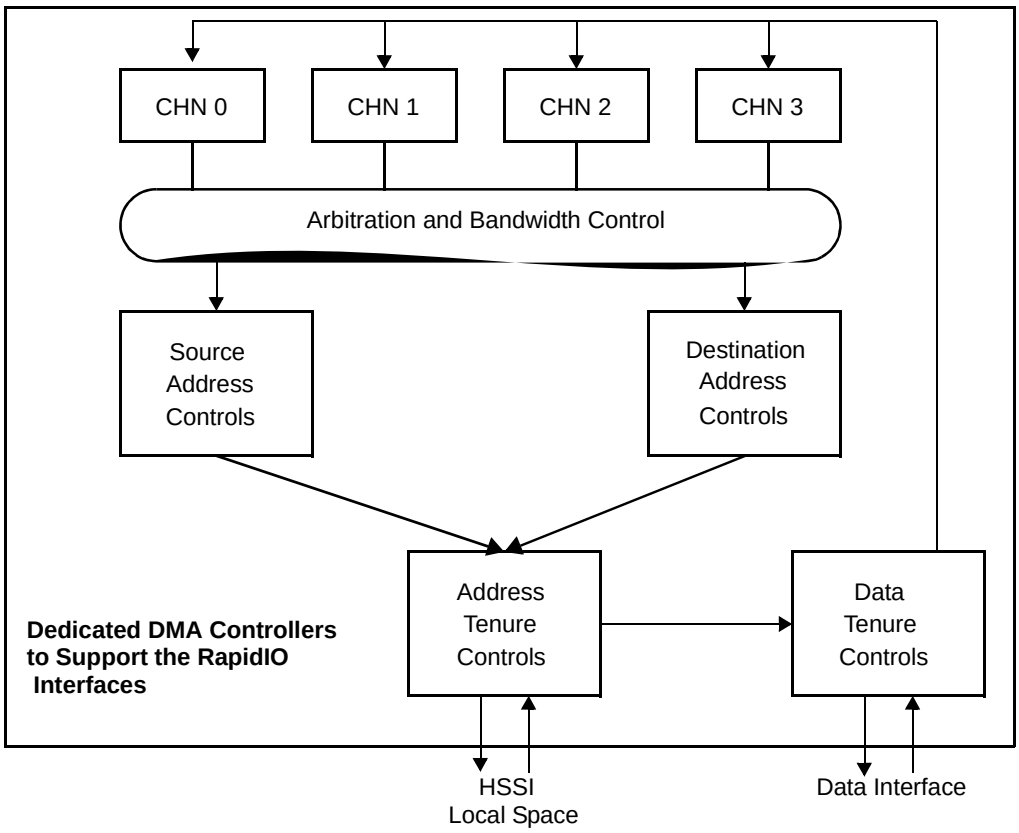


Figure 15-3. Interface Dedicated DMA Controller Block Diagram

15.5.1 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.

15.5.2 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.

15.5.3 Modes of Operation

Each MSC8157E 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 15-2** 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 15-2** before starting a transfer.
- *Chaining mode*. Software must initialize descriptors in memory and the required fields as described in **Table 15-2** 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 15-2. 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 15.5.4**, *DMA Channel Operation* for details on these modes. **Figure 15-4** shows the general DMA operational flow chart.

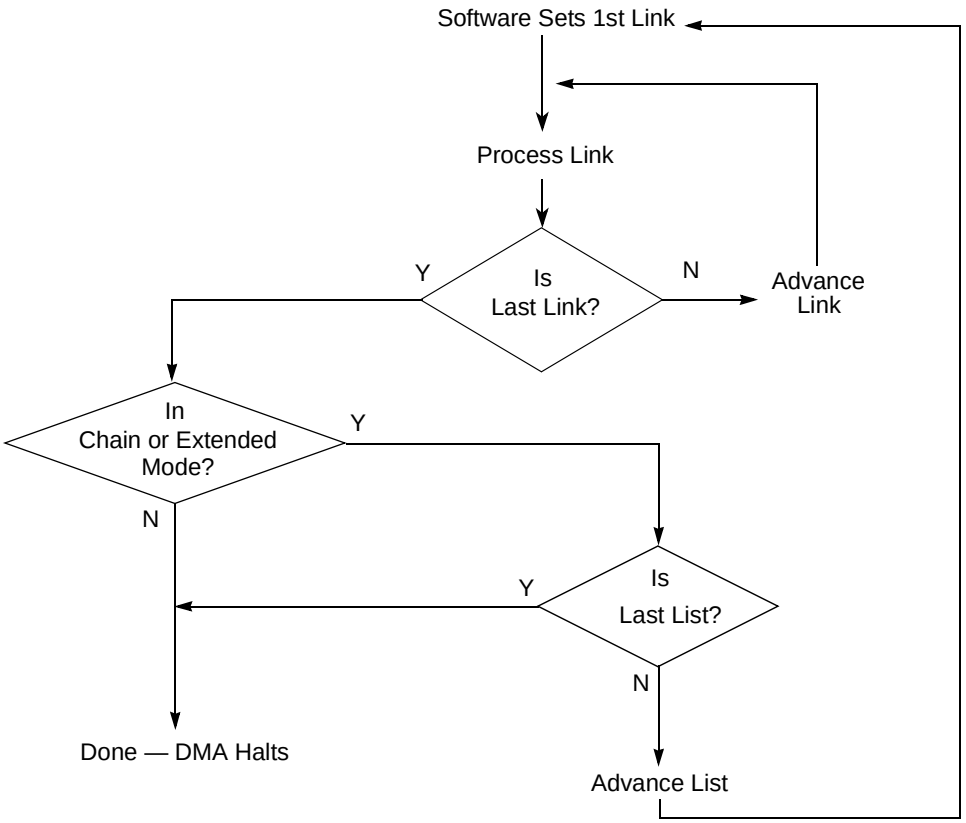


Figure 15-4. DMA Operational Flow Chart

15.5.4 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 align the source and destination addresses to a 64-byte boundary. The DMA always reads/writes the maximum number of bytes for a given transfer as described by the capability inputs of the DMA controller except for globally coherent transactions that use the size of the cache coherence granule as described by the mode select input. Using 256 bytes over the RapidIO interface reduces packet overhead 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 additional features from previous generations in which the reads can be mapped to non-coherent (NREAD), or maintenance reads. In addition, the writes can be mapped to non-coherent (NWRITE, NWRITE_R) writes, messages, 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 15-3** for more complete descriptions of the channel states and state transitions.

15.5.4.1 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.

15.5.4.1.1 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 15-3**), 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.

15.5.4.1.2 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 15-3**), 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.

15.5.4.1.3 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 15-3**), 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.

15.5.4.1.4 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 15-3**), 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.

15.5.4.1.5 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].

15.5.4.1.5.1 Extended Direct Mode

Extended direct mode has the same functionality as basic direct mode with the addition of stride capabilities. The bit settings are the same as in direct mode with the exception of the MR*n*[XFE] being set. Striding on the source address can be accomplished by setting SATR*n*[SSME] and setting the desired stride size and distance in SSR*n*. Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in DSR*n*.

15.5.4.1.5.2 Extended Direct Single-Write Start Mode

Extended direct single-write start mode has the same functionality as the basic direct single-write start mode with the addition of stride capabilities. The bit settings are also the same with the exception of MR*n*[XFE] being set. Striding on the source address can be accomplished by setting SATR*n*[SSME] and setting the desired stride size and distance in SSR*n*. Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in DSR*n*.

15.5.4.1.5.3 Extended Chaining Mode

In extended chaining mode, the software must first build list and link descriptor segments in memory. Then CLSDAR*n* must be initialized to point to the first list descriptor in memory. The DMA controller loads list descriptors and link descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the link descriptor information loaded. Once the current link descriptor is finished, the DMA controller reads the next link descriptor from memory and begins another DMA transfer. If the current link descriptor is the last in the list, the DMA controller reads the next list descriptor in memory. The transfer is finished if the current link descriptor is the last one in the last list in memory or if an error condition occurs. The sequence of events to start and complete a transfer in extended chaining mode is as follows:

1. Build link and list descriptor segments in memory.
2. Poll the channel state (see **Table 15-3**), 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.

15.5.4.1.5.4 Extended Chaining Single-Write Start Mode

In the extended mode, the single-write start feature allows a chain to be started by writing the current list descriptor pointer. Setting MR*n*[CDSM/SWSM] causes MR*n*[CS] to be set automatically when CLSDAR*n* is written. 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 15-3**), 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.

15.5.4.2 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.

15.5.4.2.1 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.

15.5.4.2.2 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.

15.5.4.3 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.

15.5.4.4 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.

15.5.4.5 Channel State

Table 15-3 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 15-3. 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

15.5.4.6 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.

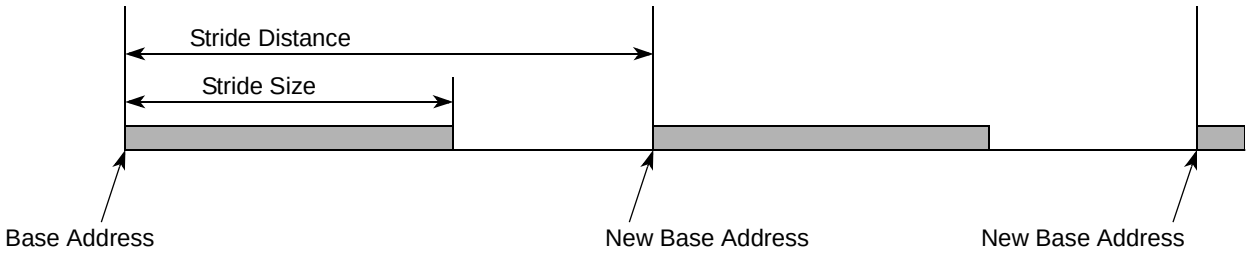


Figure 15-5. Stride Size and Stride Distance

15.5.5 DMA Transfer Interfaces

The DMA can be used to achieve data transfers across the entire memory map.

15.5.6 DMA Errors

On a transfer error (uncorrectable ECC errors on memory accesses, parity errors on MBus, 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.

15.5.7 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. Link and list descriptor fetches always snoop the local memory space.

Note: Software must ensure that each descriptor is aligned on a 32-byte boundary.

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 15-4 summarizes the DMA list descriptors.

Table 15-4. 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 15-5 summarizes the DMA link descriptors.

Table 15-5. Link DMA Descriptor Summary

Descriptor Field	Description
Source attributes register	Contains source transaction attributes
Source address	Contains the source address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the Source address register.
Destination attributes register	Contains destination transaction attributes
Destination address	Contains the destination address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the destination address register.
Next link descriptor extended address	Points to the next link descriptor in memory. After the DMA controller reads the link descriptor from memory, this field is loaded into the extended next link descriptor address registers
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.

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)

Figure 15-7 shows the format of the list descriptors for 36-bit devices.

Offset	Field
0x00	Next List Descriptor Extended Address
0x04	Next List Descriptor Address
0x08	First Link Descriptor Extended Address
0x0c	First Link Descriptor Address
0x10	Source Stride
0x14	Destination Stride
0x18	Reserved
0x1c	Reserved

Figure 15-7. List Descriptor Format (Used for 36-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 15-4** and **Table 15-5**). **Table 15-6** and **Table 15-7** show the order in which data should be stored in memory (for 32-bit or 36-bit devices, 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 15.10**, *HSSI Programming Model* for details.

Figure 15-6 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 15-8. Link Descriptor Format (Used for 32-bit devices

Figure 15-9 shows the format of the link descriptors for 36-bit devices.

Offset	Field
0x00	Source Attributes
0x04	Source Address
0x08	Destination Attributes
0x0c	Destination Address
0x10	Next Link Descriptor Extended Address
0x14	Next Link Descriptor Address
0x18	Byte Count
0x1c	Reserved

Figure 15-9. Link Descriptor Format (Used for 36-bit devices)

15.5.8 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 15.10.48**, *Local Access Window Base Address Registers 0–9 (DnLAWBAR[0–9])*, on page 15-90 and **Section 15.10.49**, *Local Access Window Attributes Registers 0–9 (DnLAWAR[0–9])*, on page 15-91 for details.

15.5.9 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.
- Coherent reads or writes are broken up into cache line accesses in the DMA.
- 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.
- Striding does not work if the destination transaction type is MESSAGE (DATR[DWRITETTYPE] = 0x0110) because messages have no memory addresses. As well, destination address hold should be disabled (MR*n*[DAHE] is cleared) unless the destination address hold transfer size indicates an 8-byte message (MR*n*[DAHTS] = 0x11). Software is responsible for disabling striding and DAHE, in this case, and for ensuring that the bandwidth control is large enough to support the desired message size. Failure to adhere to these restrictions results in undefined behavior.
- A single DMA transfer in any of the direct or chaining modes must not cross a 16 GB (34-bit) address boundary.

15.6 Serial RapidIO Complex

The Serial RapidIO complex includes two ports (SRIO0 and SRIO1) that perform the serial data transfers between the SerDes PHY and the DSP system, an enhanced messaging unit (eMSG), and the interface bridges that connect between SRIO0 and SRIO1, the eMSG module, and the CLASS1 module. Configuration and functional details for this complex are provided in **Chapter 16**, *Serial RapidIO Controller and Enhanced Message Complex*. Transfers through the SRIO modules can be tracked by the performance monitor. See **Section 25.3**, *Performance Monitor* in **Chapter 25**, *Debugging, Profiling, and Performance Monitoring* for details.

15.7 PCI Express Controller

HSSI includes one PCI Express controller that supports x1/x2/x4 link configuration. It is interfaced to the SerDes Phy through Protocol converter module and connects to the OCN fabric through PEX to OCN bridge. The functional and configuration details are provided in **Chapter 17**, *PCI Express Controller***.**

15.8 Protocol Converter

The protocol converter interfaces the Serial RapidIO, PCI Express, CPRI, and SGMII signals to the SerDes PHY. It is programmed internally by selections make in the General Configuration Registers and the power-on reset configuration (see **Section 15.9**, *SerDes PHY Interfaces*). Two of the register sets provide SerDes delay information used by the CPRI modules.

15.9 SerDes PHY Interfaces

The HSSI includes one 10-port SerDes PHY interface that is multiplexed between the two Serial RapidIO ports, one PCI Express port, six CPRI lanes, and the two SGMII ports from the QUICC Engine subsystem. Multiplexing configuration is done using the RCWLR[SP] field (see **Section 5.3.1**, *Reset Configuration Word Low Register (RCWLR)*, on page 5-16) and the HSSI general configuration registers (see **Section 8.2.6** and **Section 8.2.7** in **Chapter 8**, *General Configuration Registers*). Control of the individual lanes is done through the SRDS Control Registers described in **Section 15.10.56** through **Section 15.10.59**.

The SerDes block is designed to accept system parallel data, perform 8b/10b encoding, convert the parallel data to serial data, and, using the selected rate, transmit the data via high speed differential outputs. The block is also designed to receive serial data and, using the selected rate, convert the data to a 10b/8b decoded parallel format and present it to the system interface.

Note: For CPRI, the 8b/10b encoding-decoding is done inside CPRI controller block and does not use SerDes 8b/10b encoders-decoders. The SerDes transparently sends/receives 10b data to the CPRI controller.

15.9.1 Serdes Banks and PLL

The Serdes block is composed of two banks. Each Bank consists of a controlling PLL and SerDes Lanes. The SerDes lanes within a bank are independent from each other and can be run using any combination of protocols provided that all of the protocols within a bank operate using the same base PLL full rate frequency.

It is very important to remember that any and all SerDes lanes connected to a given PLL must have the same base full rate. For example, only 1.25 Gbps, 2.5 Gbps, and 5 Gbps protocols can be run from a PLL running at 5 GHz. The same PLL can not be used simultaneously on two SerDes lanes running 5 Gbps and 3.125 Gbps.

A bank with a PLL VCO frequency of 5 GHz can run any combination of PCI Express/Serial RapidIO lanes at 5 Gbps, PCI Express/Serial RapidIO lanes at 2.5 Gbps, and/or Serial RapidIO/SGMII lanes at 1.25Gbps. A bank with a PLL/VCO Frequency of 6.25 Gbps can only run combinations of SRIO at 3.125 Gbps. CPRI can be used when the PLL VCO frequency is either 4.9152 GHz or 6.144 GHz, but no other protocols can be used with them. The banks are completely independent of each other and have no restrictions on which PLL/VCO frequencies can run on a given bank. There is one restriction, however, when both the PLL banks are used together: both banks cannot have same PLL VCO frequency. For example, if the PLL1 VCO frequency is 5 GHz, the PLL2 VCO frequency can not be 5 GHz; instead, PLL2 must be one of 6.125 GHz, 6.144 GHz, or 4.9152 GHz.

15.9.2 SerDes PLL Reference Clocks

The PLL is used to generate the required top-bit rate frequencies needed by the SerDes lanes.

For each PLL a reference clocks must be supplied. The reference clock frequencies should be such that the PLL VCO frequency is a multiple of the PLL reference clock frequency. Three PLL reference clock frequencies can generate any data-rate supported by the device-

- 100 MHz /5 GHz supports PCI Express/Serial RapidIO lanes at 5Gbps and 2.5Gbps and/or Serial RapidIO/SGMII lanes at 1.25 Gbps.
- 125 MHz/6.125 GHz supports Serial RapidIO lanes at 3.125 Gbps.
- 122.88 MHz/6.144 GHz supports CPRI lanes at 6.144Gbps and 3.072Gbps.
- 122.88 MHz /4.9152 GHz supports CPRI at 4.9152 Gbps, 2.4576 Gbps, and 1.2288Gbps.

15.9.3 Serdes PLL Multiplexing

Each of the 10 Serdes lanes (A–J) can be sourced from either of the two PLLs. Based on the PLL source of each of the banks, there are four PLL multiplexing options

- 10×0 = Lanes A–J sourced by PLL1 and PLL2 is off
- 8×2 = Lanes A–H sourced by PLL1 and Lanes I–J sourced by PLL2
- 6×4 = Lanes A–F sourced by PLL1 and Lanes G–J sourced by PLL2
- 4×6 = Lanes A–D sourced by PLL1 and Lanes E–J sourced by PLL2

The sequence of figures below show the PLL multiplexing options. A–J indicate each of the 10 SerDes ports The BLUE lines indicate PLL1 support and the RED lines indicate PLL2 support. In case of 10×0 configuration, PLL2 is off.

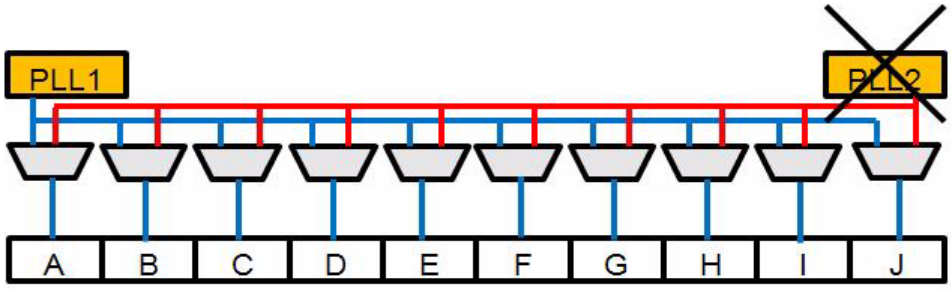


Figure 15-10. 10×0 PLL Multiplexing

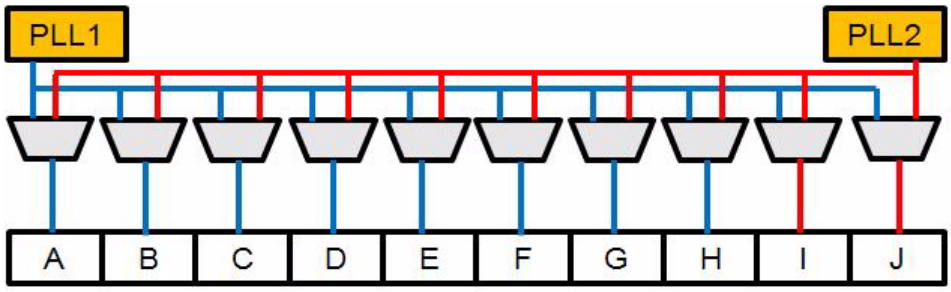


Figure 15-11. 8 × 2 PLL Multiplexing

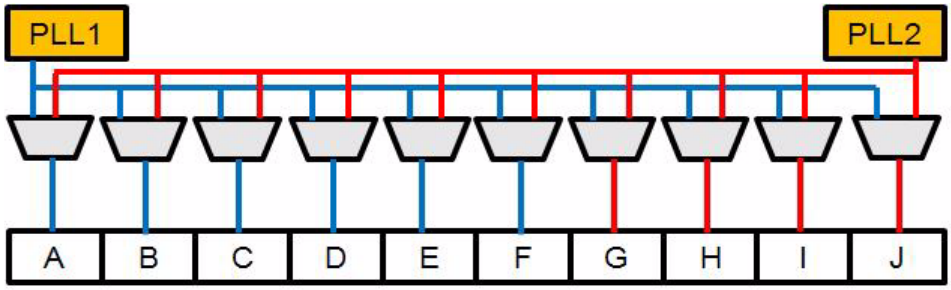


Figure 15-12. 6 × 4 PLL Multiplexing

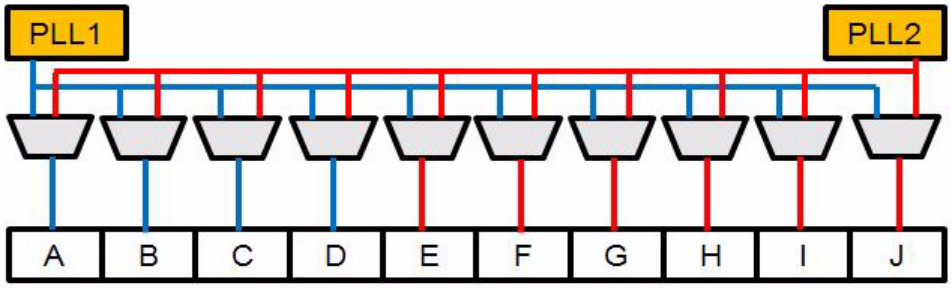


Figure 15-13. 4 × 6 PLL Multiplexing

The CPRI lanes are always sourced from PLL2 and always use some of or all of Lanes E–J. Based on the number of lanes CPRI is using, the following PLL multiplexing options exist:

- CPRI on 2 Lanes (I–J) uses 8 × 2
- CPRI on 4 Lanes (G–J) uses 6 × 4
- CPRI on 6 Lanes (E–J) uses 4 × 6

15.9.4 SerDes Clocks

Each of the 10 lanes have a pair (transmit and receive) of Ten-Bit Interface clocks. **Table 15-6** lists the Transmit/Receive clock frequencies for the different data rates.

Table 15-6. SerDes Clocks to Controllers

Data Rate (Gbps)	TBI Clock (MHz)
1.2500	125.00
2.5000	250.00
3.1250	312.50
5.0000	500.00
6.1440	614.40
4.9152	491.52
3.0720	307.20
2.4576	245.76
1.2288	122.88

15.10 HSSI Programming Model

The HSSI includes configuration, control, and status registers for the following modules:

- CLASS1 Registers use a base address of 0xFFFBA000.
  - CLASS1 Priority Mapping Registers (see **page 15-36**)
  - CLASS1 Priority Auto Upgrade Value Registers (see **page 15-37**)
  - CLASS1 Priority Auto Upgrade Control Registers (see **page 15-38**)
  - CLASS1 Error Address Registers (see **page 15-39**)
  - CLASS1 Error Extended Address Registers (see **page 15-40**)
  - CLASS1 Initiator Profiling Configuration Registers (see **page 15-41**)
  - CLASS1 Initiator Watch Point Control Registers (see **page 15-42**)
  - CLASS1 Arbitration Weight Registers (see **page 15-43**)
  - CLASS1 Start Address Decoder 1 (see **page 15-44**)
  - CLASS1 End Address Decoder 1 (see **page 15-46**)
  - CLASS1 Attributes Decoder 1 (see **page 15-48**)
  - CLASS1 Start Address Decoder 2 (see **page 15-44**)
  - CLASS1 End Address Decoder 2 (see **page 15-46**)
  - CLASS1 Attributes Decoder 2 (see **page 15-48**)
  - CLASS1 IRQ Status Register (see **page 15-51**)
  - CLASS1 IRQ Enable Register (see **page 15-53**)
  - CLASS1 Target Profiling Configuration Register (see **page 15-54**)

- CLASS1 Profiling Control Register (see **page 15-55**)
- CLASS1 Watch Point Control Register (see **page 15-56**)
- CLASS1 Watch Point Access Configuration Register (**page 15-58**)
- CLASS1 Watch Point Extended Access Configuration Register (see **page 15-59**)
- CLASS1 Watch Point Address Mask Register (see **page 15-60**)
- CLASS1 Profiling Time Out Register (see **page 15-61**)
- CLASS1 Target Watch Point Control Register (see **page 15-62**)
- CLASS1 Profiling IRQ Status Register (see **page 15-63**)
- CLASS1 Profiling IRQ Enable Register (see **page 15-64**)
- CLASS1 Profiling Reference Counter Register (see **page 15-65**)
- CLASS1 Profiling General Counter Registers (see **page 15-66**)
- CLASS1 Arbitration Control Register (see **page 15-67**)

- DMA Controller Registers. Dedicated DMA Controller 0 (D0) registers use a base address of 0xFFFA8000. Dedicated DMA Controller 1 (D1) registers use a base address of 0xFFFAA000.
  - Dn DMA 0–3 Mode Registers (DnMR[0–3]), **page 15-68**
  - Dn DMA 0–3 Status Registers (DnSR[0–3]), **page 15-71**
  - Dn DMA 0–3 Current Link Descriptor Extended Address Registers (DnECLNDAR[0–3], **page 15-73**
  - Dn DMA 0–3 Current Link Descriptor Address Registers (DnCLNDAR[0–3]), **page 15-74**
  - Dn DMA 0–3 Source Attributes Registers (DnSATR[0–3]), **page 15-75**
  - Dn DMA 0–3 Source Address Registers (DnSAR[0–3]), **page 15-76**
  - Dn DMA 0–3 Destination Attributes Registers (DnDATR[0–3]), **page 15-77**
  - Dn DMA 0–3 Destination Address Registers (DnDAR[0–3]), **page 15-78**
  - Dn DMA 0–3 Byte Count Registers (DnBCR[0–3], **page 15-79**
  - Dn DMA 1–3 Next Link Descriptor Extended Address Registers (DnENLNDAR[0–3], **page 15-80**
  - Dn DMA 0–3 Next Link Descriptor Address Registers (DnNLNDAR[0–3]), **page 15-81**
  - Dn DMA 1–3 Current List Descriptor Extended Address Registers (DnECLSDAR[0–3], **page 15-82**
  - Dn DMA 0–3 Current List Descriptor Address Registers (DnCLSDAR[0–3]), **page 15-83**
  - Dn DMA 0–3 Next List Descriptor Extended Address Registers (DnENLSDAR[0–3], **page 15-84**
  - Dn DMA 0–3 Next List Descriptor Address Registers (DnNLSDAR[0–3]), **page 15-85**
  - Dn DMA 0–3 Source Stride Registers (DnSSR[0–3]), **page 15-86**
  - Dn DMA 0–3 Destination Stride Registers (DnDSR[0–3]), **page 15-87**
  - Dn DMA General Status Register (DnDGSR), **page 15-88**

  - Dn Local Access Window Base Address Registers 0–9 (DnLAWBAR[0–9]), **page 15-90**
  - Dn Local Access Window Attributes Registers 0–9 (DnLAWAR[0–9]), **page 15-91**
- Protocol Converter Registers use a base address of 0xFFFAD000:
  - CPRI[1–6] Protocol Converter Control Register 0 (PCVTRCPRI[1–6]CR0), **page 15-93**
  - CPRI[1–6] Protocol Converter Control Register 1 (PCVTRCPRI[1–6]CR1), **page 15-94**

- SerDes PHY Interface Registers. SerDes Port uses a base address of 0xFFFAD000.
  - SRDS Bank 1–2 Reset Control Register (SRDSB[1–2]RSTCTL), **page 15-95**
  - SRDS Bank 1–2 PLL Control Register 0 (SRDSB[1–2]PLLCR0), **page 15-97**
  - SRDS Bank 1–2 PLL Control Register 1 (SRDSB[1–2]PLLCR1), **page 15-97**
  - Lane A–J General Control Register 0 (L[A–J]GCR0), **page 15-98**
  - Lane A–J Receive Equalization Control Register 0 (L[A–J]RECR0), **page 15-101**
  - Lane A–J Transmit Equalization Control Register 0 (L[A–J]TECR0), **page 15-104**

There are also several general configuration registers (base address 0xFFF28000) used to control and monitor HSSI operation (see **Chapter 8**, *General Configuration Registers* for details. These include the following:

- General Control Register 2 (GCR2), **page 8-4**
- High Speed Serial Interface Status Register (HSSI_SR), **page 8-8**
- High Speed Serial Interface Control Register 1 (HSSI_CR1), **page 8-12**
- High Speed Serial Interface Control Register 2 (HSSI_CR2), **page 8-15**
- General Control Register 5 (GCR5), **page 8-22**
- General Status Register 2 (GSR2), **page 8-24**
- General Status Register 3 (GSR3), **page 8-27**
- General Interrupt Register 5 (GIR5), **page 8-42**
- General Interrupt Enable Register 5 (GIER5_[0–5]), **page 8-44**
- General Interrupt Register 7 (GIR7), **page 8-74**
- General Interrupt Enable Register 7 (GIER7_[0–5]), **page 8-76**

15.10.1 CLASS1 Priority Mapping Registers (C1PMRx)

C1PMR[0–5] CLASS1 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

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

C1PMRx 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 CLASS1 transactions.

Table 15-7 lists the C1PMRx bit field descriptions.

Table 15-7. C1PMRx 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 15-7. C1PMRx 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

15.10.2 CLASS1 Priority Auto Upgrade Value Registers (C1PAVRx)

C1PAVR[0–5] CLASS1 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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C1PAVRx holds the value loaded to the priority auto-upgrade counter.

Note: You can write to this register while there are open CLASS1 transactions. The AUV field is loaded into the auto-upgrade counter only when you set the AUE bit in C1PACRx. Therefore, always update the AUV field in the C1PAVRx before you set the AUE bit.

Table 15-8 lists the C1PAVRx bit field descriptions.

Table 15-8. C1PAVRx Bit Descriptions

Name	Reset	Description
— 31–16	0	Reserved. Write to 0 for future compatibility.
AUV 15—0	0	**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.

15.10.3 CLASS1 Priority Auto Upgrade Control Registers (C1PACRx)

C1PACR[0–5] CLASS1 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

C1PACRx controls the priority auto-upgrade mechanism.

Note: You can write to this register while there are open CLASS1 transactions.

Table 15-9 lists the C1PACRx bit field descriptions.

Table 15-9. C1PACRx 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.

15.10.4 CLASS1 Error Address Registers (C1EARx)

C1EAR[0–5] CLASS1 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 C1EAR 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 C1ISR, the internal transaction address is stored and the C1EARx is locked and does not update even if another error with a different transactions address occurs. Only when the AEIx bit in the C1ISR is cleared (either by a hardware reset or by writing a 1 to it) is C1EARx unlocked.

Table 15-10 lists the C1EARx bit field descriptions.

Table 15-10. C1EARx 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:

— C1EAR0 = Address generated by OCN-to-MBus port 0 (O2M0).
— C1EAR1 = Address generated by CPRI MBus port 1 — CPRI read
— C1EAR2 = Address generated by AXI-to-MBus port 1 (AXI2M1)—eMSG read.
— C1EAR3 = Address generated by OCN-to-MBus port 1 (O2M1)
— C1EAR4 = Address generated by CPRI MBus port 0 — CPRI write
— C1EAR5 = Address generated by AXI-to-MBus port 0 (AXI2M0)—eMSG write.

15.10.5 CLASS1 Error Extended Address Registers (C1EEARx)

C1EEAR[0-5] CLASS1 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 C1EEAR 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 C1ISR, the internal transaction address is stored and the C1EEAR 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 C1EEAR unlocked.

Table 15-11 lists the C1EEARx bit field descriptions.

Table 15-11. C1EEARx 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.	01100 OCN-to-MBus Port (O2M0) 10001 CPRI MBus Port 1—CPRI read 10011 AXI-to-MBus port 1 (AXI2M1)—eMSG read 01101 OCN-to-MBus Port 1 (O2M1) 10010 CPRI MBus Port 0 — CPRI write 10100 AXI-to-MBus port 0 (AXI2M0)—eMSG write
ERR_ADD 3–0	0	**Error Address** This field stores the 4 msbs of the address of the internal transaction that caused the error.	

15.10.6 CLASS1 Initiator Profiling Configuration Registers (C1IPCRx)

C1IPCR[0–5] CLASS1 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

C1IPCRx controls the CLASS1 initiator profiling measurements. Each initiator has a dedicated C1IPCR which is numbered according to the initiator number within each CLASS1 module. The CLASS1 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 C1IPCR and make sure all the other C1IPCR and the C1TPCR for that CLASS1 are cleared.

Note: Only one PMM field among all C1IPCRx and C1TPCR can be > 0 during profiling.

Table 15-12 lists the C1IPCRx bit field descriptions.

Table 15-12. C1IPCRx 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. 10010–11111 reserved

Table 15-13. Initiator Numbers

Initiator Number	Initiator Module
0	OCN-to-MBus Port 0 (O2M0)
1	CPRI MBus Port 1— CPRI read
2	AXI-to-MBus port 1 (AXI2M1)—eMSG read
3	OCN-to-MBus Port 1 (O2M1)

Table 15-13. Initiator Numbers

Initiator Number	Initiator Module
4	CPRI MBus Port 0 — CPRI write
5	AXI-to-MBus port 0 (AXI2M0)—eMSG write

15.10.7 CLASS1 Initiator Watch Point Control Registers (C1IWPCRx)

C1IWPCR[0–5] CLASS1 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

C1IWPCRx controls the Watch Point Unit operation for the associated initiator. The Watch Point Unit monitors a specific access defined by the C1WPCRx, C1WPACRx, C1WPEACRx, and C1WPAMR. Each initiator can be enabled/disabled to monitor the selected access. You can write to this register while there are open CLASS1 transactions.

Note: Only one WPEN field can be set among all C1IWPCRx and C1TWPCRx when snooping watch point events.

Table 15-14 lists the C1IWPCRx bit field descriptions.

Table 15-14. C1IWPCRx 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.

15.10.8 CLASS1 Arbitration Weight Registers (C1AWRx)

C1AWR[0–5] CLASS1 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
Boot	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
	—												WEIGHT			
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The value in C1AWRx 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 CLASS1 Arbiter chooses the higher priority request instead of the weighted winner.

Table 15-15 lists the C1AWRx bit field descriptions.

Table 15-15. C1AWRx 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 this value.

Using the reset values can result in poor initial system performance. **Table 15-15** 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. Also, see **Table 15-37** for recommended initial settings for the CLASS1 Arbitration Control Register (C1ACR).

15.10.9 CLASS1 Start Address Decoder 1 (C1SAD1)

C1SAD1 CLASS1 Start Address Decoder 1 Offset 0xC04

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																
SAD1	0	0	0	0	0	0	0	0	0	0	0	0	0	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																
SAD1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C1SAD1 controls the functionality of the decoder for CLASS1 toward the CLASS0 initiator port 8. It also controls the decoder toward initiator port 12 in split read/write mode (when C1ATD1[SPRW] is set (1). It contains 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 C1ATD1[DEN] bit before changing the contents of C1SAD1.

This register is reset by a hardware reset only. **Table 15-16** lists the C1SAD1 bit field descriptions.

Table 15-16. C1SAD1 Bit Descriptions

Name	Reset	Description
— 31–24	0	Reserved. Write to 0 for future compatibility.
SA[35–12] 23–0	0x000000	**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 this register when there are open transactions being handled by the CLASS to the specified target controlled by the register.

15.10.10 CLASS1 Start Address Decoder 2(C1SAD2)

C1SAD2 CLASS1 Start Address Decoder 2 Offset 0xC08

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																
SAD2	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
	SA27	SA26	SA25	SA24	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16	SA15	SA14	SA13	SA12
Type	R/W															
Reset																
SAD2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C1SAD2 configures the address decoding of CLASS1 toward the CLASS0 initiator port 12. It contains 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 C1ATD2[DEN] bit before changing the contents of C1SAD2.

Note: C1SAD2 is valid only if C1ATD1[SPRW] is clear (0).

This register is reset by a hardware reset only. **Table 15-17** lists the C1SAD2 bit field descriptions.

Table 15-17. C1SAD2 Bit Descriptions

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

Never write to this register when there are open transactions being handled by the CLASS to the specified target controlled by the register.

15.10.11 CLASS1 End Address Decoder 1 (C1EAD1)

C1EAD1 CLASS1 End Address Decoder 1 Offset 0xC44

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																
EAD1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	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																
EAD1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

C1EAD1 configures the address decoding of CLASS1 toward the CLASS0 initiator port 8. It also controls the decoder toward initiator port 12 in split read/write mode (when C1ATD1[SPRW] is set (1). It contains 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 C1ATD1[DEN] bit before changing the contents of C1EAD1.

This register is reset by a hardware reset only. **Table 15-18** lists the C1EAD1 bit field descriptions.

Table 15-18. C1EAD1 Bit Descriptions

Name	Reset	Description
— 31–24	0	Reserved. Write to 0 for future compatibility.
EA[35–12] 23–0	0x0ffffe	**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 this register when there are open transactions being handled by the CLASS1 to the specified target controlled by the register.

15.10.12 CLASS1 End Address Decoder 1 (C1EAD2)

C1EAD2 CLASS1 End Address Decoder 2 Offset 0xC48

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																
EAD2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	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																
EAD2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C1EAD2 configures the address decoding of CLASS1 toward the CLASS0 initiator port 12. It contains 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 C1ATD2[DEN] bit before changing the contents of C1EAD2.

Note: C1EAD2 is valid only if C1ATD1[SPRW] is clear (0).

This register is reset by a hardware reset only. **Table 15-19** lists the C1EAD2 bit field descriptions.

Table 15-19. C1EAD2 Bit Descriptions

Name	Reset	Description
— 31–24	0	Reserved. Write to 0 for future compatibility.
EA[35–12] 23–0	0x07ffff	**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.

Never write to this register when there are open transactions being handled by the CLASS1 to the specified target controlled by the register.

15.10.13 CLASS1 Attributes Decoder 1 (C1ATD1)

C1ATD1 CLASS0 Attributes Decoder 1 Offset 0xC84

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
	—											SPRW	—			DEN
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

C1ATD1 controls the functionality of the decoder for CLASS1 toward the CLASS0 initiator port 8. The register contains the bit that enables/disables the specific port. In addition to containing the bit that enables/disables the port 1, this register also contains a bit to determine whether the access between CLASS1 and CLASS0 uses direct connections or uses split read/write access. If a decoder is disabled, it never indicates a hit, even if the address corresponds to the decoder address window. In this case, the CLASS1 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 C1SAD1 and C1EAD1.

This register is reset by a hardware reset only. **Table 15-20** lists the C1ATD1 bit field descriptions.

Table 15-20. C1ATD1 Bit Descriptions

Name	Reset	Description	Settings
— 31–5	0	Reserved. Write to 0 for future compatibility.	
SPRW 4	1	**Split Read/Write** Determines whether reads and writes both occur on a port or they are split between two ports. **Note:** **Table 15-21** summarizes the split access configuration options.	0 Read and write accesses use both target ports 0 and 1. Address decoding for each port is done by the respective register sets: C1SAD1/C1EAD1 are used for port 1 and C1SAD2/C1EAD2 are used for port 0. Use of this mode may improve performance for applications with an unbalanced proportion of HSSI reads and writes, or it can be used to allocate memory accesses to reduce collisions between the HSSI initiators. In this mode, you must configure the start/end registers so that all desired memory space (including M2 banks, M3, DDR, and MAPLE slave ports) is split between the two ports to optimize the specific application access needs. 1 Reads and writes go through different ports (Reads use target 1 and writes use target 0). The start and end addresses for both ports are controlled by C1SAD1 and C1EAD1.
— 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 this register when there are open transactions being handled by the CLASS1 to the specified target controlled by the register.

Table 15-21. Split HSSI Access Configurations

Scenario	C1ATD1[DEN]	C1ATD2[DEN]	C1ATD1[SPRW]	Result
1	0	0	x	No accesses can be made by the HSSI
2	0	1	x	All HSSI accesses use Target0 as defined by C1SAD2/C1EAD2. (no benefit)
3	1	0	0	All HSSI accesses use Target1 as defined by C1SAD1/C1EAD1. (no benefit)
4	1	x	1	Writes from the HSSI use Target 1. Reads to HSSI use Target 0. Both ports are defined by C1SAD1/C1EAD1.
5	1	1	0	Some accesses use Target 1 (defined by C1SAD1/C1EAD1). Some accesses use Target 0 (defined by C0SAD2/C0EAD2). Use for cases of unbalanced reads/writes or to split address space among different HSSI masters. **Note:** A hit in C1SAD2/C1EAD2 has a higher priority than a hit in C1SAD1/C1EAD1 window.

15.10.14 CLASS1 Attributes Decoder 1 (C1ATD2)

C1ATD2 CLASS0 Attributes Decoders Offset 0xC88

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	0

C1ATD2 controls the functionality of the decoder for CLASS1 toward the CLASS0 initiator port 12. The register contains 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 CLASS1 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 C1SAD2 and C1EAD2.

Note: C1ATD2 is valid only if C1ATD1[SPRW] is clear (0).

This register is reset by a hardware reset only. **Table 15-22** lists the C1ATD2 bit field descriptions.

Table 15-22. C1ATD2 Bit Descriptions

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

Never write to this register when there are open transactions being handled by the CLASS1 to the specified target controlled by the register.

15.10.15 CLASS1 IRQ Status Register (C1ISR)

C1ISR CLASS1 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
	—										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 C1ISR 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 15-23 lists the C1ISR bit field descriptions.

Table 15-23. C1ISR Bit Descriptions

Name	Reset	Description	Settings
— 31–6	0	Reserved. Write to 0 for future compatibility.	
AEI5 5	0	**Address Error Interrupt 5** 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.
AEI4 4	0	**Address Error Interrupt 4** 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.
AEI3 3	0	**Address Error Interrupt 3** 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.
AEI2 2	0	**Address Error Interrupt 2** 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.

Table 15-23. C1ISR Bit Descriptions (Continued)

Name	Reset	Description	Settings
AEI1 1	0	**Address Error Interrupt 1** 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.
AEI0 0	0	**Address Error Interrupt 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.

15.10.16 CLASS1 IRQ Enable Register (C1IER)

C1IER CLASS1 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
	—										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 C1IER is used to enable/disable the generation of interrupts that have occurred. There is a dedicated bit for each initiator. If a C1IER bit is cleared the corresponding bit in the C1ISR is masked. This register is reset by the hardware reset only.

Table 15-24 lists the C1IER bit field descriptions.

Table 15-24. C1IER Bit Descriptions

Name	Reset	Description	Settings
— 31–6	0	Reserved. Write to 0 for future compatibility.	
AEIE5 5	0	**Address Error5 Interrupt Enable** Used to enable/disable the address error interrupt for an initiator.	0 Interrupt masked. 1 Interrupt enabled.
AEI4 4	0	**Address Error4 Interrupt Enable** Used to enable/disable the address error interrupt for an initiator.	0 Interrupt masked. 1 Interrupt enabled.
AEIE3 3	0	**Address Error3 Interrupt Enable** Used to enable/disable the address error interrupt for an initiator.	0 Interrupt masked. 1 Interrupt enabled.
AEIE2 2	0	**Address Error2 Interrupt Enable** Used to enable/disable the address error interrupt for an initiator.	0 Interrupt masked. 1 Interrupt enabled.
AEI1 1	0	**Address Error1 Interrupt Enable** Used to enable/disable the address error interrupt for an initiator.	0 Interrupt masked. 1 Interrupt enabled.
AEI0 0	0	**Address Error0 Interrupt Enable** Used to enable/disable the address error interrupt for an initiator.	0 Interrupt masked. 1 Interrupt enabled.

15.10.17 CLASS1 Target Profiling Configuration Register (C1TPCR)

C1TPCR CLASS1 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

C1TPCR is used to control the CLASS1 target profiling measurements. Each CLASS1 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 C1IPCRx are cleared.

Note: For each CLASS1 module, you can only monitor one transaction. Therefore, only one PMM field in C1IPCRx and C1TPCR can be greater than 0 during profiling.

Table 15-25 lists the C1TPCR bit field descriptions.

Table 15-25. C1TPCR 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.
TN 11–8	0	**Target Number** Indicates the number of selected target.	0010 HSSI Port 0 (CLASS Init. 8) 0001 HSSI Port 1 (CLASS Init. 12) All other values reserved.
— 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.

15.10.18 CLASS1 Profiling Control Register (C1PCR)

C1PCR CLASS1 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

C1PCR controls the CLASS1 profiling operation. The register is reset only by a hardware reset. **Table 15-26** lists the C1PCR bit field descriptions.

Table 15-26. C1PCR 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.

15.10.19 CLASS1 Watch Point Control Register (C1WPCR)

C1WPCR CLASS1 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	ATE	SIE	PRE	BCE	ATRE	ATAE	RSE	SNE	OPE	TSTE	SPVE	RWE	AE	CE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C1WPCR controls the CLASS1 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 15-27** lists the C1WPCR bit field descriptions.

Table 15-27. C1WPCR 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 Upgrades type compare with C1WPEACR 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 C1WPEACR enabled.
EATE 14	0	**EOT Attributes Compare Enable** Enables/disables the EOT attributes comparison.	0 EOT attributes compare disabled. 1 EOT attributes compare with C1WPEACR enabled.
ATE 13	0	**Attributes Compare Enable** Enables/disables the attributes comparison.	0 Attributes compare disabled. 1 Attributes compare with C1WPEACR enabled.
SIE 12	0	**Source ID Compare Enable** Enables/disables the source ID comparison.	0 Source ID compare disabled. 1 Source ID compare with C1WPEACR enabled.
PRE 11	0	**Priority Level Compare Enable** Enables/disables the priority level comparison.	0 Priority level compare disabled. 1 Priority level compare with C1WPEACR 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 C1WPEACR 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 C1WPACR 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 C1WPACR enabled.
RSE 7	0	**Read-Safe Access Compare Enable** Enables/disables the read-safe type comparison.	0 Read-safe type compare disabled. 1 Read-safe type compare with C1WPACR enabled.

Table 15-27. C1WPCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
SNE 6	0	**Snoop Compare Enable** Enables/disables the snoop type comparison.	0 Snoop type compare disabled. 1 Snoop type compare with C1WPACR enabled.
OPE 5	0	**Optimize Compare Enable** Enables/disables the optimize type comparison.	0 Optimize type compare disabled. 1 Optimize type compare with C1WPACR enabled.
TSTE 4	0	**Test Access Compare Enable** Enables/disables the test type comparison.	0 Test type compare disabled. 1 Test type compare with C1WRACR enabled.
SPVE 3	0	**Supervisor Access Compare Enable** Enables/disables supervisor access comparison.	0 Supervisor type compare disabled. 1 Supervisor type compare with C1WRACR enabled.
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 C1WPACR enabled.
AE 1	0	**Address Compare Enable** Enables/disables comparison of the access address.	0 Address compare disabled. 1 Address compare with C1WPACR 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.

15.10.20 CLASS1 Watch Point Access Configuration Register (C1WPACR)

C1WPACR CLASS1 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	RS	SN	OP	TST	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

C1WPACR, along with C1WPEACR, configures the selected access to monitor. The watch point monitoring occurs only if the respective function is enabled in the C1WPCR. The register is reset only by a hardware reset. **Table 15-28** lists the C1WPACR bit field descriptions.

Table 15-28. C1WPACR 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.
RS 29	0	**Read-Safe Access** Defines the read-safe access type to monitor.	0 Non-read-safe access. 1 Read-safe access.
SN 28	0	**Snoop Access** Defines the snoop access type to monitor.	0 Non-snoop access. 1 Snoop access.
OP 27	0	**Optimize Access** Defines the optimize access type to monitor.	0 Non-optimized access. 1 Optimized access.
TST 26	0	**Test Access** Defines the test access type to monitor.	0 Non-test access. 1 Test access.
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 C1WPAWMR, defines the start and range of the addresses the watch point unit monitors. **Note:** For every bit in C1WPAMR[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.	

15.10.21 CLASS1 Watch Point Extended Access Configuration Register (C1WPEACR)

C1WPEACR CLASS1 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

C1WPEACR, along with C1WPACR, configures the selected access to monitor. The watch point monitoring occurs only if the respective function is enabled in the C1WPCR. The register is reset only by a hardware reset. **Table 15-29** lists the C1WPEACR bit field descriptions.

Table 15-29. C1WPEACR 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.	0x9 Target port 1 CLASS initiator port 8 0xA Target port 2 CLASS initiator port 12 All other values reserved.
— 23–16	0	Reserved. Write to 0 for future compatibility.	
SI 15–11	0	**Source** Defines the source ID to monitor.	0x00 O2M0 (OCN to MBus port 0) 0x01 CPRI MBus read port 0x02 AXI2M1 (eMSG read) 0x03 O2M1 (OCN to MBus port 1) 0x04 CPRI Mbus write port 0x05 AXI2M1 (eMSG write) All others reserved.
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.

15.10.22 CLASS1 Watch Point Address Mask Registers (C1WPAMR)

C1WPAMR CLASS1 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

C1WPAMR controls the address range monitored by the watch point unit. The register is reset only by a hardware reset. **Table 15-30** lists the C1WPAMR bit field descriptions.

Table 15-30. C1WPAMR 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 C1WPACR[ADDR]. **Note:** For every bit in ADDM that is cleared, make sure the corresponding bit is cleared in the C1WPACR.	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.

15.10.23 CLASS1 Profiling Time-Out Register (C1PTOR)

C1PTOR CLASS1 Profiling Time-Out Register 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

C1PTOR is used to stop the profiling unit operation. When the C1PRCR reaches the value stored in C1PTOR and C1PCR[TOE] is set, the CLASS1 clears the C1PCR[PE] bit to disable the profiling unit. When C1PCR[PE] clears, the CLASS1 stops all profiling counters. The register is reset only by a hardware reset. **Table 15-31** lists the C1PTOR bit field descriptions.

Table 15-31. C1PTOR 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.

15.10.24 CLASS1 Target Watch Point Control Register (C1TWPCR)

C1TWPCR CLASS1 Target Watch Point Control Register 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
	—													WPEN2	WPEN1	—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C1TWPCR controls the watch point unit operation for CLASS1 targets. The watch point unit monitors a specific access defined in C1WPCR, C1WPACR, C1WPEACR, and C1WPAMR. 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 C1IWPCRx and C1TWPCR. That is, only one watch point unit can be active at a time.

Table 15-23 lists the C1TWPCR bit field descriptions.

Table 15-32. C1TWPCR Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to 0 for future compatibility.	
WPEN[2–1] 2–1	0	**Watch Point Enable 2–1** 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.
— 0	0	Reserved. Write to 0 for future compatibility.	

15.10.25 CLASS1 Profiling IRQ Status Register (C1PISR)

C1PISR CLASS1 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

C1PISR indicates that a watch point event occurred or that the C1PRCR overflowed. An interrupt is generated if the status bit is set and the corresponding bit in C1PIERx 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 15-33** lists the C1PISR bit field descriptions.

Table 15-33. C1PISR 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 C1PRCR overflowed (reached 0xFFFFFFFF) during the last measurement.

15.10.26 CLASS1 Profiling IRQ Enable Register (C1PIER)

C1PIER CLASS1 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

C1PIER 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 15-34** lists the C1PIER bit field descriptions.

Table 15-34. C1PIER 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.

15.10.27 CLASS1 Profiling Reference Counter Register (C1PRCR)

C1PRCR CLASS1 Profiling Reference Counter Registers Offset 0xE40

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

C1PRCR 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 C1PRCR stops when the profiling unit is disabled, or when the C1PRCR reaches the value stored in C1PTOR and TOE is set, which causes the CLASS1 to clear the PE bit to disable the profiling unit. When PE clears, the CLASS1 stops all profiling counters. The register is reset only by a hardware reset only. **Table 15-35** lists the C1PRCR bit field descriptions.

Table 15-35. C1PRCR Bit Descriptions

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

15.10.28 CLASS1 Profiling General Counter Registers (C1PGCRx)

C1PGCR[0–3] CLASS1 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

C1PGCRx 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 C1PRCR stops when the profiling unit is disabled, or when the C1PRCR reaches the value stored in C1PTOR and TOE is set, which causes the CLASS1 to clear the PE bit to disable the profiling unit. When PE clears, the CLASS1 stops all profiling counters. The register is reset only by a hardware reset. **Table 15-36** lists the C1PGCR bit field descriptions.

Table 15-36. C1PGCR Bit Descriptions

Name	Reset	Description
CNT 31–0	0	**Counter** Holds the counter value of the selected measurement. **Table 15-1** lists the measurements counted by each counter for each configuration combination.

15.10.29 CLASS1 Arbitration Control Register (C1ACR)

C1ACR CLASS1 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
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
	—													LA2	LA1	—
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	1

The C1ACR controls the CLASS1 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 about the winner according to the MDBW parameter and the byte count of the winner access. When Late Arbitration mode is disabled, the arbiter makes a decision every clock cycle.The register is reset by a hard reset only. **Table 15-37** lists the C1ACR bit field descriptions.

Table 15-37. C1ACR 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.
— 27–3	0	Reserved. Write to 0 for future compatibility.	
LA[2–1] 2–1	0	**Late Arbitration 2–1** Enables/disables late arbitration mode for the associated arbiter. **Note:** As with the arbitration weight (see **Section 15.10.8**, *CLASS1 Arbitration Weight Registers (C1AWRx)*, on page 15-43), the default value for this field does not yield optimal performance. Freescale recommends that after reset, write a value of 0b11 to this field. This value assigns late arbitration to system CLASS accesses. 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.
— 0	0	Reserved. Write to 0 for future compatibility.	

15.10.30 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	1	0	0	0	0	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 15-38** describes the fields of the DnMR.

Table 15-38. DnMR Field Descriptions

Bits	Reset	Description	Setting
— 31–28	0	Reserved. Write to zero for future compatibility.	
BWC 27–24	1000	**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 15-38. 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 15-38. 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.

15.10.31 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 15-39** describes the fields of the DnSR.

Table 15-39. 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 15.5.6**, *DMA Errors,* on page 15-25 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 15-39. 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.

15.10.32 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 15-40** describes the DnECLNDAR fields.

Table 15-40. DnECLNDAR Field Descriptions

Bits	Reset	Description	Setting
— 31–4	0	Reserved. Write to zero for future compatibility.	

Table 15-40. 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.	

15.10.33 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 15-41** describes the DnCLNDAR fields.

Table 15-41. 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 15-41. 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.	

15.10.34 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 15-42 describes the fields of the DnSATR.

Table 15-42. 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, do not snoop local processor. All other values reserved.

Table 15-42. 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.	

15.10.35 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 15-43** describes the DnSAR fields.

Table 15-43. 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.

15.10.36 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 15-44 describes the fields of the DATR.

Table 15-44. 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 15-44. 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.	

15.10.37 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 15-45 describes the DnDAR fields.

Table 15-45. 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.

15.10.38 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 15-46** describes the fields of the DnBCR.

Table 15-46. 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.

15.10.39 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 15-47** describes the ENLNDAR fields.

Table 15-47. 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.	

15.10.40 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 15-48** describes the DnNLNDAR fields.

Table 15-48. 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 15-48. 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.

15.10.41 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 15-49** describes the DnECLSDAR fields.

Table 15-49. 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.	

15.10.42 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 15-50** describes the DnCLSDAR fields.

Table 15-50. 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.	

15.10.43 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 15-51 describes the DnENLSDAR fields.

Table 15-51. 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.	

15.10.44 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 15-52** describes the DnNLSDAR fields.

Table 15-52. 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.

15.10.45 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 15-52** describes the DnSSR fields.

Table 15-53. 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.

15.10.46 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 15-54** describes the DnDSR fields.

Table 15-54. 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.

15.10.47 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 15-55** describes the fields of the DnDGSR.

Table 15-55. 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 15-55. 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.

15.10.48 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 15-56** describes the DnLAWBARx fields.

Table 15-56. 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.	

15.10.49 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								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 15-57** describes the DnLAWARx fields.

Table 15-57. 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 15-57. 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

Note: When using the same OCN-to-MBus (O2M) for Read and Write transactions, the write transactions may write incorrect data for specific access sequences. To preclude this scenario, use one bridge for Write transactions and the other bridge for read transactions.

15.10.50 CPRI*n* PCVTR Control Register 0(PCVTRCPRI*n*CR0)

PCVTRCPRI1CR0 CPRI 1–6 PCVTR Control Register 0 Offset 0x150
PCVTRCPRI2CR0 Offset 0x158
PCVTRCPRI3CR0 Offset 0x160
PCVTRCPRI4CR0 Offset 0x168
PCVTRCPRI5CR0 Offset 0x170
PCVTRCPRI6CR0 Offset 0x178

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RST_CPRI	PD_CPRI	—													
Type	RW		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
	RX_DLY															
Type	RW															
Reset	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x

PCVTRCPRI*n*CR0 contains the control bits used for the CPRI1, CPRI2, CPRI3, CPRI4, CPRI5, and CPRI6 Protocol Converter logic.

Table 15-58. PCVTRCPRI[1–6]CR0 Field Descriptions

Bits	Reset	Description	Settings
RST_CPRI 31	0	**CPRI Reset** When set, it is in application mode.	0 Reset 1 Application mode.
PD_CPRI 30	0	**CPRI Power Down** When set, the protocol converter is powered down.	0 Application mode. 1 PCVTR power down.
— 29–16	0	Reserved. Write to zero for future compatibility.	
RX_DLY 15–0	0	**Receive Delay** Indicates the receive delay in 8 ps units. The value in the register comes from the reset configuration word when coming out of Reset. Every time that software preforms a SerDes reset request, the register values are overwritten based on the recommended settings for each link rate.	0x09A6 1.2288 Gbps 0x052D 2.4576 Gbps 0x042F 3.0720 Gbps 0x02F1 4.9152 Gbps 0x0265 6.1440 Gbps

15.10.51 CPRI*n* PCVTR Control Register 1(PCVTRCPRI*n*CR1)

PCVTRCPRI1CR1 CPRI 1–6 PCVTR Control Register 1 Offset 0x154
PCVTRCPRI1CR1 Offset 0x15C
PCVTRCPRI1CR1 Offset 0x164
PCVTRCPRI1CR1 Offset 0x16C
PCVTRCPRI1CR1 Offset 0x174
PCVTRCPRI1CR1 Offset 0x17C

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**
	TX_DLY															
Type	R/W															
Reset	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x

PCVTRCPRI*n*CR1 contains the control bits used for the CPRI1, CPRI2, CPRI3, CPRI4, CPRI5, and CPRI6 Protocol Converter logic.

Table 15-59. PCVTRCPRI[1–6]CR1 Field Descriptions

Bits	Reset	Description	Settings
— 31–16	0	Reserved. Write to zero for future compatibility.	
TX_DLY 15–0	0	**Transmit Delay** Indicates the transmit delay in 8 ps units. The value in the register comes from the reset configuration word when coming out of Reset. Every time that software preforms a SerDes reset request, the register values are overwritten based on the recommended settings for each link rate.	0x0E20 1.2288 Gbps 0x072B 2.4576 Gbps 0x05C7 3.0720 Gbps 0x03B1 4.9152 Gbps 0x02FF 6.1440 Gbps

15.10.52 SRDS Bank 1 Reset Control Register (SRDSB1RSTCTL)

SRDB1RSTCTL SRDS Bank 1 Reset Control Register Offset 0x0000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RST REQ	—														
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

SRDSB1RSTCTL contains the control/status bits for the SerDes PHY and protocol converter for port 0 and SerDes PLL1.

Note: Always write to reserved bits with the value they return when read. That is, when programming the register, read the value, modify appropriate fields, and write the modified value back to the register.

Note: The SerDes Port uses a base address of 0xFFFAD000.

Table 15-60 describes the SRDSB1RSTCTL fields.

Table 15-60. SRDSB1RSTCTL Field Descriptions

Bits	Description	Settings
RSTREQ 31	**SerDes Reset Request** Resets the SerDes PLL1 lock detection and test circuitry and also resets all logic in all lanes associated with SerDes PLL1. To initiate a SerDes reset, software writes a 1. The reset state machine clears the bit before reset is completed. **Note:** Software can only set this bit but not clear it. If the bit cleared before reset is complete, the reset state machine ignores the change.	0 No reset requested. 1 SerDes reset requested.
— 30–0	Reserved. Write to zero for future compatibility.	

15.10.53 SRDS Bank 2 Reset Control Register (SRDSB2RSTCTL)

SRDB2RSTCTL SRDS Bank 2 Reset Control Register Offset 0x0020

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

SRDSB2RSTCTL contain the control/status bits for the SerDes PHY and protocol converter for port 1 and SerDes PLL2.

Note: Always write to reserved bits with the value they return when read. That is, when programming the register, read the value, modify appropriate fields, and write the modified value back to the register.

Note: The SerDes Port uses a base address of 0xFFFAD000.

Table 15-60 describes the SRDSB2RSTCTL fields.

Table 15-61. SRDSB2RSTCTL Field Descriptions

Bits	Description	Settings
RSTREQ 31	**SerDes Reset Request** Resets the PLL lock detection and test circuitry and also resets all logic in all lanes associated with a PLL. To initiate a SerDes reset, software writes a 1. The reset state machine clears the bit before reset is completed. **Note:** Software can only set this bit but not clear it. If the bit cleared before reset is complete, the reset state machine ignores the change.	0 No reset requested. 1 SerDes reset requested.
— 30–28	Reserved. Write to zero for future compatibility.	
RFCK_EN 27	**Reference Clock Enable** Used to enable the SerDes reference clock.	0 Disable the SerDes Reference Clock 1 (SerDes PLL2). 1 Enable a buffered version of the SerDes Reference Clock 1 (SerDes PLL2).
— 26–8	Reserved. Write to zero for future compatibility.	
PLLRST 7	**PLL Master Reset** Invokes an internal signal that resets the PLL lock detection and test circuitry. It is used in CPRI auto rate negotiation.	0 Application mode. 1 PLL Reset.
— 6–0	Reserved. Write to zero for future compatibility.	

15.10.54 SRDS Bank 1–2 PLL Control Register 0 (SRDSB[1–2]PLLCR0)

SRDB1PLLCR0 SRDS Bank 1 PLL Control Register 0 Offset 0x0004
SRDB2PLLCR0 SRDS Bank 2 PLL Control Register 0 Offset 0x0024

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

SRDSB[1–2]PLLCR0 contain the control/status bits for the SerDes interface.

Note: The SerDes Port uses a base address of 0xFFFAD000.

Table 15-60 describes the SRDSB[1–2]PLLCR0 fields.

Table 15-62. SRDSB[1–2]PLLCR0 Field Descriptions

Bits	Description	Settings
— 31–19	Reserved. Write to zero for future compatibility.	
FRATE_SEL 18–16	**Full Rate Select** Selects the PLL VCO frequency. All lanes within a bank must operate at a multiple of this frequency and within the specified protocol limits.	000 5 GHz. 001 6.25 GHz 010 Reserved. 011 4.915 GHz 100 6.144 GHz (valid on SRDSB2PLLCR0 only) All other values reserved.
— 15–0	Reserved. Write to zero for future compatibility.	

15.10.55 SRDS Bank 1–2 PLL Control Register 1 (SRDSB[1–2]PLLCR1)

SRDB1PLLCR1 SRDS Bank 1 PLL Control Register 1 Offset 0x0008
SRDB2PLLCR1 SRDS Bank 2 PLL Control Register 1 Offset 0x0028

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

SRDSB[1–2]PLLCR0 contain the control/status bits for the SerDes interface.

Note: The SerDes Port uses a base address of 0xFFFAD000.

Table 15-60 describes the SRDSB[1–2]PLLCR0 fields.

Table 15-63. SRDSB[1–2]PLLCR1 Field Descriptions

Bits	Description	Settings
— 31–28	Reserved. Write to zero for future compatibility.	
PLLBW_SEL 27	**PLL Bandwidth Select** Selects PLL bandwidth.	0 Nominal PLL bandwidth (recommended for non-PCI Express configurations) 1 PLL bandwidth for PCI Express (recommended for PCI Express configurations).
— 26–10	Reserved. Write to zero for future compatibility.	
RFCK_PTRM_VCM 9–8	**Selects Rx Termination Common Mode** Rx termination common mode.	00 If the SerDes controller or the lane power down is asserted, common mode is high impedance. Otherwise, common mode is the calibrated termination to xcorevss. 01 Common mode is always calibrated termination to xcorevss (recommended) 10 Common mode is high impedance, Rx termination is uncalibrated 120 Ω differential. 11 Common mode is 0.7*xcorevdd through 3 KΩ, Rx termination is uncalibrated 120 Ω differential.
— 7–0	Reserved. Write to zero for future compatibility.	

15.10.56 Lane A–J General Control Register 0 (L[A–J]GCR0)

LAGCR0	Lane A General Control Register 0	Offset 0x0200
LBGCR0	Lane B General Control Register 0	Offset 0x0240
LCGCR0	Lane C General Control Register 0	Offset 0x0280
LDGCR0	Lane D General Control Register 0	Offset 0x02C0
LEGCR0	Lane E General Control Register 0	Offset 0x0300
LFGCR0	Lane F General Control Register 0	Offset 0x0340
LGGCR0	Lane G General Control Register 0	Offset 0x0380
LHGCR0	Lane H General Control Register 0	Offset 0x03C0
LIGCR0	Lane I General Control Register 0	Offset 0x0400
LJGCR0	Lane J General Control Register 0	Offset 0x0440

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

L[A–J]GCR0 contains functional control bits for the SerDes logic on lanes A to J.

Note: The SerDes Port uses a base address of 0xFFFAD000.

Table 15-64 describes the L[A–J]GCR0 fields.

Table 15-64. L[A–J]GCR0 Field Descriptions

Bits	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
RRAT_SEL 29–28	Configured by Reset	**Receiver Speed Selection** Selects the lane receiver speed	00 Full speed 01 Half speed 10 Quarter speed 11 reserved
— 27–26	0	Reserved. Write to zero for future compatibility.	
TRAT_SEL 25–24	Configured by Reset	**Transmitter Speed Selection** Selects the lane receiver speed	00 Full speed 01 Half speed 10 Quarter speed 11 reserved
— 23–0	0	Reserved. Write to zero for future compatibility.	

15.10.57 Lane A–J General Control Register 1 (L[A–J]GCR1)

LAGCR1	Lane A General Control Register 1	Offset 0x0204
LBGCR1	Lane B General Control Register 1	Offset 0x0244
LCGCR1	Lane C General Control Register 1	Offset 0x0284
LDGCR1	Lane D General Control Register 1	Offset 0x02C4
LEGCR1	Lane E General Control Register 1	Offset 0x0304
LFGCR1	Lane F General Control Register 1	Offset 0x0344
LGGCR1	Lane G General Control Register 1	Offset 0x0384
LHGCR1	Lane H General Control Register 1	Offset 0x03C4
LIGCR1	Lane I General Control Register 1	Offset 0x0404
LJGCR1	Lane J General Control Register 1	Offset 0x0444

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

L[A–J]GCR1 contains functional control bits for the SerDes logic on lanes A to J.

Note: The SerDes Port uses a base address of 0xFFFAD000.

Table 15-64 describes the L[A–J]GCR1 fields.

Table 15-65. L[A–J]GCR1 Field Descriptions

Bits	Reset	Description	Settings
— 31–27	0	Reserved. Write to zero for future compatibility.	
OPAD_CTL 26	Configured by Reset	**Tx Output Pad Control Signal for Common Mode** Selects the transmitter common mode.	0 Transmitter enabled. 1 Force transmitter output to common mode.
— 25–0	0	Reserved. Write to zero for future compatibility.	

15.10.58 Lane A–J Receive Equalization Control Register 0 (L[A–J]RECR0)

LARECR0	Lane A Receive Equalization Control Register 0	Offset 0x210
LBRECR0	Lane B Receive Equalization Control Register 0	Offset 0x250
LCRECR0	Lane C Receive Equalization Control Register 0	Offset 0x290
LDRECR0	Lane D Receive Equalization Control Register 0	Offset 0x2D0
LERECR0	Lane E Receive Equalization Control Register 0	Offset 0x310
LFRECR0	Lane F Receive Equalization Control Register 0	Offset 0x350
LGRECR0	Lane G Receive Equalization Control Register 0	Offset 0x390
LHRECR0	Lane H Receive Equalization Control Register 0	Offset 0x3D0
LIRECR0	Lane I Receive Equalization Control Register 0	Offset 0x410
LJRECR0	Lane J Receive Equalization Control Register 0	Offset 0x450

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—			GK20VD					—				GK30VD			
Type	R			R/W					R				R/W			
Reset	0	0	0	x	x	x	x	x	0	0	0	0	x	x	x	x
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	GK20VD_EN	GK30VD_EN	OSETOVD_EN	—							OSETOVD					
Type	R/W			R							R/W					
Reset	x	x	0	0	0	0	0	0	0	0	0	1	1	1	1	1

L[A–J]RECR0 contains functional control bits for the SerDes logic on lanes A to J.

Note: The SerDes Port uses a base address of 0xFFFAD000.

Table 15-66 describes the L[A–J]RECR0 fields.

Table 15-66. L[A–J]RECR0 Field Descriptions

Bits	Reset	Description	Settings
— 31–29	0	Reserved. Write to zero for future compatibility.	
GK2OVD 28–24	Configured by Reset	**K2 Gain Control Equalization Override** Selects equalization override value. Recommended settings per protocol: • PCI Express = 00000 • SGMII 1.25 Gbps = 01111 • Serial RapidIO 5.000 Gbps = 10000 • Serial RapidIO 3.125 Gbps = 00000 • Serial RapidIO 2.5 Gbps = 00000 • CPRI 6.144 Gbps = 10000 • CPRI 4.915 Gbps = 10000 • CPRI 3.072 Gbps = 00000 • CPRI 2.4576 Gbps = 00000 • CPRI 1.2288 Gbps = 01111	00000 Maximum Gain K2 Equalization 01111 Minimum Gain K2 Equalization All other values represent intermediate values between minimum and maximum.
— 23–20	0	Reserved. Write to zero for future compatibility.	

Table 15-66. L[A–J]RECR0 Field Descriptions (Continued)

Bits	Reset	Description	Settings
GK3OVD 19–16	Configured by Reset	**K3 Gain Control Equalization Override** Selects equalization override value. Recommended settings per protocol: • PCI Express = 0000 • SGMII 1.25 Gbps = 1111 • Serial RapidIO 5.00 Gbps = 0000 • Serial RapidIO 3.125 Gbps = 0000 • Serial RapidIO 2.5 Gbps = 0000 • CPRI 6.144 Gbps = 0000 • CPRI 4.9152 Gbps = 0000 • CPRI 3.072 Gbps = 0000 • CPRI 2.4576 Gbps = 0000 • CPRI 1.2288 Gbps = 1111	0000 Maximum Gain K3 Equalization 1111 Minimum Gain K3 Equalization All other values represent intermediate values between minimum and maximum.
GK2OVD_EN 15	Configured by Reset	**K2 Gain Control Override Enable** Enables K2 gain control override. Recommended settings per protocol: • PCI Express = 0000 • SGMII 1.25 Gbps = 1 • Serial RapidIO 5.00 Gbps = 0 • Serial RapidIO 3.125 Gbps = 0 • Serial RapidIO 2.5 Gbps = 0 • CPRI 6.144 Gbps = 0 • CPRI 4.9152 Gbps = 0 • CPRI 3.072 Gbps = 0 • CPRI 2.4576 Gbps = 0 • CPRI 1.2288 Gbps = 1	0 Use adaptation derived from K2 gain. 1 Use K2 gain override.
GK3OVD_EN 14	Configured by Reset	**K3 Gain Control Override Enable.** Enables K3 gain control override. Recommended settings per protocol: • PCI Express = 0000 • SGMII 1.25 Gbps = 1 • Serial RapidIO 5.00 Gbps = 0 • Serial RapidIO 3.125 Gbps = 0 • Serial RapidIO 2.5 Gbps = 0 • CPRI 6.144 Gbps = 0 • CPRI 4.9152 Gbps = 0 • CPRI 3.072 Gbps = 0 • CPRI 2.4576 Gbps = 0 • CPRI 1.2288 Gbps = 1	0 Use adaptation derived from K3 gain. 1 Use K3 gain override.

Table 15-66. L[A–J]RECR0 Field Descriptions (Continued)

Bits	Reset	Description	Settings
OSETOVD_EN 13	Configured by Reset	**Offset Override Enable** Used to enable the offset override value.	0 Use adaptation offset (recommended for all protocols). 1 Use offset override.
— 12–6	0	Reserved. Write to zero for future compatibility.	
OSETOVD 5–0	Configured by Reset	**Adaptive Equalization Offset Override** Value overrides the adaptive equalization offset control.	000000 +Maximum imposed offset 011111 0 imposed offset (recommended) 111111 –Maximum imposed offset All other values reserved.

15.10.59 Lane A–J Transmit Equalization Control Register 0 (L[A–J]TECR0)

LATECR0	Lane A Transmit Equalization Control Register 0	Offset 0x218
LBTECR0	Lane B Transmit Equalization Control Register 0	Offset 0x258
LCTECR0	Lane C Transmit Equalization Control Register 0	Offset 0x298
LDTECR0	Lane D Transmit Equalization Control Register 0	Offset 0x2D8
LETECR0	Lane E Transmit Equalization Control Register 0	Offset 0x318
LFTECR0	Lane F Transmit Equalization Control Register 0	Offset 0x358
LGTECR0	Lane G Transmit Equalization Control Register 0	Offset 0x398
LHTECR0	Lane H Transmit Equalization Control Register 0	Offset 0x3D8
LITECR0	Lane I Transmit Equalization Control Register 0	Offset 0x418
LJTECR0	Lane J Transmit Equalization Control Register 0	Offset 0x458

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		TEQ_TYPE		—								RATIO_PST1Q			
Type	R		R/W		R								R/W			
Reset	0	0	0	1	0	0	0	0	0	0	0	0	x	x	x	x
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—										AMP_RED					
Type	R										R/W					
Reset	0	0	0	0	0	0	0	0	0	0	x	x	x	x	x	x

L[A–J]TECR0 contains functional control bits for the SerDes logic on lanes A to J.

Note: The SerDes Port uses a base address of 0xFFFAD000.

Table 15-67 describes the L[A–J]TECR0 fields.

Table 15-67. L[A–J]TECR0 Field Descriptions

Bits	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
TEQ_TYPE 29–28	01	**Transmit Equalization Type** Selects the type and amount of transmit equalization.	00 No Tx equalization (recommended for SGMII 1.25 Gbps). 01 2 Levels of Tx Equalization (+1 post-cursor) (recommended for all others) 10 3 Levels of Tx Equalization (+1 pre-cursor and +1 post-cursor) 11 4 Levels of Tx Equalization (+1 pre-cursor and +2 post-cursors)
— 27–20	0	Reserved. Write to zero for future compatibility.	

Table 15-67. L[A–J]TECR0 Field Descriptions (Continued)

Bits	Reset	Description	Settings
RATIO_PST1Q 19–16	Configured by Reset	**Ratio Full Transition Bit to Pre-Cursor for 2-Tap Equalization** Selects the ratio value. Recommended values per protocol are as follows:. • PCI Express 2.5 Gbps = 1100 • PCI Express 5.0 Gbps = 1100 • Serial RapidIO = 1011 • SGMII 1.25 Gbps = 1000 • CPRI = 1011	For RATIO_PST1Q[3]: 0 Negative sign 1 Positive sign For RATIO_PST1Q[2–0]: 000 No equalization. 001 1.09x 010 1.20x 011 1.33x 100 1.50x 101 1.71x 110 2.00x 111 reserved
— 15–6	0	Reserved. Write to zero for future compatibility.	
AMP_RED 5–0	Configured by Reset	**Amplitude Reduction** Selects the amplitude reduction value. Recommended settings are as follows: • PCI Express = 000000 • Serial RapidIO 5.0 Gbps short run = 001101 • Serial RapidIO 5.0 Gbps long run = 000000 • Serial RapidIO 3.125/2.5 Gbps short run = 001000 • Serial RapidIO 3.125/2.5 Gbps long run = 000000 • SGMII 1.25 Gbps = 001011 • CPRI = 000000	For AMP_RED[5]: Reserved. For AMP_RED[4–0]: 00000 Full differential swing. 00011 0.9x full swing 01000 0.75x full swing 10100 0.5x full swing All other values represent intermediate values.

15.10.60 Lane A–J Test Control/Status Register 3 (L[A–J]TCSR3)

LATCSR3	Lane A Test Control/Status Register 3	Offset 0x23C
LBTCSR3	Lane B Test Control/Status Register 3	Offset 0x27C
LCTCSR3	Lane C Test Control/Status Register 3	Offset 0x2BC
LDTCSR3	Lane D Test Control/Status Register 3	Offset 0x2FC
LETCSR3	Lane E Test Control/Status Register 3	Offset 0x33C
LFTCSR3	Lane F Test Control/Status Register 3	Offset 0x37C
LGTCSR3	Lane G Test Control/Status Register 3	Offset 0x3BC
LHTCSR3	Lane H Test Control/Status Register 3	Offset 0x3FC
LITCSR3	Lane I Test Control/Status Register 3	Offset 0x43C
LJTCSR3	Lane J Test Control/Status Register 3	Offset 0x47C

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

L[A–J]TCSR3 contains functional control bits for the test control/status logic on lanes A to J.

Note: The SerDes Port uses a base address of 0xFFFAD000.

Table 15-67 describes the L[A–J]TCSR3 fields.

Table 15-68. L[A–J]TCSR3 Field Descriptions

Bits	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
LPBK_EN 29–28	Configured by Reset	**Loopback Enable** Enables loopback from the Tx serializer to the Rx amplifier to allow for on-chip system check. The recommended settings for all protocols is 00. **Note:** Loopback using PCI-Express requires Root Complex configuration for the PCI Express interface. End point loopback is not supported.	00 Application mode. 01 Digital loopback. 10 Reserved. 11 Reserved.
— 27–0	0	Reserved. Write to zero for future compatibility.	

16 Serial RapidIO Controller and Enhanced Message Complex

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 MSC8157E includes a serial RapidIO subsystem that supports two RapidIO ports and a shared enhanced messaging unit (eMSG) that comply with the *RapidIO Interconnect Specification, Revision 1.3* and includes some features defined by *Revision 2.1*. The RapidIO subsystem provides pass-through support, direct inbound I/O, indirect outbound I/O (using the HSSI DMA controllers), and direct messaging. Data throughput is managed using a buffer manager (BMLite) and a frame manager (FMLite).

The MSC8157E device can connect directly to a host, another MSC8157E device, or a serial RapidIO switch. Each port in the switch is point-to-point connected to the MSC8157E device through a serial RapidIO link that typically carries packets in both directions. Packets ready for processing are transported from the host to the MSC8157E, and the processed packets are transported from the MSC8157E device back to the host.

The Serial RapidIO controller directs the traffic flow between the MSC8157E and any other RapidIO device through the eMSG for messages and doorbells and through the RapidIO DMA channels for NWRITEs, NWRITE_Rs, NREADs, and SWRITEs.

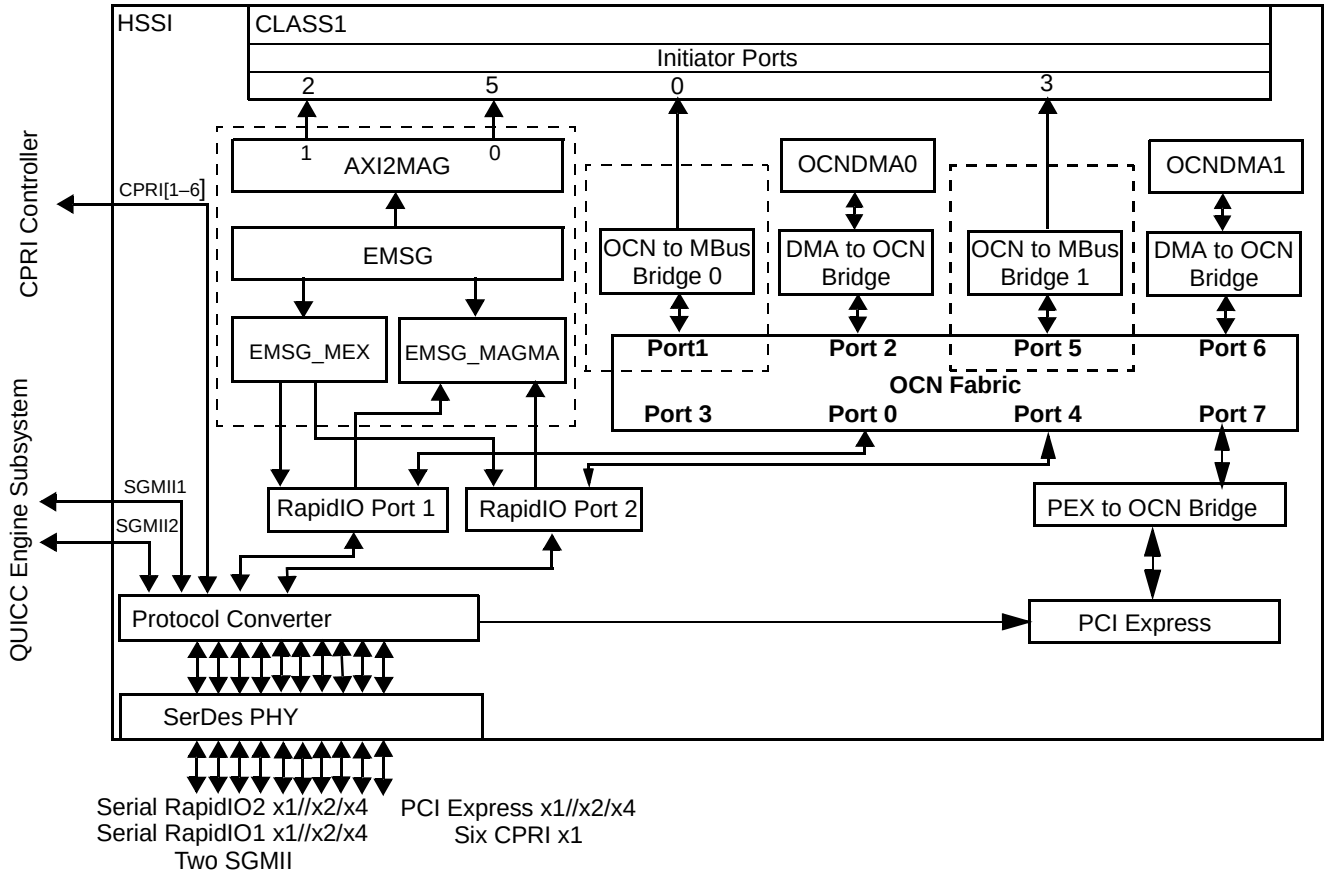
The RapidIO subsystem is supported by the HSSI (see **Chapter 15**, *High Speed Serial Interface (HSSI) Subsystem*), which includes the following:

- CLASS1 non-blocking fabric to provide a high-speed connection to the system CLASS
- Two HSSI DMA controllers for outbound I/O traffic
- OCN packet-based 64-bit interconnect fabric using 6 ports
- Protocol multiplexing unit
- Single 10-lane SerDes PHY that supports both RapidIO controllers in x4 or x2 or x1 mode at 1.25, 2.5, 3.125, or 5.0 Gbaud per lane.
- Performance monitoring

The host and the MSC8157E communicate as follows:

- The host sends messages to the destination MSC8157E 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 MSC8157E and the MSC8157E 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 MSC8157E 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 MSC8157E 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 MSC8157E 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.

Figure 16-1 shows a block diagram of the HSSI



Note: The actual signal multiplexed for each channel in the PHY is determined by the SerDes configuration field contents in the lower 32 bits of the reset configuration word, which are recorded in RCWLR[SP]. See **Chapter 5**, *Reset* for details.

Figure 16-1. HSSI Block Diagram

16.1 Serial RapidIO and eMSG Complex Overview

This section summarizes the Serial RapidIO controller and eMSG features, operating modes, and signal functionality.

16.1.1 Serial RapidIO Ports

The serial RapidIO interface provides a RapidIO port to communicate with other RapidIO devices. **Figure 16-2** shows the RapidIO port unit.

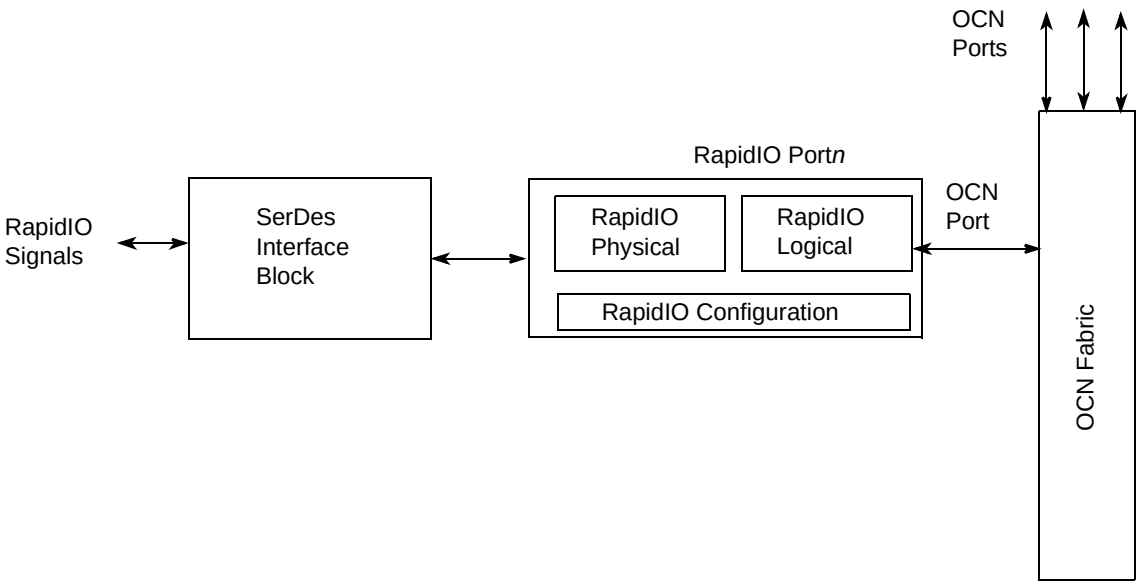


Figure 16-2. RapidIO Endpoint

The RapidIO port supports the following sections of the *RapidIO Interconnect Specification, Revision 2.1*, including all transaction flows and priorities:

- Small or large size transport information field only.
- 34-bit addressing.
- Up to 256-byte data payload.
- Up to eight outstanding unacknowledged RapidIO transactions.
- Hardware recovery only.
- All transaction flows and all priorities
- Critical request flow (CRF) as described in the *RapidIO Interconnect Specification, Revision 1.3*, Part 6: 1x/rx LP-Serial Physical Layer Specification.
- Register and register bit extensions as described in the *RapidIO Interconnect Specification, Revision 1.2, Part VIII: Error Management Extensions Specification*.
- Hot swap
- ATOMIC set/clr/inc/dec for read-modify-write operations

- IO_READ_HOME and FLUSH w/data for accessing cache-coherent data from a remote memory system
- Only supports receiver-controlled flow control
- Inbound transactions to the configuration registers are limited to 32-bit accesses only.
- Outbound maintenance transactions can be any valid size.

RapidIO endpoint supports the following user-defined features:

- Nine outbound ATMU windows with each window having up to 32 subwindows except the default window
- Five inbound ATMU windows
- Logical outbound packet time-to-live counter to prevent local processor from hanging when the RapidIO interface fails
- Accept-all mode of operation for failover support
- RapidIO random bit error injection
- Performance monitor interface

RapidIO endpoint supports the following features of RapidIO LP-Serial:

- 1x, 2x, and 4x LP-Serial link interfaces
- Transmission rates of 1.25, 2.5, 3.125, and 5 Gbaud (data rates of 1.0, 2.0, 2.5, and 4.0 Gbps) per lane
- Auto detection of 1x, 2x, and 4x mode operation during port initialization
- Error detection for packets and control symbols
- Support for link initialization, synchronization, error recovery, and time-out

The individual RapidIO endpoint does not support the following features of RapidIO LP-Serial:

- RapidIO endpoint cannot be configured as four 1x or two 2x ports

16.1.2 eMSG Unit

The eMSG supports the following:

- *RapidIO Interconnect Specification Revision 1.3*, Part 1: NWRITE and Streaming writes.
- *RapidIO Interconnect Specification Revision 1.3*, Part 2: Message Passing Logical Specification.
- *RapidIO Interconnect Specification Revision 1.3*, Part 10: Data Streaming Logical Specification.
- *RapidIO Interconnect Specification Revision 2.0*, Part 10: Stream Management Flow Control. Basic stream management flow control (XON/XOFF) using extended header message format.

- 4 Kbyte virtual memory space for partition of resources in a multi-core environment.
- 32-bit address space.
- 24 concurrent inbound reassembly operations with flow-based reservation of reassembly resources.
- One or more Type11 outbound message units with the following features:
  - byte transfers with automatic padding to double-word size.
  - supports multicast and non-multicast single segment messages to 32 RapidIO destinations and non-multicast multiple segment messages.
  - transmitting to any mailbox and extended mailbox for a single segment message.
  - segment size up to 256 bytes.
  - up to sixteen segment messages with a total payload of up to 4 Kbyte.
  - one entire message (up to 16 message segments) can be transmitted before receiving any response.
  - one entire single segment message to all multicast destination (up to 32) can be transmitted before receiving any response.
- One or more Type5 NWrite outbound units with up to 64 Kbyte total data payload.
- One or more Type6 outbound streaming write outbound units with up to 64 Kbyte total data payload.
- One or more Type11 message inbound classification units with the following features:
  - transaction steering through message header classification.
  - segment size up to 256 bytes.
  - up to sixteen segment messages with a total payload of up to 4 Kbyte.
  - all message responses are upgraded to priority 3.
- One or more Type9 data streaming inbound classifications units with the following features:
  - transaction steering through packet header classification.
  - support for concurrent interleaved Protocol Data Units (PDUs).
- One or more Type9 data streaming outbound segmentation units with up to 64 Kbyte total data payload.
- One or more Type10 doorbell inbound classification units:
  - transaction steering through doorbell header classification.
  - all doorbell responses are upgraded to priority 3.
- One or more Type10 doorbell outbound units with ordering maintained with respect to other types of traffic.
- One or more Type8 port-write inbound classification units with the following features:
  - transaction steering through port-write header classification.
  - data payloads of 4 to 64 bytes.
- One or more Type8 outbound port-write units with data payloads of 4 to 64 bytes.

- Multicast mode for Type11 in which a single segment message (256 bytes or less) is sent to multiple destinations.

RapidIO endpoint does not support or has limited support for the following features of the *RapidIO Interconnect Specification, Revision 1.3*:

- 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:

- Both controllers supports x4/x2/x1 modes
- Transmission rates of 1.25, 2.5, 3.125, and 5.0 Gbaud (data rates of 1.0, 2.0, 2.5, and 4 Gbps, respectively).
- Auto detection of x1/x2/x4 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 x1/x2/x4 operation:

- RapidIO endpoint cannot be configured as four x1 ports.

16.1.3 Internal Processing Support

The block includes the following support modules:

- *Queue Manager (QMLite)*. Facilitates the processing of message queues using 16-byte command descriptors (CD).
- *Buffer Manager (BMLite)*. Performs a memory reservation resource allocation function by reserving sets of buffers in memory (internal in M2/M3 and external in DDR memory) for use in message processing. By maintaining blocks of available reserved memory buffer space, the queue manager and eMSG unit can request and have access to memory without additional arbitration or the risk of the memory location being overwritten by another master device. The individual producers and consumers increment their respective indices by the number of entries they are producing/consuming. Their PI/CI registers wrap to 0x0 at twice the size of the ringer buffer. For example, the values for a 16-entry PI/CI ring wrap to 0x0 when they reach 0x1F. A 32-entry ring wraps to 0 when it reaches 0x3F, and so on. Using PI/CI values that are one bit wider than needed to address the ring entries means that the *(PI/CI) mod (index–width)* gives the correct fit of the ring buffer over the entire range where full can be distinguished from empty.

16.1.4 Operating Modes

The main operating modes of the RapidIO ports are as follows:

- x1/x2/x4 LP-Serial link interfaces (both controllers)
- Transmission rates of 1.25. 2.5, 3.125, and 5.0 Gbaud (data rates of 1.0, 2.0, 2.5, and 4.0 Gbps, respectively).

Note: Transferring high data rates at 5 Gbaud in x1 mode is not recommended and can result in excessive errors (including CCS, CRC, DE, and PE) and dropped data. For high data rates, use 5 Gbaud in x2 or x4 mode. For lower data rates, use 3.125 Gbaud, 2.5 Gbaud, or 1.25 Gbaud in x1, x2, or x4 mode.

- 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.

16.1.5 x1/x2/x4 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 x1 or x4 link. One differential pair output per link. This implementation supports data rates of 1.25, 2.5, 3.125, or 5 Gbaud.	Primary output pads. Asynchronous outputs.	O	—	x
SD_RX[0–3]/ SD_RX[0–3]	**Receive Data** Serial data input for x1 or x4 link. One differential pair input per link. This implementation supports data rates of 1.25, 2.5, 3.125, or 5 Gbaud.	Primary input pads. Asynchronous inputs.	I	—	x

16.1.6 RapidIO Interface Activation

There are two basic activation scenarios:

- Boot over the Serial RapidIO interface
- Non-boot operation

16.1.6.1 Initialization for Booting the MSC8157E DSP

When the RapidIO interface is use to boot the MSC8157E, 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 x4 sync sequence ends and the master device can begin accessing the RapidIO interface.

16.1.6.2 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 MSC8157E, for example. Sync signals are received over the Rx lines, although the Tx lines remain tri-stated. The device is unable to sync in x4 mode and times out because the Tx lines are tri-stated, and the interface syncs on x1 mode. To sync on x4 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 x4 mode.

16.1.7 Link Training

During port initialization, the serial RapidIO port performs lane synchronization (detecting valid symbols on a lane) and lane alignment (coordinating multiple lanes to receive valid data across lanes). Internal errors in lane synchronization and lane alignment may cause failure to achieve link initialization at the configured port width. An SRIO port configured as an x4 port may see one of three scenarios:

- One or more lanes fail to achieve lane synchronization. Depending on which lanes fail, this may result in downtraining from x4 to x2, x4 to x1 on lane 0, x4 to x1 on lane R, x2 to x1 on lane 0, or x2 to x1 on lane R. (R indicates a lane other than lane 0).
- The link may fail to achieve lane alignment as a x4, even though all four lanes achieve lane synchronization, and downtrain to x2 or x1 mode.
- If the link downtrains to x2, it may fail to complete link initialization (that is, PnCCSR[PU]=0, indicating that the port initialized, but [PO]=0, indicating that the port is not OK).

A serial RapidIO port configured as x2 may downtrain to x1 on lane 0 or lane R because of the first scenario. A serial RapidIO port configured as n x1 may fail to complete port initialization

(PnCCSR[PU] will never deassert) because of the same scenario. Once a port completes link initialization successfully, it will operate normally.

To facilitate proper training, perform the following software sequences on each serial RapidIO port prior to initiating any traffic to that port.

16.1.7.1 Initialize Link

1. Ensure that the configured port width matches the actual port width in the system.
   a. If the configured port width is wider than the width supported by the link partner or system, the SRIO port will train down to the highest common denominator, causing a false detection of a training issue.
   b. The configured port width per port is readable in SERDES SRDSPCCR0.
2. Start a counter set to ~2 ms after the SERDES reset is complete (SERDES SRDSBnRSTCTL[RST_DONE]=1 for n corresponding to the SERDES bank/PLL for the SRIO port).
3. Poll the port uninitialized status (SRIO PnESCSR[PO]) until PO=1 or the counter expires. If the counter expires, the port has failed initialization: go to Reset Link
4. Read SRIO PnCCSR
   a. For a port configured as a x4, if IPW≠b010, the port has failed initialization: go to Reset Link.
   b. For a port configured as a x2, if IPW≠b011, the port has failed initialization: go to Reset Link
5. Continue with normal SRIO initialization procedures or software retraining sequence

16.1.7.2 Reset Link

1. Set SRIO PnCCSR[PD]=1
2. Wait 50 µs.
3. Set SERDES LmGCR0BnGCRm0[RRST]=0 for each bank n and lane m used by the SRIO port
4. The lanes used by the SRIO port are readable in SERDES SRDSPCCR0.
5. Read SERDES LmGCR0BnGCRm0 for each bank n and lane m in step 3
6. Wait ≥ 100 ns.
7. Set SERDES LmGCR0BnGCRm0 [RRST]=1 for each bank n and lane m in step 3
8. Read SERDES LmGCR0BnGCRm0 for each bank n and lane m in step 3
9. Wait ≥ 300 ns.
10. Write 1s to clear all bits in SRIO PnSLCSR
11. Set SRIO PnCCSR[PD]=0.

12. Go to Initialize Link (**Section 16.1.7.1**).

16.1.7.3 Software Retraining

If the serial RapidIO port needs to retrain for any reason (for example, link partner reset), execute the following modified software retraining sequence:

1. Software on the host must ensure that all RapidIO transactions have completed and link activity has stopped.
2. Set SRIO PnCCSR[PD]=1
3. Set SRIO PnPCR[OBDEN] on the host to enable the discarding of any pending packets. (There should be none if proper steps were taken to ensure there was no link activity.)
4. Clear SRIO PnPCR[OBDEN] on the host.
5. Configure new operating width (via PnCCSR[PWO]) or any other new configurations.
6. Go to Reset Link

Note: Step 1 of the Reset Link procedure is redundant with step 2 of Software Retraining sequence, but is not harmful to repeat.

Note: The retraining sequence resumes at step 7 after completing the Initialize Link sequence.

7. Poll PnESCSR[PO] on the host until it is set, which indicates the link has attained port and link initialization.
8. Poll PnESCSR[PO] on the agent until it is set, which indicates the link has attained port and link initialization.
9. Poll PnESCSR[OES] on the agent until it is clear, which indicates the link has completed the error recovery sequence initiated from the port disable.
10. Poll PnESCSR[OES] on the host until it is clear, which indicates the link has completed the error recovery sequence initiated from the port disable.
11. Clear the agent error and status registers.
12. Clear the host error and status registers.
13. Begin normal packet transfer.

16.1.8 Special Case of x2/x1 Modes

RapidIO devices that support x2 mode as well as x1 mode transmit data on both lane 0 and lane 1 even when operating as a x1 port. The Serial RapidIO specification requires a x1/x2 Mode Detect State Machine per port to distinguish between a x2 port (transmitting striped data) and a x1 port (transmitting identical data on lanes 0 and 1). The Freescale RapidIO controller may fail to

complete training if it tries to connect to a RapidIO device that supports x2 mode and x1 mode if the devices transmits the same data on lanes 0 and 1 when in x1 mode.

Note: Freescale Serial RapidIO controllers transmit only on lane 0 if initialized as x1 on lane 0 and only transmit on lane 1 when initialized as x1 on lane 1.

Always ensure that the serial RapidIO controller is configured to match the configured width of the link partner. The preferred mechanism is through the RCW configuration. To recover after detecting a training failure due to mismatched width configuration and before restarting traffic, execute the Software Retraining sequence documented in **Section 16.1.7.3** and disable x2 operation by configuring PnCCSR[PWO] equal to 0b010, 0b011, or 0b110.

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 the RapidIO I/O transactions listed in **Table 16-2**.

Table 16-2. RapidIO I/O Transactions

IO Transaction	FTYPE	TTYPE	Status	Description
NREAD	0010	0100	NA	Read
ATOMIC inc		1100		Read and post-increment with response
ATOMIC dec		1101		Read and post-decrement with response
ATOMIC set		1110		Read and set to all-1s with response
ATOMIC clr		1111		Read and set to all-0s with response
NWRITE	0101	0100		Write with no response
NWRITE_R		0101		Write with response
SWRITE	0110	N/A		Streaming-Write
MAINT read	1000	0000		Maintenance read
MAINT write		0001		Maintenance write
MAINT read response		0010	0000	Done maintenance read response
			0111	Error response
MAINT write response		0011	0000	Done maintenance write response
			0111	Error response
MAINT port-write		0100	NA	Maintenance port-write (

Table 16-2. RapidIO I/O Transactions (Continued)

IO Transaction	FTYPE	TTYPE	Status	Description
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

16.2.2 Message Passing

The serial RapidIO interprocessor communication can use explicit messages instead of going through shared memory. A message can consist of up to 16 packets of up to 256 Byte (4 Kbyte maximum) or streaming up to 64 Kbyte. The receiver defines the memory location for the messages to be buffered. **Figure 16-3** shows how messages are passed.

Figure 16-3. Message Passing

Table 16-3 summarizes the RapidIO message passing transactions that are supported by this RapidIO endpoint.

Table 16-3. RapidIO Message Passing Transactions

MSG Transaction	FTYPE	TTYPE	Status	Description
DOORBELL	1010	NA	NA	Doorbell
MESSAGE	1011			Message
RESPONSE without data	1101	0000	0000	Doorbell done response
			0011	Doorbell retry response
			0111	Doorbell error response
		0001	0000	Message done response
			0011	Message retry response
			0111	Message error response

16.2.3 RapidIO Data Streaming (Type9) Transactions

Table 16-4 summarizes the RapidIO data streaming (type 9) transactions that are supported by this RapidIO endpoint.

Table 16-4. RapidIO Data Streaming Transactions

Data Streaming Header Type	FTYPE	Start	End	XH	Description
START	1001	1	0	0	Start segment packet.
SINGLE	1001	1	1	0	Single segment packet.
CONTINUATION	1001	0	0	0	Continuation segment packet.
END	1001	0	1	0	End segment packet.
FLOW CONTROL	1001	-	-	1	Flow Control packet.

16.2.4 RapidIO GSM Transactions

Table 16-5 summarizes the RapidIO GSM transactions that are supported by this RapidIO endpoint.

Table 16-5. RapidIO GSM Transactions

GSM Transaction	FTYPE	TTYPE	Status	Description
IO_READ_HOME	0010	0010	NA	I/O Read Home (limited to RapidIO packet generation only)
FLUSH w/data	0101	0001		GSM Flush with data (limited to RapidIO packet generation only)
RESPONSE without data	1101	0000	0000	GSM done response
			0011	GSM retry response
			0101	GSM done intervention response
			0111	GSM error response
RESPONSE with data		1000	0000	GSM done response
			0001	GSM data only response

16.2.5 RapidIO Packet Format

Table 16-6 summarizes the small transport field packet formats of RapidIO transaction types for LP-Serial operation. Details on packet fields are located in the *RapidIO Interconnect Specification, Revision 2.0.1*.

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 MSC8157E 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.RapidIO Control Symbol Summary

Table 16-6. RapidIO Small Transport Field Packet Format

Transaction	Bits																
	32								32					16			64
	5	2	1	2	2	4	8	8	4	4	8	8	8	13	1	2	
NREAD, ATOMIC (inc/dec/inc/dec), IO_READ_HOME	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	rdsize	src TID	addr			wd ptr	xam bs	NA
NWRITE_R, FLUSH w/data	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	wrsize	src TID	addr			wd ptr	xam bs	dword 0 -> dword n
NWRITE	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	wrsize	don't care	addr			wd ptr	xam bs	dword 0 -> dword n
SWRITE	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	addr(29), rsv(1)=0, xambs(2)					dword 0 -> dword n			

Table 16-6. RapidIO Small Transport Field Packet Format (Continued)

Transaction	Bits																
	32								32					16			64
	5	2	1	2	2	4	8	8	4	4	8	8	8	13	1	2	
MAINT read	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	rdsize	src TID	hop cnt	cfg offset		wd ptr	rsv= 0	NA
MAINT write	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	wrsize	src TID	hop cnt	cfg offset		wd ptr	rsv= 0	dword 0 (32-bit)
MAINT port-write	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	wrsize	rsv=0	hop cnt	rsv=0		wd ptr	rsv= 0	dword 0 -> dword n
MAINT response without data	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	hop cnt	rsv=0				NA
MAINT response with data	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	hop cnt	rsv=0				dword 0 (32-bit)
RESPONSE without data	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	NA					
RESPONSE without data for message	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	letter(2), mbox(2), msgseg(4)	NA					
RESPONSE with data	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	dword 0 -> dword n					
DOORBELL	ack ID	rsv =0	crf	prio	tt	ftype	des tID	src ID	rsv=0		src TID[1]	Info-msb	Info-lsb	NA			
MESSAGE	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	msglen(4), ssize(4), letter(2), mbox(2), msgseg(4)			dword 0 -> dword n					
DATA STREAMING (start)	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	cos		s(1),e(1), rsv(3), xh(1)=0, rsv(2)	streamID		hword 0 -> hword n			
DATA STREAMING (single)	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	cos		s(1),e(1), rsv(3), xh(1)=0, o(1),p(1)	streamID		hword 0 -> hword n			
DATA STREAMING (continuation)	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	cos		s(1),e(1), rsv(3), xh(1)=0, rsv(2)	hword 0 -> hword n					
DATA STREAMING (end)	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	cos		s(1),e(1), rsv(3), xh(1)=0, o(1),p(1)	length		hword 0 -> hword n			
DATA STREAMING (Flow Control)	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	cos		rsv(2), xtype(3), xh(1)=1, rsv(2)	length		hword 0 -> hword n			

1. src TID[3] of [0:7] bus must be set to ‘1’ for doorbell accesses.

16.2.6 RapidIO Control Symbol Summary

Table 16-7 summarizes the x1/x2/x4 LP-Serial control symbols and their format. Refer to the *RapidIO Interconnect Specification, Revision 1.3* Part VI: Physical Layer x1/x2/x4 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 16-7. x1/x2/x4 LP-Serial Control Symbol Format

Bits						Description
24						
stype0	param0	param1	stype1	cmd	CRC	
3	5	5	3	3	5	
000	pkt_ackID	buf_stat	—		crc	Packet accepted
001	pkt_ackID	buf_stat	—		crc	Packet retry
010	pkt_ackID	cause	—		crc	Packet not accepted Cause: 00001 Received unexpected ackID on packet. 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

Table 16-7. x1/x2/x4 LP-Serial Control Symbol Format (Continued)

Bits						Description
24						
stype0	param0	param1	stype1	cmd	CRC	
3	5	5	3	3	5	
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
—			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)

16.2.7 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.

16.2.7.1 Inbound Maintenance Accesses

There are two recommended methods by which RapidIO transactions can target RapidIO configuration register space in local memory.

The first method is based on RapidIO NREAD and NWRITE_R requests hitting a RapidIO address window defined by the Local Configuration Space Base Address Command and Status Register (LCSxBA1CSR), which is described on **page 16-190**. If external configuration accesses are disabled (LLCR[ECRAB] = 1; see **page 16-218**), 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.

16.2.7.2 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.

16.2.7.3 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.

16.2.8 Interaction with the Message Unit

The RapidIO controller is designed to interact with a message unit, which provides it with a message passing architecture. Please see **Section 16.1.2**, *eMSG Unit*, on page 16-5 for details on the message passing architecture. There are both inbound (Rx) and outbound (Tx) interfaces with the message unit, in addition to the Rx and Tx interfaces with the system and link.

Figure 16-4 shows RapidIO controller interaction with the message unit.

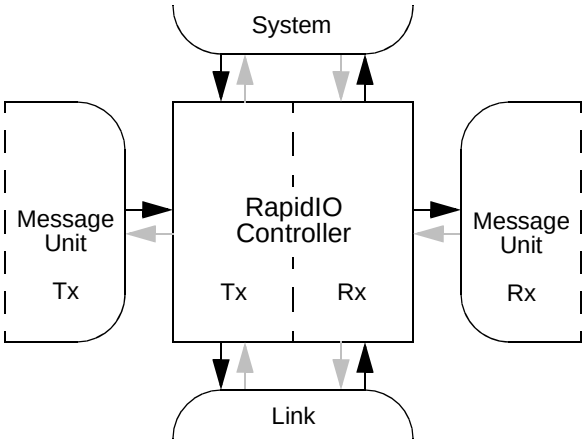


Figure 16-4. Interaction with the Message Unit

16.2.8.1 Inbound (Rx)

The inbound (Rx) interaction with the message unit is used receive message unit packets from the RapidIO link. The RapidIO link can receive message requests, responses, or flow control packets. The inbound interaction directs incoming packets to either the message unit or the controller based upon various RapidIO fields. Inbound message packets will be buffered into either a request or response queue before being sent out on the magenta interface. A round robin arbitration scheme is used to select the next transaction out to the message unit on the magenta bus.

16.2.8.2 Outbound (Tx)

The outbound (Tx) interaction with the message unit is used to transmit message unit packets to the RapidIO link. The message unit can transmit requests, responses, or flow control packets. Tx message unit packets are buffered within the RapidIO controller until they have been accepted on the link (received ack accepted). In addition, Tx message unit packets arbitrate with Tx system packets within the RapidIO controller before being transmitted to the link.

16.2.8.3 Buffer Allocation

The RapidIO controller contains buffers for Tx message unit packets. The buffers are separated into two categories...

- Tx message unit request packets
- Tx message unit response / flow control packets

Note: **Figure 16-5** shows the buffers for Tx message unit packets within the RapidIO controller.

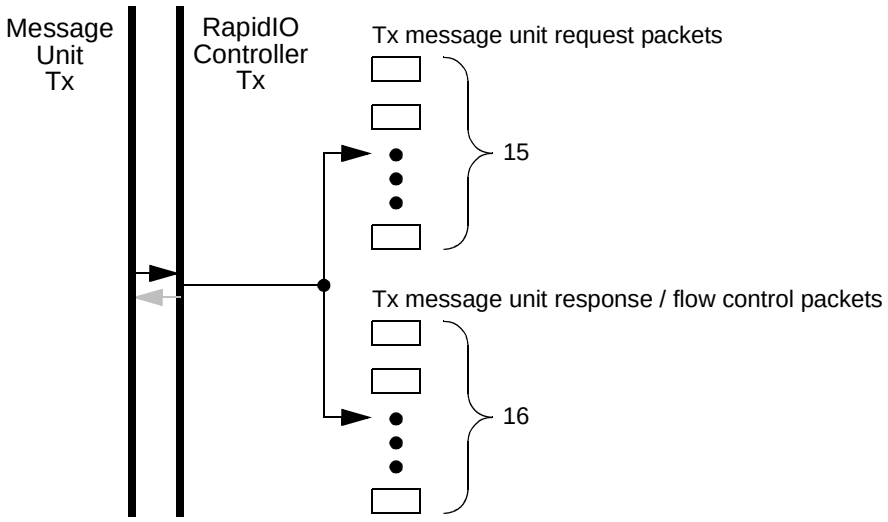


Figure 16-5. Tx Message Unit Packet Buffers

Within each category, there is dedicated and generic buffer allocation. Dedicated buffer allocation holds a specific type of message unit packet (within that category). Generic buffer allocation holds any type of message unit packet (within that category). Each buffer, regardless of its allocation (dedicated or generic), can only hold 1 message unit packet.

16.2.8.3.1 Tx Message Unit Request Packets

Tx message unit request packets have 8 types...

- Tx message unit request packet of arbitration group 0
- Tx message unit request packet of arbitration group 1
- Tx message unit request packet of arbitration group 2
- Tx message unit request packet of arbitration group 3
- Tx message unit request packet of arbitration group 4
- Tx message unit request packet of arbitration group 5
- Tx message unit request packet of arbitration group 6
- Tx message unit request packet of arbitration group 7

Each type mentioned above may have its own dedicated buffer allocation. In addition, there is also a generic buffer allocation (holds any of the types mentioned above). The number of dedicated and generic buffers for Tx message unit request packets is configurable with the following registers...

- **Section 16.4.1.53**, *Port n Message Request Tx Buffer Allocation Configuration Register 0 (PnMReqTxBACR0)*, on page 16-232
- **Section 16.4.1.54**, *Port n Message Request Tx Buffer Allocation Configuration Register 1 (PnMReqTxBACR1)*, on page 16-233

- **Section 16.4.1.55**, *Port n Message Request Tx Buffer Allocation Configuration Register 2 (PnMReqTxBACR2)*, on page 16-235

Note: There are only 15 buffers that hold message unit request packets. If the total value of the fields within the registers mentioned above exceeds 15 undefined behavior will result.

These registers are overwritten when the maximum arbitration group changes, which occurs with a write to a QMLite (OBMQDSCR1[AGQ7]) register. The new values support 1 dedicated buffer for each arbitration group, up to the maximum, and the remaining buffers are generic. The maximum total of all buffers (both dedicated and generic) is 15.

For example, when the maximum arbitration group is set to 7, then DedAG0[3:0] - DedAG7[3:0] is set to 0x1 (1 dedicated buffer for each arbitration group) and GenAG[3–0] is set to 0x7 (the remaining buffers are generic).

Another example, when the maximum arbitration group is set to 5, then DedAG0[3:0] - DedAG5[3:0] is set to 0x1, DedAG6[3:0]–DedAG7[3:0] are set to 0x0, and GenAG[3:0] is set to 0x9.

A user can change these registers after the maximum arbitration group changes. The new user values are used. A subsequent change of the maximum arbitration group overwrites the values of these registers again.

16.2.8.3.2 Tx Message Unit Response/Flow Control Packets

Tx message unit response/flow control packets have 2 types...

- Tx message unit response packet
- Tx message unit flow control packet

Each type may have its own dedicated buffer allocation. In addition, there is also generic buffer allocation (holds any of these types).

The number of dedicated and generic buffers for Tx message unit response/flow control packets is configurable using the following register...

- **Section 16.4.1.56**, *Port n Message Response/Flow Control Tx Buffer Allocation Configuration Register (PnMRspFcTxBACR)*, on page 16-236

Note: There are only 16 buffers that hold message unit response and flow control packets. The total value of the fields within the register mentioned above, must not exceed 16 (undefined behavior will result).

16.2.8.3.3 Arbitration

The RapidIO controller arbitrates amongst its Tx message unit packet buffers and an arbitration winner from the system packet buffers. Two main types of arbitration are used throughout: strict priority and weighted round robin. Strict priority arbitration, with starvation avoidance selects a winner based on priority. Starvation avoidance, which may be disabled, elevates lower priority requestors to the highest priority. This can only occur when a starvation avoidance counter has expired. This counter only counts when a requestor has lost. A counter threshold of 0s will disable this feature.

Note: Disabling starvation avoidance can cause deadlocks.

Weighted round robin arbitration selects a winner based on round robin, but with weights. The weights delay moving to the next requestor until a number, equal to the corresponding weight, of winners has been chosen. **Figure 16-6** shows the details of Tx message unit packet arbitration.

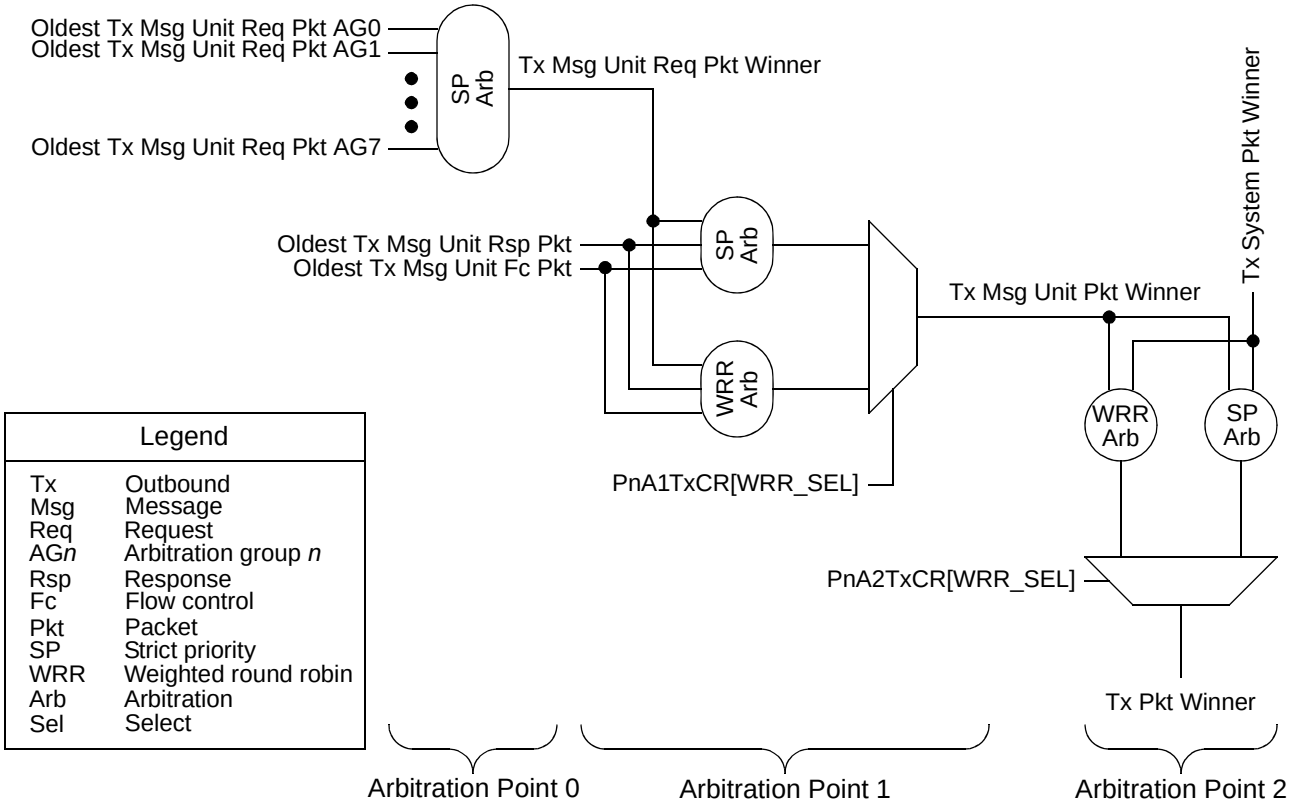


Figure 16-6. Tx Message Unit Packet Arbitration

16.2.8.3.3.1 Arbitration Point 0

Arbitration point 0 chooses a Tx message unit request packet winner. Only the oldest Tx message unit request packet of each arbitration group is eligible. Strict priority arbitration is used to select the winner. Tx message unit request packets of arbitration group 0 are the highest priority, and Tx message unit request packets of arbitration group 7 are the lowest.

Note: Arbitration point 0 is configurable with the following register...

- **Section 16.4.1.50**, *Port n Arbitration 0 Tx Configuration Register (PnA0TxCR)*

16.2.8.3.3.2 Arbitration Point 1

Arbitration point 1 chooses a Tx message packet winner. Only the Tx message unit request packet winner (chosen from arbitration point 0), the oldest Tx message unit response packet, and the oldest Tx message unit flow control packet are eligible. Two types of arbitration are available: strict priority and weighted round robin. If strict priority arbitration is selected, then the weighted round robin arbitration result is ignored. The RapidIO priorities (CRF and 2-bit priority field in the packet's header) of the 3 requestors (mentioned above) are compared. The packet with the highest RapidIO priority is chosen as the winner. In the case of a tie, round robin arbitration, based on past history, is used to select the winner.

If weighted round robin arbitration is selected, then the strict priority arbitration result is ignored. Each of the 3 requestors is considered, based on its corresponding weight, and a winner is chosen.

Arbitration point 1 is configurable with the following register...

- **Section 16.4.1.51**, *Port n Arbitration 1 Tx Configuration Register (PnA1TxCR)*

16.2.8.3.3.3 Arbitration Point 2

Arbitration point 2 chooses a Tx packet winner. Only the Tx message unit packet winner (chosen from arbitration point 1) and the system packet winner are eligible. Two types of arbitration are available: strict priority and weighted round robin. If strict priority arbitration is selected, then the weighted round robin arbitration result is ignored. The RapidIO priorities (CRF and 2-bit priority field in the packet's header) of the 2 requestors (mentioned above) are compared. The packet with the highest RapidIO priority is chosen as the winner. In the case of a tie, round robin arbitration, based on past history, is used to select the winner.

If weighted round robin arbitration is selected, then the strict priority arbitration result is ignored. Each of the 2 requestors is considered, based on its corresponding weight, and a winner is chosen.

Arbitration point 2 is configurable with the following register...

- **Section 16.4.1.52**, *Port n Arbitration 2 Tx Configuration Register (PnA2TxCR)*

16.2.9 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 MSC8157E 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.

16.2.9.1 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 16-7** 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.

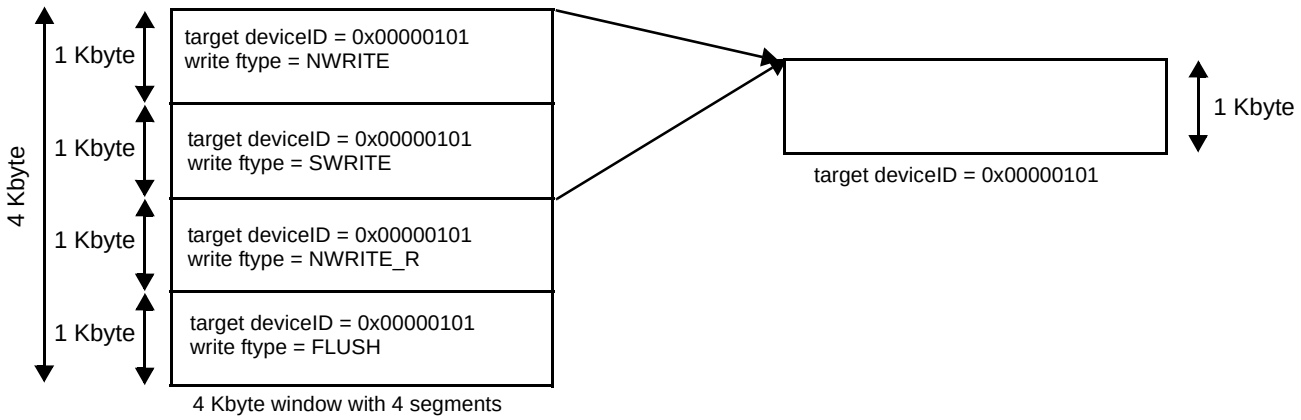


Figure 16-7. Single Target Example

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 16-8** 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.

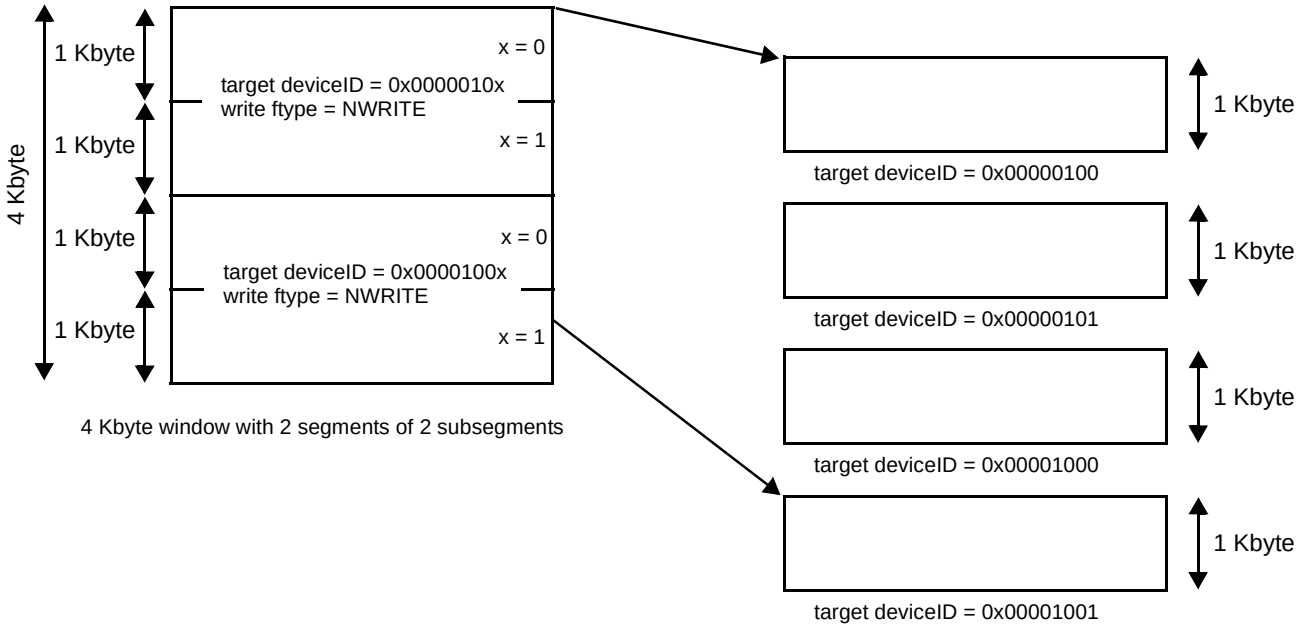


Figure 16-8. Multi-Targeting Example

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.

16.2.9.2 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.

16.2.9.3 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 16-8 lists the various combination options.

Table 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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 16-8. 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.

16.2.9.3.1 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.

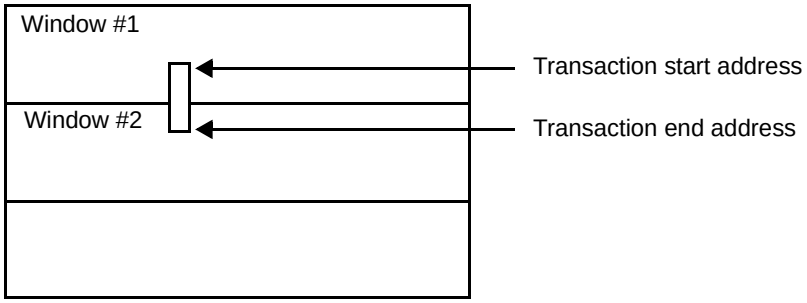


Figure 16-9. Valid Hit that Extends Into a Lower Priority Window

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.

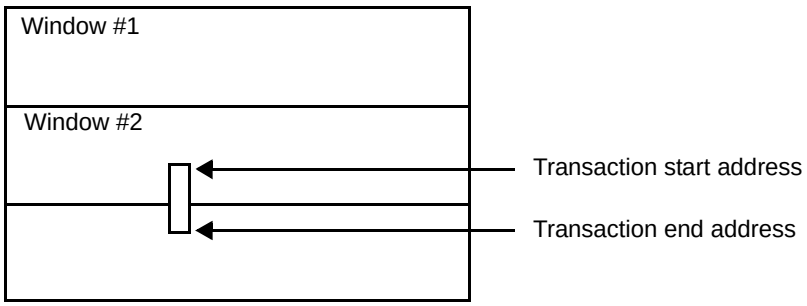


Figure 16-10. Valid Hit that Extends Beyond the Window Boundary

16.2.9.3.2 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.

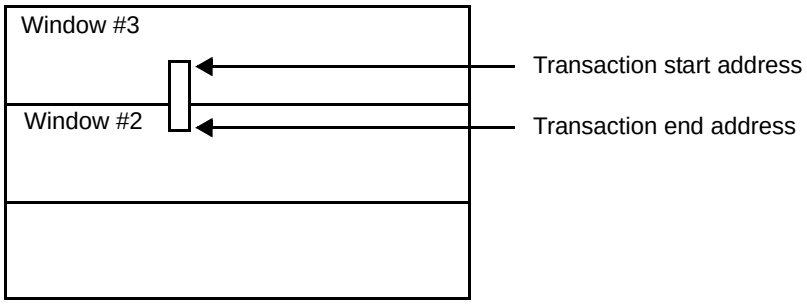


Figure 16-11. Boundary Crossing Error Due to Extension Into a Higher Priority Window

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.

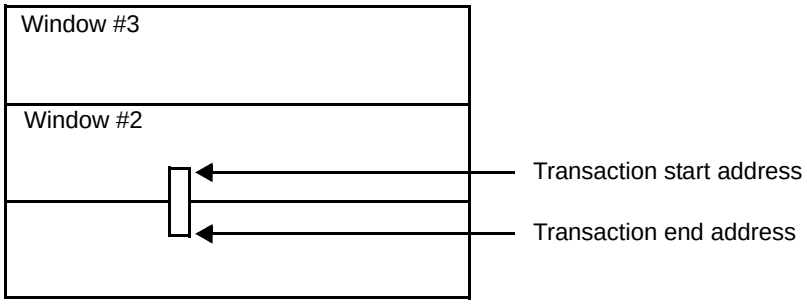


Figure 16-12. Boundary Crossing Error Due to Extension Beyond the Higher Priority Window Boundary

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.

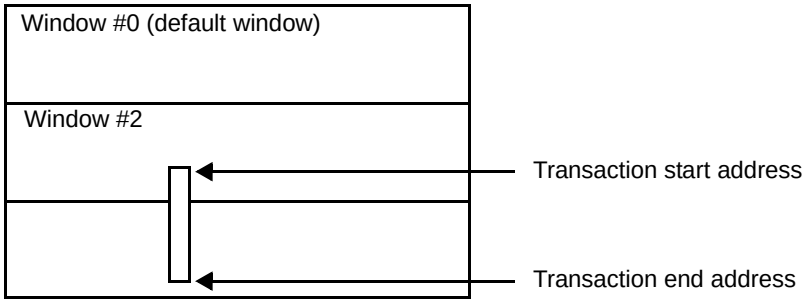


Figure 16-13. Boundary Crossing Error Due to Transaction Size Exceeding the Window Size

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.

16.2.9.4 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 MSC8102 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 16-9** describes the registers for configuring the window parameters, along with the number of the page where each register is described in detail.

Table 16-9. ATMU Registers for Configuring Window Parameters

ATMU Registers	Acronym	Description	Page
Port 1–2 RapidIO Inbound Window Translation Address Registers 1–4	PnRIWTAR[1–4]	Define the starting-point for the RapidIO address translation.	**page 16-244**
Port 1–2 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 16-245**
Configuring the Four Comparison Windows			
The Port 1–2 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 16-245**
Port 1–2 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 16-244**
Port 1–2 RapidIO Inbound Window Attributes Register 0	PnRIWAR0		**page 16-245**

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]}.

16.2.9.4.1 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.

16.2.9.4.2 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.

16.2.10 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).

16.2.11 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 16.2.13**, *Software Assisted Error Recovery Register Support*, on page 16-59.

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).

16.2.12 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.

16.2.13 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.

16.2.14 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 1.3,* part VII (Error Management Extensions Specifications).

16.2.14.1 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 1.3*. In these cases, the appropriate bit is set in the Port 1–2 Error Detect CSR. Only the packet containing the first detected recoverable error that is enabled for error capture (by the Port 1–2 Error Enable CSR) is captured in the Port 1–2 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 Portn 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 1–2 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.

16.2.14.2 Physical Layer RapidIO Errors

Table 16-10 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 16-211**) allows the error to increment the error rate counter and lock the Port 1–2 Error Capture registers—and also which PnEDCSR bit is set when the error is detected (see **page 16-210**).

Table 16-11 shows the RapidIO endpoint behavior after certain preset limits are exceeded (degraded threshold, failed threshold, retry threshold). **Table 16-12** shows the threshold response.

Table 16-10. 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
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

Table 16-10. Physical RapidIO Errors Detected (Continued)

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable/ Detect
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
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 16-11. 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]
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).

16.2.14.3 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 1.3*, part VII (Error Management Extensions Specifications).

Table 16-12 through **Table 16-26** 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 16-12. 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	
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 16.2.9.4.2**, *Window Boundary Crossing Errors,* on page 16-56.	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.				

Table 16-12. Hardware Errors For NREAD Transaction (Continued)

Error	Interrupt	Status Bit Set	Error Response	Comments
The Logical/Transport Layer Address Capture Command and Status Register described on **page 16-207** 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 16-13. 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.				

Table 16-13. Hardware Errors For Maintenance READ/WRITE Request Transaction

Error	Interrupt	Status Bit Set	Error Response	Comments
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	
The Logical/Transport Layer Address Capture Command and Status Register described on **page 16-207** 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 16-14. 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 16.2.9.4.2**, *Window Boundary Crossing Errors,* on page 16-56.	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 16-14. 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 16-15. 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 16.2.9.4.2**, *Window Boundary Crossing Errors,* on page 16-56.	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 16-15. 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 16-16. 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 16-16. 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 16-17. Hardware Errors For IO/GSM 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
Critical Request Flow CRF field is set when corresponding request's CRF field was clear	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
TransportType Received reserved TT for this ftype	Yes if LTLEECSR [TSE] is set	LTLEDCSR [TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR [TSE] is set	LTLEDCSR [TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR [ITTE] is set	LTLEDCSR [ITTE]	No	Same as first entry	RapidIO packet is dropped and ignored

Table 16-17. Hardware Errors For IO/GSM Response Transactions (Not Maintenance)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
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
Not UR IO read response does not correspond to an outstanding valid IO/GSM read request. IO write response does not correspond to an outstanding valid IO/GSM write request.	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Status Not UR IO transaction - Is not "Done" or "Error" GSM transaction IO_Read_Home Is not "Done - Data-Only", "Done - Done-Intervention", "Done", "Retry" or "Error". Flush_w_Data response is not "Done", "Retry" or "Error" Transaction type of "Response_with_data" and status is not done	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Status Not UR GSM Error response	Yes if LTLEECSR [GER] is set	LTLEDCSR [GER]	Yes if data is not received for this request.	Same as first entry except error capture is done from original request packet.	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 16-17. Hardware Errors For IO/GSM Response Transactions (Not Maintenance)

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 16-17. Hardware Errors For IO/GSM Response Transactions (Not Maintenance)

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 16-17. Hardware Errors For IO/GSM Response Transactions (Not Maintenance)

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 16-18. 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 16-18. 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 16-19. 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 16-20. 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 16-21. 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 16-21. 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 16-22. 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 16-22. 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 16-22**, 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 16-22**, 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 16-23. Hardware Errors For Type9 Transactions

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
TransportType Received reserved TT	Yes if LTLEECSR [TSE] is set	LTLEDCSR [TSE]	No	Using the incoming RapidIO packet, **for Small Transport Type**, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets 0's **For Large Transport type packets**. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24–31,LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped and ignored
Received TT Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR [TSE] is set	LTLEDCSR [TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR [ITTE] is set	LTLEDCSR [ITTE]	No	Same as first entry	RapidIO packet is dropped and ignored

Table 16-24. 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 16-25. 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 16-26. 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 16.2.9.4.2**, *Window Boundary Crossing Errors,* on page 16-56. 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 Enhanced Message Unit (eMSG) Communication

The RapidIO message manager (eMSG) supports a message passing programming model for inter-processor and inter-device communication. The most common use of the message passing model is in systems where a processing element is only allowed to access memory local to itself, and communication between processing elements is achieved through message passing and is address-independent. A data streaming protocol is also supported, which defines a mechanism for transporting an arbitrary protocol over a standard RapidIO interface between two end points. The eMSG is compliant with the Type9 *RapidIO Interconnect Specification Part 2: Message Passing Logical Specification Rev. 1.3*, and Type11 *RapidIO Interconnect Specification Part 10: Data Streaming Logical Specification Rev. 1.3a.*

16.3.1 Overview

The eMSG has a total of 64 inbound classification units supporting RapidIO Type8–11. The classification units are separated into eight units per block, where the units are numbered 0–7 and the blocks 0–7. The units are managed through a set of run-time registers. Segments of a properly classified transaction are reassembled and put onto a queue for processing by software. There are 6 outbound segmentation units supporting RapidIO Type5–6 and Type8–11. These segmentation units are fed by outbound transaction queues managed by software, as follows:

- A Type5 outbound NWrite unit enables a producer to send up to 64 Kbyte of addressable data across the interconnect fabric to a consumer. eMSG does not support receiving of Type5 packets. NWrite allows for sub double-word data transfers.
- A Type6 outbound streaming write unit enables a producer to send up to 64 Kbyte of addressable data across the interconnect fabric to a consumer. eMSG does not support receiving of Type6 streaming writes. Streaming writes must be a multiple of double-words and does not include transfer size information.
- A Type8 port-write is intended as an error reporting mechanism from an end point-less device to a control processor or other system host. The eMSG also supports generation of outbound port-writes for testability and debug. Inbound port-writes have dedicated hardware for guaranteed delivery.
- A Type9 outbound segmentation unit enables a producer to send a packetized data unit (PDU) across the interconnect fabric to a consumer’s reassembly hardware. The receiving reassembly hardware places the PDU descriptor onto an inbound message queue. A PDU may consist of multiple segments supporting a total size of 64 Kbyte. PDUs can be queued for transmission in the producer’s memory and the segmentation hardware then processes them sequentially. PDUs can also be queued in the consumer’s memory while software processes them sequentially. The depths of the queues at the producer and consumer are configurable by software. Basic stream management flow control is also supported in Type9.

Note: No more than one PDU from a given RapidIO flow should be segmented at a time.

- A Type10 outbound doorbell unit enables a producer to send a small amount of software-defined information across the interconnect fabric to a consumer's doorbell unit. It is the responsibility of the processor receiving the doorbell to determine the action to undertake. A small data payload of 2 bytes is used for doorbells.
- A Type11 outbound eMSG enables a producer to send a message across the interconnect fabric to a consumer's message hardware, called a mailbox. The receiving mailbox hardware places a message descriptor onto an inbound message queue. A message may consist of one to sixteen segments. Messages can be queued for transmission in the producer's memory and the message hardware will process them sequentially. Messages can also be queued in the consumer's memory while software processes them sequentially. The depths of the queues at the producer and consumer are configurable by software. A multicast function allows single segment messages to be sent to multiple consumers.

Figure 16-14 shows a block diagram of the eMSG unit.

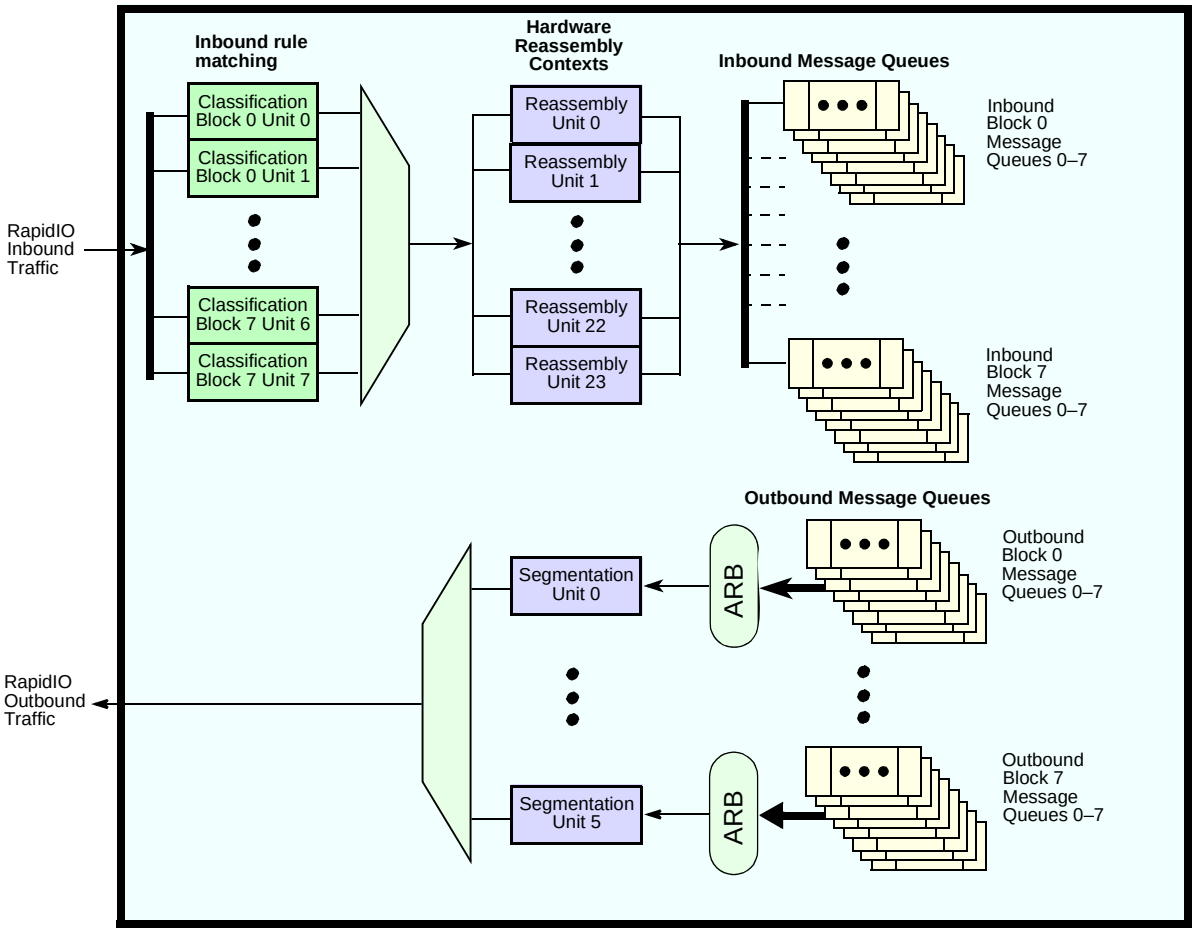


Figure 16-14. eMSG Block Diagram

16.3.2 Modes of Operation

The eMSG has two modes of operation: outbound and inbound.

16.3.2.1 Outbound Modes of Operation

The outbound units support Type5, Type6, and Type8–11 transactions. They also support Multicast Mode for Type11 in which a single segment message (256 bytes or less) is sent to multiple destinations.

16.3.2.2 Inbound Modes of Operation

The inbound classification units can be configured to handle Type8 through Type11 inbound traffic. There are a total of 64 inbound units (8 per block) that all work to classify incoming traffic based on header attributes to direct the transactions to a particular target message queue for later processing.

Classification mode is the inbound mode. In this mode, an inbound unit matches a programmable set of attributes to the inbound header, allowing steering of the transaction. A value/mask register pair is used to classify incoming packets based on header information. This process is referred to as classification and has the highest priority for inbound messages, doorbells, and packets, unless an active context is already open.

Figure 16-15 shows an example of traffic classification, where packets of Type8–11 are steered to the correct inbound unit for processing. If there is any aliasing between units, the lower numbered unit wins. A transaction that is not classified is subject to error response or is dropped.

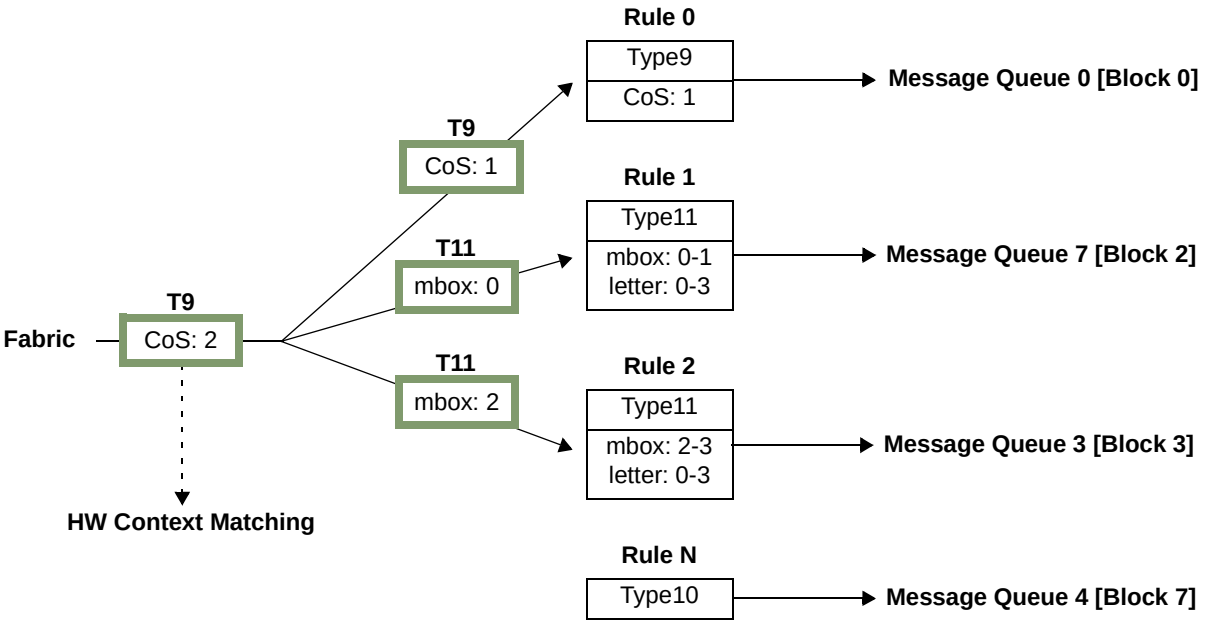


Figure 16-15. Inbound Traffic Classification

Note: The following sections describe the individual processing for each of the message types, including the outbound and inbound descriptor formats, and outbound and inbound message unit operation (where applicable).

16.3.3 Command Descriptor Format

The message unit relies on the use of the command descriptor (CD) format to enqueue/dequeue status/commands to and from the message queues.

16.3.3.1 Inbound Command Descriptor Format

The message unit enqueues Type8–11 command descriptors to the inbound message queues. Only command descriptor formats applicable to the message unit is shown in **Figure 16-16**. The status/command field (bits 31–0) is written by the message unit hardware during enqueue of inbound CD. Reserved fields must be cleared.

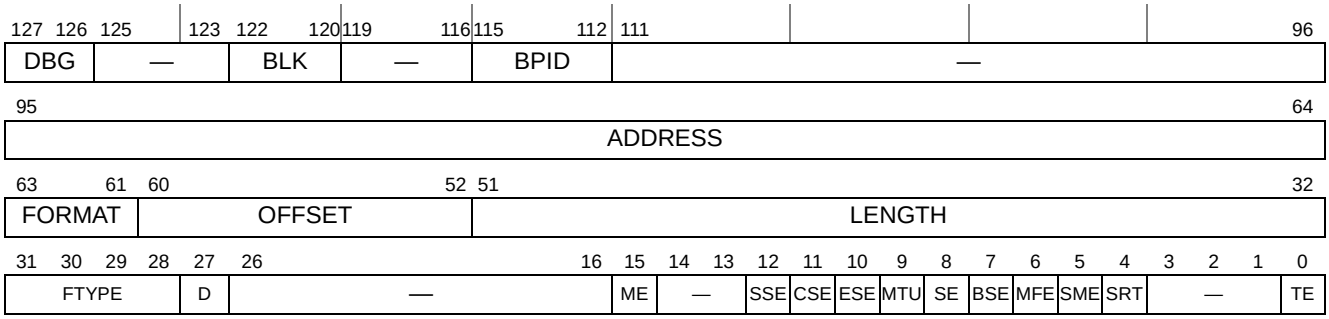


Figure 16-16. Inbound Command Descriptor Format

Table 16-27 describes the inbound command descriptor fields.

Table 16-27. Inbound Command Descriptor Field Descriptions

Bits	Name	Description
127–126	DBG	Debug
125–123	—	Reserved
122–120	BLK	Block number Added by hardware to determine the completion queue block for outbound and the enqueue block for inbound. Non-DPAA only
119–116	—	Reserved
115–112	BPID	Buffer Pool ID Specifies the buffer pool identifier used for buffer allocation associated with the data pointed to by address, offset and length fields. Non-DPAA only
111–96	—	Reserved
95–64	ADDRESS	32-bit address. See **Section 16.3.13**, *Address Alignment Requirements*.
63–61	FORMAT	Format (bits 60–32) 000 Single buffer frame, 9-bit offset, 20-bit length 100 Scatter/Gather table, 9-bit offset, 20-bit length. Valid for RapidIO Type9 only. All other values reserved or unused.

Table 16-27. Inbound Command Descriptor Field Descriptions (Continued)

Bits	Name	Description
60–52	OFFSET	Byte offset from starting address where information is stored. See **Section 16.3.13**, *Address Alignment Requirements*.
51–32	LENGTH	Total length of the data payload, excluding any scatter/gather table overhead. A buffer length of zero signifies that no data is valid and the address should not be accessed. For inbound transactions, the length indicates the total byte count written to memory, which may be less than expected if an error was encountered during reassembly.
Status and Command Fields This field is provided to allow the message unit to communicate “out of band” information to the consumer of those command descriptors.		
31–28	FTYPE	RapidIO type 1000 Type8 1001 Type9 1010 Type10 1011 Type11 All other values reserved.
27	D	Direction 0 Outbound 1 Inbound
26–16	—	Reserved
15	ME	Multiple errors Multiple errors of the same kind are reported. • Continuation segment error (CSE) • MTU size error (MTU) • Message format error (MFE)
14–13	—	Reserved
12	SSE	Single/Start segment error A start or single segment was received for an already opened context. A new PDU context is opened and the old PDU context is considered defective. Inbound Type9 only
11	CSE	Continuation segment error A continuation segment was received without a previously opened context. PDU context is considered defective. Inbound Type9 only
10	ESE	End segment error An end segment was received without a previously opened context. PDU context is considered defective. Inbound Type9 only
9	MTU	MTU violation PDU is considered defective.Inbound Type9 only
8	SE	Source error PDU was aborted by source. Inbound Type9 only
7	BSE	Buffer size error Message unit was unable to store the transaction due to insufficient buffer pool size. Inbound Type8/10/11 only
6	MFE	Message format error Duplicate, out of range, inconsistent segment number or illegal RapidIO priority or segment size received. Inbound Type10/11 only
5	SME	Size mismatch error Inbound length or byte count mismatch. Inbound Type9 only

Table 16-27. Inbound Command Descriptor Field Descriptions (Continued)

Bits	Name	Description
4	SRT	Segment request time-out error Time between end of segment reassembly to receipt of next segment. The time-out threshold value used is the same as the RapidIO PRTOCCSR (see **Section 16.4.1.16**, *Port Link Time-Out Control Command and Status Register (PLTOCCSR)*).
3–1	—	Reserved
0	TE	Internal transaction error Write access to local memory failed and the BMan resources are depleted.

16.3.3.2 Outbound Command Descriptor Format

The message unit dequeues command descriptors (CDs) of Type5, Type6, and Type8–11 from the outbound message queues. Command descriptors are also used for conveying completion status for outbound transactions when they are enqueued. Only command descriptor formats applicable to the message unit are shown **Figure 16-17**. Part of the status/command field (bits 100–127) is written by the message unit hardware during completion enqueue of CD. Reserved fields must be cleared.

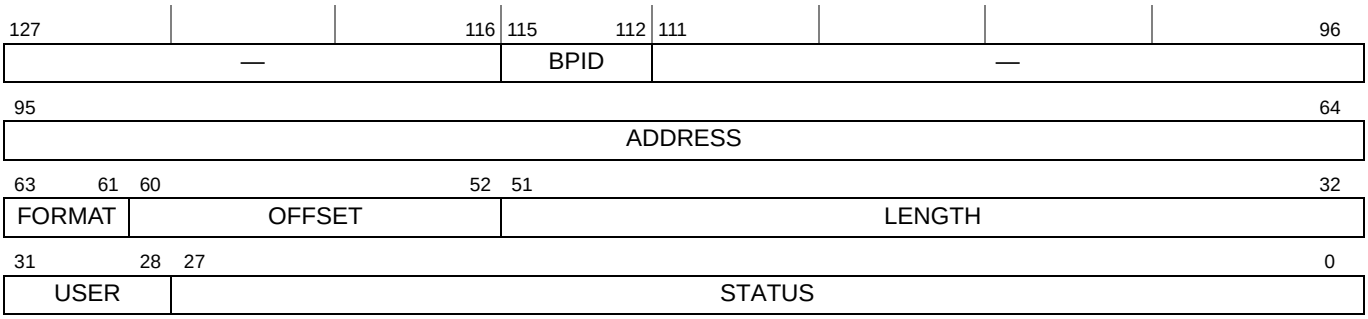


Figure 16-17. Outbound Command Descriptor Format

Table 16-28 describes the outbound command descriptor fields.

Table 16-28. Abound Command Descriptor Field Descriptions

Bits	Name	Description
127–116	—	Reserved
115–112	BPID	Buffer pool ID Specifies the buffer pool identifier used for buffer allocation associated with the data pointed to by address, offset and length fields.
111–96	—	Reserved
95–64	ADDRESS	32-bit address. See **Section 16.3.13**, *Address Alignment Requirements*.
63–61	FORMAT	Format (bits 60–32) 000 Single buffer frame, 9-bit offset, 20-bit length 100 Scatter/Gather table, 9-bit offset, 20-bit length. Valid for RapidIO Type5, Type6 and Type9 only All other values reserved or unused.

Table 16-28. Abound Command Descriptor Field Descriptions (Continued)

Bits	Name	Description
60–52	OFFSET	Byte offset from starting address where information is stored. See **Section 16.3.13**, *Address Alignment Requirements*.
51–32	LENGTH	Total length of the data payload, excluding any scatter/gather table overhead. A buffer length of zero signifies that no data is valid and the address should not be accessed.
31–28	USER	User defined field. This field is optional by the producer and is returned as-is during completion
Status Fields These field are provided to allow the message unit to communicate “out of band” information to the consumer during completion command descriptors.		
27	D	Direction 0 Outbound 1 Inbound
26–16	—	Reserved
15	ME	Multiple errors. Multiple errors of the same kind reported. • Error response received (ERR) • Multicast error response received (MCE)
14	—	Reserved
13	DE	Descriptor error A programming error was encountered when reading the scatter/gather table or message descriptor. Outbound only
12–5	—	Reserved
4	SRT	Segment response time-out error Time from segment transmitted by SRIO to logical response received by segmentation unit. The time-out threshold value used is the same as the RapidIO PRTOCCSR (see **Section 16.4.1.17**, *Port Response Time-Out Control Command and Status Register (PRTOCCSR)*)
3	MCE	Multicast error Failed to deliver message to one or more devices. The cause of error is indicated by one or more of the bits ERR/SRT/RTE/TE set. Outbound Type11 only
2	ERR	Error response received. Outbound Type10/11 only
1	RTE	Retry threshold exceeded. Outbound Type10/11 only
0	TE	Internal transaction error Data payload or scatter/gather table read access to local memory failed. **Note**: If a transaction error is encountered while reading the message descriptor, only global the transaction error is reported (MUMEDR[OTE]).

16.3.3.3 Outbound Completion Queues

Completion status information for outbound Type5, Type6, and Type8–11 transactions can be reported back to an outbound completion queue if the CS/ES bit is set in the message descriptor, unless a transaction error occurred while fetching the message descriptor. The status field, bits 100–127, carried within the command descriptor, is updated after segmentation completes. If buffer release (BR = 1) is performed successfully, the effective length field indicates zero. It is the responsibility of the outbound completion queue consumer to deallocate buffers as needed when the command descriptor length field is non-zero. If transaction or descriptor error status has

been reported, buffer pointers may have been partially released. There is one completion queue for each set of eight outbound queues.

Note: Completion order is preserved within the outbound completion queue, although transactions from the same outbound message queue may execute out of order, as dictated by the outbound ordering rules.

16.3.4 Scatter/Gather Tables

RapidIO Type5–6 and Type9 transactions may use the 16-byte scatter/gather table format for data management. The command descriptor has a format field indicating if the address/offset fields are pointing to a direct data memory location or another scatter/gather table. The scatter/gather table must be aligned to a 16-byte boundary. The definition of a scatter/gather table entry is described in **Figure 16-18**. Each scatter/gather table contains one or more entries, and may be extended to include additional tables. Each scatter/gather entry points to a data buffer or chains to another scatter/gather table as required.

Note: Do not use the scatter/gather table format for Type5, Type6, or Type9 messages during outbound segmentation. Using this format can result in segmentation unit counters being corrupted by a descriptor error (DE).

Note: Each scatter/gather entry has a 32-bit address field and a 30-bit buffer length field. The E (extension) bit determines if the address is pointing to a data buffer (E = 0) or to a new scatter/gather table (E = 1). The F (final) bit determines when the last entry of the table has been reached. Reserved fields must be cleared.

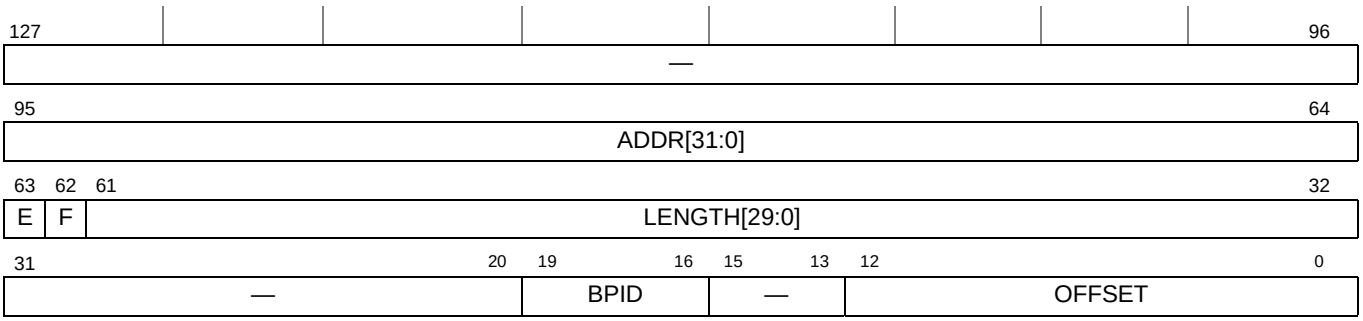


Figure 16-18. Scatter/Gather Table Entry Format

Table 16-29 describes the scatter/gather table entry fields.

Table 16-29. Scatter/Gather Table Entry Field Descriptions

Bits	Name	Description
127–96	—	Reserved
95–64	ADDRESS	32-bit byte address to data buffer or discontinued scatter/gather table entry, depending on E bit setting.

Table 16-29. Scatter/Gather Table Entry Field Descriptions (Continued)

Bits	Name	Description
63	E	Extension. 0 Address references a data buffer. 1 Address references a scatter/gather table. The extension bit has precedence over the final bit. If both are set, the final bit is ignored.
62	F	Final entry. 0 Not last table entry 1 Indicates final table entry This bit is ignored if the extension bit is set.
61–32	LENGTH	Effective length of the data buffer when E = 0, ignored when E = 1.
31–20	—	Reserved
19–16	BPID	Buffer pool ID Specifies the buffer pool identifier used for buffer allocation associated with the data pointed to by address, offset and length fields
15–13	—	Reserved
12–0	OFFSET	Byte offset from ADDRESS where data payload or scatter/gather table resides

A buffer length of zero is not allowed when the extension bit is deasserted and results in an undefined behavior. The length is ignored if the extension bit is set. Only the last scatter/gather segment may chain by setting the extension bit it does not have to chain. The offset value may be non-zero to indicate significant data is stored there.

The overall data payload length as defined by the command descriptor is also used to determine the end of a scatter/gather table. If the length has been reached, regardless of the final bit setting, the last data segment has been reached. The following occurs when the length has been reached, but the final bit is not set, and a buffer release was requested (BR=1):

- Any subsequent data buffer pointer(s) are not released
- Current and subsequent S/G table pointer(s) are not released

If software requires all pointers to be released, the final bit must be set for the last data entry as determined by the length field.

Figure 16-19 shows the format for scatter/gather chaining. If one scatter/gather table is insufficient for storing data, tables can be chained together until the desired number of entries are achieved.

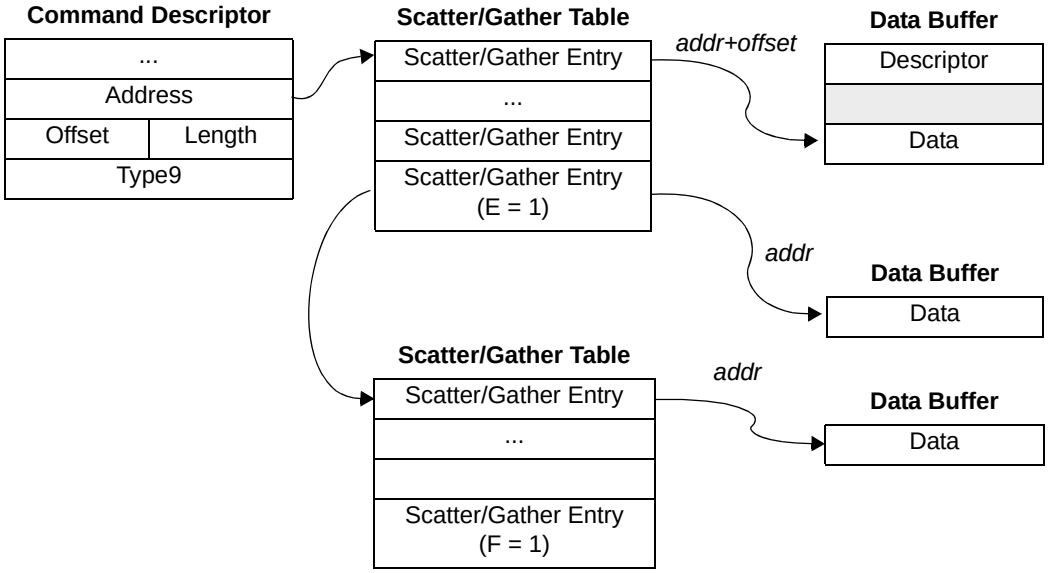


Figure 16-19. Type9 Scatter/Gather Table Chaining

16.3.5 Type5 NWrite Unit Functional Description

This section describes the Type5 NWrite descriptor formats. Type5 NWrites are only supported outbound.

16.3.5.1 Type5 Outbound NWrite Descriptor Format

The Type5 outbound descriptor contains information for the message unit to send an NWrite transaction. The message unit receives the outbound NWrite descriptor by dequeueing a command descriptor from an outbound message queue. The message descriptor points to a data buffer containing the descriptor and NWrite data payload.

Figure 16-20 shows the format of an outbound NWrite descriptor. Reserved fields must be cleared.

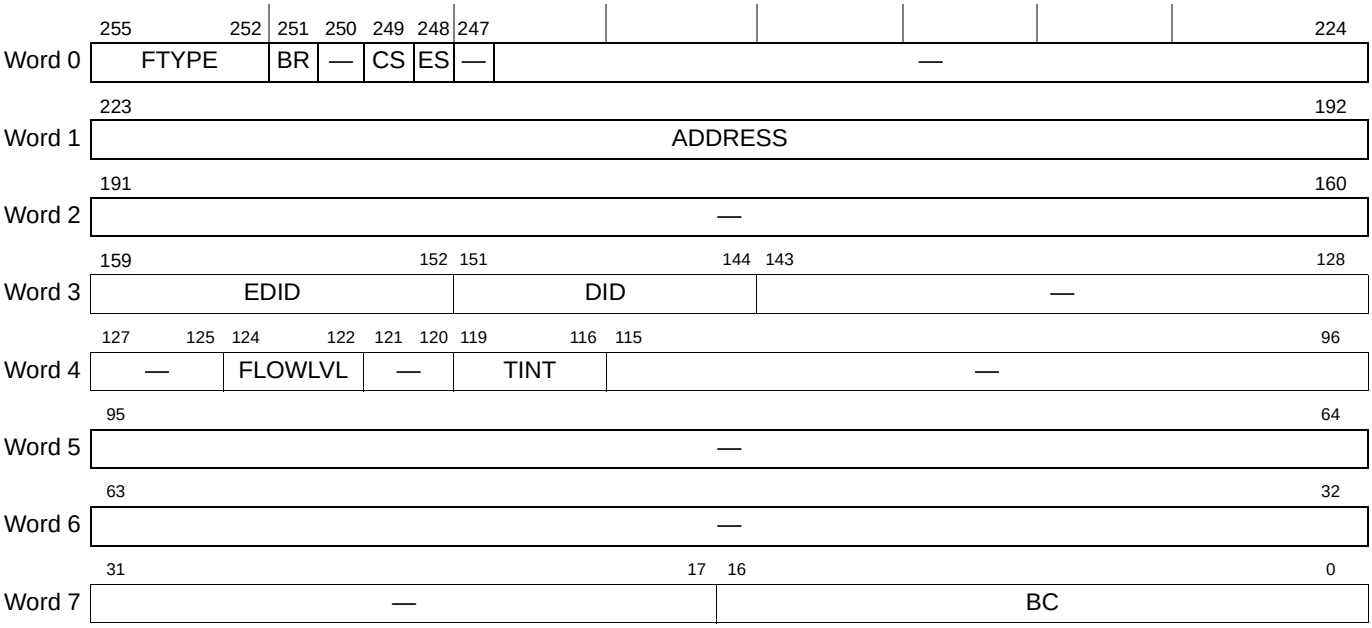


Figure 16-20. Type5 Outbound NWrite Descriptor

Table 16-30 describes the Type5 outbound NWrite descriptor fields.

Table 16-30. Type5 Outbound NWrite Descriptor Field Descriptions

Bits	Name	Description
Word 0—Header		
255–252	FTYPE	Descriptor type select 0101 Type5 All other values reserved.
251	BR	Buffer release enable 0 No hardware buffer releases performed. 1 Hardware buffer releases performed. If TE/DE error status was reported, not all buffers may have been released successfully.
250	—	Reserved
249	CS	Completion status 0 Status message queue is not updated when NWrite completes normally. 1 Status message queue is updated when NWrite completes normally.
248	ES	Error status 0 Status message queue is not updated when NWrite aborted due to error. 1 Status message queue is updated when NWrite aborted due to error.
247–224	—	Reserved
Word 1—Address		
223–192	ADDRESS	Byte aligned address of write transaction This address represents the RapidIO Type5 logical address without any address translation.
Word 2—Reserved		
191–160	—	Reserved
Word 3—Destination Port		

Table 16-30. Type5 Outbound NWrite Descriptor Field Descriptions (Continued)

Bits	Name	Description
159–152	EDID	Extended destination ID Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
151–144	DID	Destination ID Contains the destination ID field of the transaction (device ID of the target).
143–128	—	Reserved
Word 4—Destination Attributes		
127–125	—	Reserved
124–122	FLOWLVL	Transaction flow level 000 Lowest priority transaction request flow 001 Next highest priority transaction request flow ... 100 Next highest priority transaction request flow 101 Highest priority transaction request flow 110 Reserved 111 Reserved
121–120	—	Reserved
119–116	TINT	Target interface 0000 RapidIO Port 1 0001 RapidIO Port 2 All other values reserved.
115–96	—	Reserved
Word 5—Reserved		
95–64	—	Reserved
Word 6—Reserved		
63–32	—	Reserved
Word 7—Byte Count		
31–17	—	Reserved
16–0	BC	Byte count Contains the number of bytes for the write operation. 0x00001 1 byte 0x00002 2 bytes 0x00003 3 bytes ... 0x0FFFF 64 Kbytes –1 0x10000 64 Kbytes All other values are reserved.

Figure 16-21 shows the format for Type5 scatter/gather tables usage. The first entry of the scatter/gather table holds the 32-byte descriptor. The data offset is specified through the

scatter/gather entry offset field. Non-scatter/gather format is also supported for Type5 if the data payload plus the descriptor offset is less than or equal to 64 Kbytes.

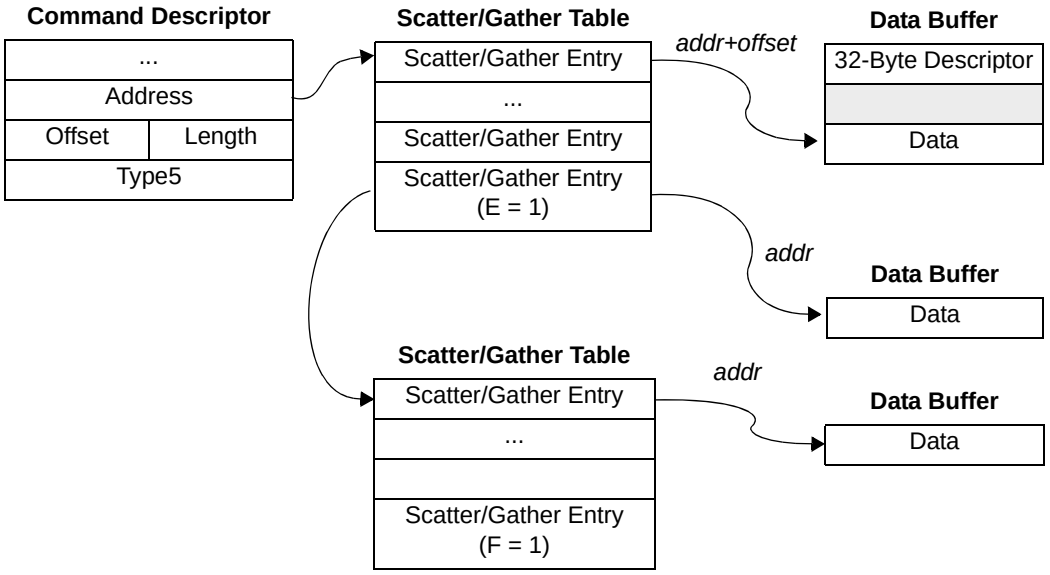


Figure 16-21. Outbound Type5 Scatter/Gather Table Format

16.3.5.2 Type5 Outbound NWrite Operation

RapidIO supports an NWrite type that contains a data payload from 1 to 64 Kbytes. The outbound message unit is responsible for sending NWrites enqueued by a producer or producers onto one of the outbound message queue owned by the message unit. Concurrent processing may occur between any number of message queues as long as the outbound ordering rules apply; see **Section 16.3.14.2**, *Outbound Ordering Rules*.

16.3.5.2.1 Work Scheduling

The message unit receives all work from the outbound message queues. A producer or producers enqueueue work onto the outbound message queues owned by the message unit. To maintain any ordering between transactions, a producer must enqueue transactions with dependencies onto the same outbound message queue. The outbound ordering rules are described in **Section 16.3.14.2**, *Outbound Ordering Rules*.

16.3.5.2.2 Adding NWrites to a Message Queue

A producer may add new NWrites to any message queue owned by the message unit. The producer needs to allocate buffer space for the NWrite descriptor as appropriate and enqueue the command descriptor. The message unit dequeues the command descriptor, determines the type, and proceeds to read the NWrite descriptor.

A producer should follow the following guidelines to add a new NWrite to a message queue:

- Create the 32-byte outbound NWrite descriptor as described in **Section 16.3.5.1**, *Type5 Outbound NWrite Descriptor Format*.
  - Set the descriptor type select field FTYPE = 4b0101 to indicate NWrite type.
  - Allocate buffer for the NWrite descriptor from one of the buffer pools available, preferably with the smallest buffer size (that is, 64 bytes).
  - For all other fields, see **Section 16.3.5.1**, *Type5 Outbound NWrite Descriptor Format*.
- Add the NWrite descriptor to a command descriptor and set FTYPE = 4b0101 to indicate the NWrite type.
- Enqueue the command descriptor onto the message queue.
- The message unit releases buffers used by the producer to hold the NWrite descriptor and data payload when the operation completes if the buffer release (BR) bit is set in the message descriptor.

16.3.5.2.3 NWrite Initialization

See **Section 16.3.17.6**, *Initializing Outbound Message Queues*.

16.3.5.2.4 Error Handling

The message unit can report completion status and error information after completing the command descriptor by updating the command descriptor status field.

The following steps should be performed to enable this mode.

- Software should observe the status field of the command descriptor after it has been written to the completion queue to determine success.

Table 16-31 indicates the possible error conditions and the outcome when detected. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected, no additional error checking beyond the current error level is performed.

Table 16-31. Outbound Segmentation Hardware Errors

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Comments
NWrite Request	Command descriptor effective length less than message descriptor payload byte count	1	No	Descriptor error in command descriptor enqueued	No	No buffers released regardless of BR setting
NWrite Request	Invalid FTYPE	1	No	Descriptor error in command descriptor enqueued	No	No buffers released regardless of BR setting

Table 16-31. Outbound Segmentation Hardware Errors (Continued)

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Comments
NWrite Request	Invalid target interface specified	1	No	Descriptor error in command descriptor enqueued	No	No buffers released regardless of BR setting
NWrite Request	End of scatter/gather table reached before message descriptor payload byte count accounted for	2	No	Descriptor error in command descriptor enqueued	Partially	—
NWrite Request	Memory transaction error	2	Message unit error if MUIER[TE] set	Transaction error in command descriptor enqueued. MUEDR[TE]. Command descriptor captured in MUECCDR0–3.	No	NWrite aborted. No additional buffers released

16.3.5.2.4.1 Descriptor Error

When a descriptor error is detected by the message unit, the following occurs.

- Completion message queue
  - The segmentation process is halted and the descriptor error bit (DE) in the command descriptor is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status bit (ES) is set.

16.3.5.2.4.2 Transaction Errors

When a transaction error occurs during a local memory data read by the message unit, the following occurs.

- Segments that encounter a transaction error are not sent because the packet data is not available.
- Memory reads that were already generated before the transaction error occurred are also not transferred.
- Command descriptor is captured in MUMECCDR0–3.
- Global message unit error detect register MUMEDR[OTE] is set.
- Completion message queue
  - After the segmentation operation completes, the internal transaction error bit in the TE field in the command descriptor is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status bit (ES) is set.

16.3.6 Type6 Streaming Write Functional Description

This section describes the different Typ6 SWrite descriptor formats. Type6 SWrites are only supported outbound.

16.3.6.1 Type6 Outbound SWrite Descriptor Format

The Type6 outbound descriptor contains information for the message unit to send an SWrite transaction. The message unit receives the outbound SWrite descriptor by dequeueing a command descriptor from an outbound message queue. The message descriptor points to a data buffer containing the descriptor and SWrite data payload.

Figure 16-22 shows the format of an outbound SWrite descriptor. Reserved fields must be cleared.

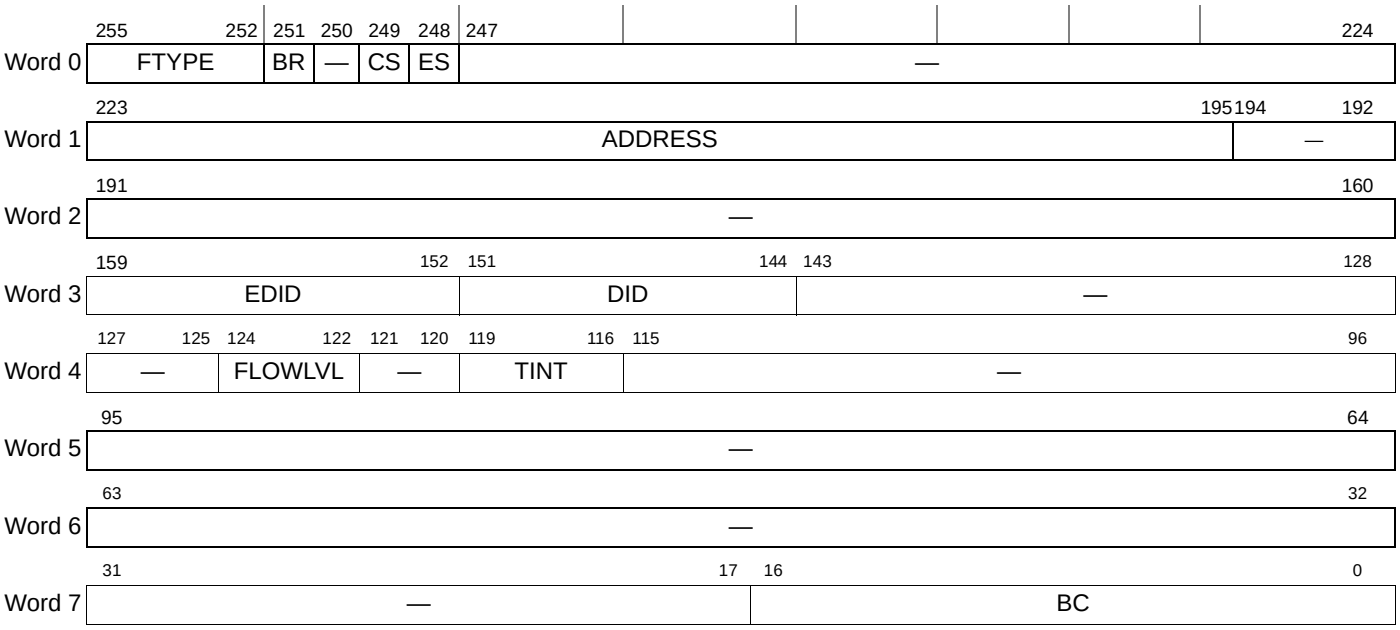


Figure 16-22. Type6 Outbound SWrite Descriptor

Table 16-32 describes the Type6 outbound SWrite descriptor fields.

Table 16-32. Type6 Outbound SWrite Descriptor Field Descriptions

Bits	Name	Description
Word 0—Header		
255–252	FTYPE	Descriptor type select 0110 Type6 All other values are reserved.
251	BR	Buffer release enable 0 No hardware buffer releases performed. 1 Hardware buffer releases performed. If TE/DE error status was reported, not all buffers may have been released successfully.

Table 16-32. Type6 Outbound SWrite Descriptor Field Descriptions (Continued)

Bits	Name	Description
250	—	Reserved
249	CS	Completion status 0 Status message queue is not updated when SWrite completes normally. 1 Status message queue is updated when SWrite completes normally.
248	ES	Error status 0 Status message queue is not updated when SWrite aborted due to error. 1 Status message queue is updated when SWrite aborted due to error.
247–224	—	Reserved
Word 1—Address		
223–195	ADDRESS	Double-word aligned address of write transaction This address represents the RapidIO Type6 logical address without any address translation.
194–192	—	Reserved
Word 2—Reserved		
191–160	—	Reserved
Word 3—Destination Port		
159–152	EDID	Extended destination ID Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
151–144	DID	Destination ID Contains the destination ID field of the transaction (Device ID of the target).
143–128	—	Reserved
Word 4—Destination Attributes		
127–125	—	Reserved
124–122	FLOWLVL	Transaction flow level 000 Lowest priority transaction request flow 001 Next highest priority transaction request flow ... 100 Next highest priority transaction request flow 101 Highest priority transaction request flow 110 Reserved 111 Reserved
121–120	—	Reserved
119–116	TINT	Target interface 0000 RapidIO Port 1 0001 RapidIO Port 2 All other values reserved.
115–96	—	Reserved
Word 5—Reserved		
95–64	—	Reserved
Word 6—Reserved		
63–32	—	Reserved

Table 16-32. Type6 Outbound SWrite Descriptor Field Descriptions (Continued)

Bits	Name	Description
Word 7—Byte Count		
31–17	—	Reserved
16–0	BC	Byte count Contains the number of bytes for the write operation. Only multiples of 8 bytes are allowed. 0x00008 8 bytes 0x00010 16 bytes 0x00018 24 bytes ... 0x0FFF8 64 Kbytes – 8 0x10000 64 Kbytes All other values reserved.

Figure 16-23 shows the format for Type6 scatter/gather tables usage. The first entry of the scatter/gather table holds the 32-byte descriptor. The data offset is specified through the scatter/gather entry offset field. Non-scatter/gather format is also supported for Type6 if the data payload plus the descriptor offset is less than or equal to 64 Kbytes.

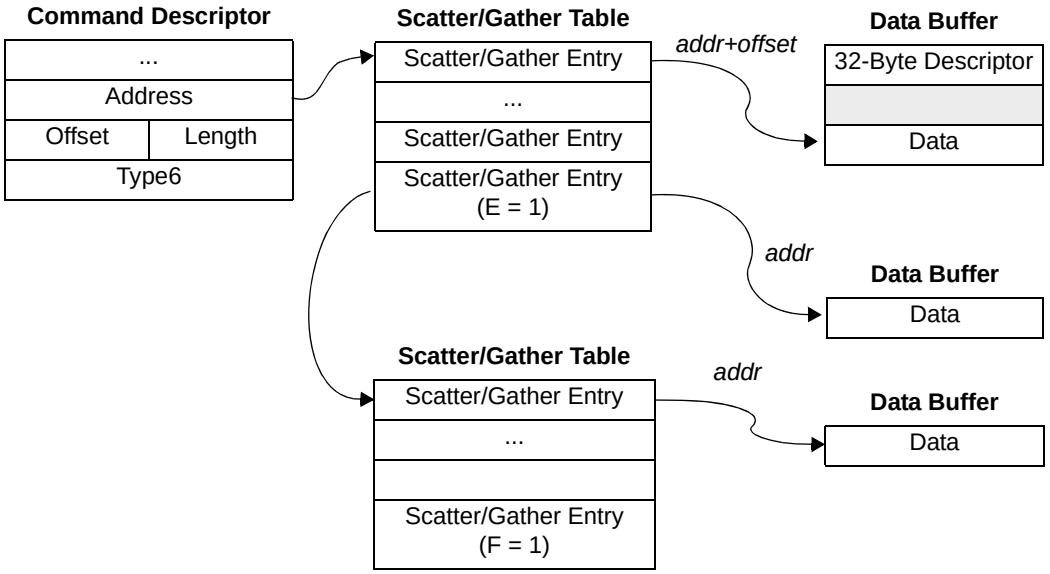


Figure 16-23. Outbound Type6 Scatter/Gather Table Format

16.3.6.2 Type6 Outbound SWrite Operation

RapidIO supports an SWrite type that contains a data payload from 1 to 64 Kbytes. The segmentation units are responsible for sending messages enqueued by a producer or producers from one of the outbound frame queues owned by the message unit. Concurrent processing may also occur between any number of message queues as long as the outbound ordering rules apply; see **Section 16.3.14.2**, *Outbound Ordering Rules*.

16.3.6.2.1 Work Scheduling

The message unit receives all work from the outbound message queues. A producer or producers enqueueue work onto the outbound message queues owned by the message unit. To maintain any ordering between transactions, a producer must enqueue transactions with dependencies onto the same outbound message queue. The outbound ordering rules are described in **Section 16.3.14.2**, *Outbound Ordering Rules*.

16.3.6.2.2 Adding NWrites To A Message Queue

A producer may add new SWrites to any message queue owned by the message unit. The producer needs to allocate buffer space for the SWrite descriptor as appropriate and enqueue the command descriptor. The message unit dequeues the command descriptor, determines the type, and proceeds to read the SWrite descriptor.

A producer should follow the following guidelines to add a new SWrite to a message queue:

- Create the 32-byte outbound SWrite descriptor as described in **Section 16.3.6.1**, *Type6 Outbound SWrite Descriptor Format*.
  - Set the descriptor type select field FTYPE = 4b0110 to indicate SWrite type.
  - Allocate buffer for the SWrite descriptor from one of the buffer pools available, preferably with the smallest buffer size (that is, 64 bytes).
  - For all other fields, see **Section 16.3.6.1**, *Type6 Outbound SWrite Descriptor Format*.
- Add the SWrite descriptor to a command descriptor and set FTYPE=4b0110 to indicate SWrite type.
- Enqueue the command descriptor onto the message queue.
- The message unit releases buffers used by the producer to hold the SWrite descriptor and data payload when the operation completes, if the buffer release bit (BR) is set in the message descriptor.

16.3.6.2.3 SWrite Initialization

See **Section 16.3.17.6**, *Initializing Outbound Message Queues*.

16.3.6.2.4 Error Handling

The message unit can report completion status and error information after completing the command descriptor by updating the command descriptor status field.

The following steps should be performed to enable this mode.

- Set the message descriptor completion (CS) or error (ES) status bit.
- Software should observe the status field of the command descriptor after it has been written to the completion queue to determine success.

Table 16-33 indicates the possible error conditions and the outcome when detected. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected, no additional error checking beyond the current error level is performed.

Table 16-33. Outbound Segmentation Hardware Errors

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Comments
SWrite Request	Command descriptor effective length less than message descriptor payload byte count	1	No	Descriptor error in command descriptor enqueued	No	No buffers released regardless of BR setting
SWrite Request	Invalid FTYPE	1	No	Descriptor error in command descriptor enqueued	No	No buffers released regardless of BR setting
SWrite Request	Invalid target interface specified	1	No	Descriptor error in command descriptor enqueued	No	No buffers released regardless of BR setting
SWrite Request	End of scatter/gather table reached before message descriptor payload byte count accounted for	2	No	Descriptor error in command descriptor enqueued	Partially	—
SWrite Request	Memory transaction error	2	Message unit error if MUIER[TE] is set	Transaction error in command descriptor enqueued. MUMEDR[TE]. Command descriptor captured in MUECCDR0–3	No	SWrite aborted. No additional buffers released

16.3.6.2.4.1 Descriptor Error

When a descriptor error is detected by the message unit the following occurs:

- Completion message queue
  - The segmentation process is halted and the descriptor error bit DE in the command descriptor is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

16.3.6.2.4.2 Transaction Errors

When an transaction error occurs during a local memory data read by the message unit the following occurs.

- Segments that encounter a transaction error are not sent because the packet data is not available.
- Memory reads that were already generated before the transaction error occurred are also not transferred.
- Command descriptor is captured in MUECCDR0–3.
- Global message unit error detect register MUEDR[OTE] is set.
- Completion message queue
  - After the segmentation operation completes, the internal transaction error bit in the TE field, in the command descriptor, is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

16.3.7 Type8 Port-Write Functional Description

Type8 port-writes are supported for both inbound and outbound, although port-writes are normally not generated by an end-point.

16.3.7.1 Type8 Outbound Port-Write Descriptor Format

The Type8 outbound descriptor contains information for the message unit to send a maintenance port-write. The message unit receives the outbound maintenance descriptor by dequeueing a command descriptor from an outbound message queue. The message descriptor points to a data buffer containing the descriptor and port-write data payload.

Figure 16-24 shows the format of an outbound port-write descriptor. Reserved fields must be cleared.

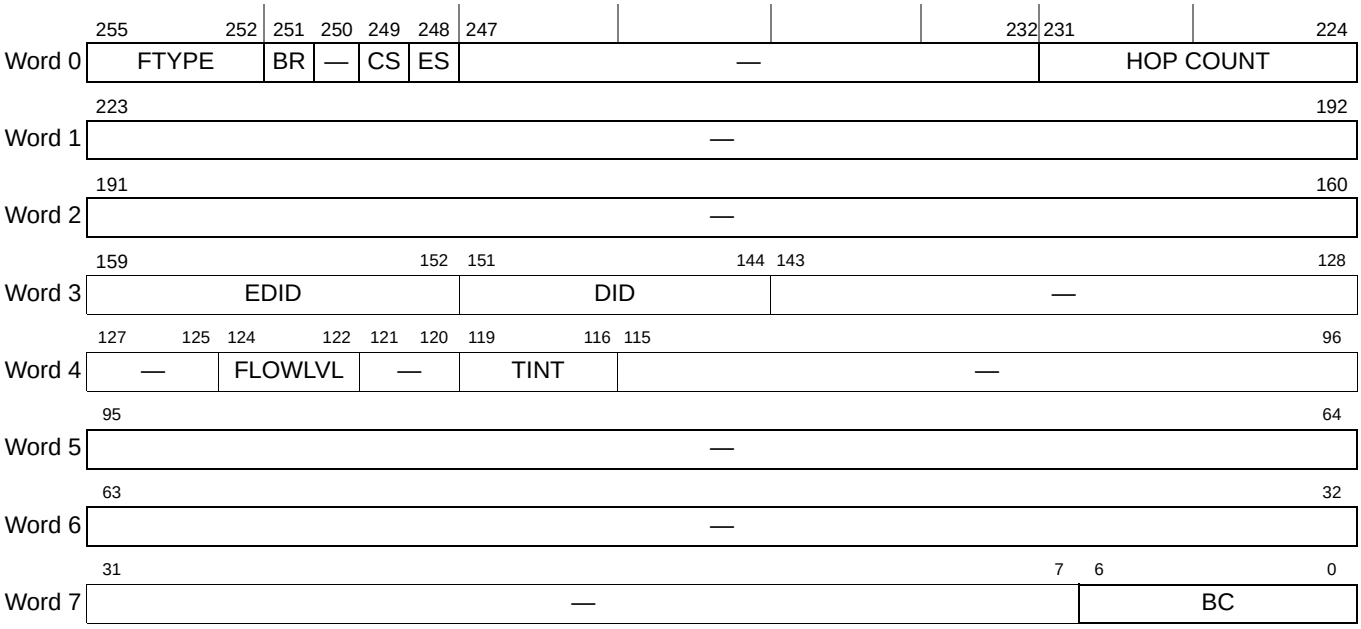


Figure 16-24. Type8 Outbound Port-Write Descriptor

Table 16-34 describes the Type8 outbound port-write descriptor fields.

Table 16-34. Type8 Outbound Port-Write Descriptor Field Descriptions

Bits	Name	Description
Word 0—Header		
255–252	FTYPE	Descriptor type select. 1000 Type8 All other values reserved.
251	BR	Buffer release enable. 0 No hardware buffer releases performed. 1 Hardware buffer releases performed. If TE/DE error status was reported, not all buffers may have been released successfully.
250	—	Reserved
249	CS	Completion status. 0 Status message queue is not updated when port-write completes normally. 1 Status message queue is updated when port-write completes normally.
248	ES	Error status 0 Status message queue is not updated when port-write aborted due to error. 1 Status message queue is updated when port-write aborted due to error.
247–232	—	Reserved
231–224	HOP COUNT	Hop count field in RapidIO port-write packet.
Word 1—Reserved		
223–192	—	Reserved
Word 2—Reserved		
191–160	—	Reserved
Word 3—Destination Port		
159–152	EDID	Extended destination ID**.** Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
151–144	DID	Destination ID**.** Contains the destination ID field of the transaction (Device ID of the target).
143–128	—	Reserved
Word 4—Destination Attributes		
127–125	—	Reserved
124–122	FLOWLVL	Transaction flow level. 000 Lowest priority transaction request flow 001 Next highest priority transaction request flow ... 110 Next highest priority transaction request flow 111 Highest priority transaction request flow
121–120	—	Reserved
119–116	TINT	Target interface. 0000 RapidIO Port 1 0001 RapidIO Port 2 All other values reserved.
115–96	—	Reserved

Table 16-34. Type8 Outbound Port-Write Descriptor Field Descriptions (Continued)

Bits	Name	Description
Word 5—Reserved		
95–64	—	Reserved
Word 6—Reserved		
63–32	—	Reserved
Word 7—Byte Count		
63–7	—	Reserved
6–0	BC	Byte count. Contains the number of bytes for the port-write operation. Only multiples of 8 bytes are valid for transactions larger than 8 bytes. 000 0100 4 bytes 000 1000 8 bytes 001 0000 16 bytes 001 1000 24 bytes ... 011 1000 56 bytes 100 0000 64 bytes For sub-8 byte transfers, only RapidIO byte mask of 8b1111_0000 is supported. All other values reserved.

Figure 16-25 illustrates an example of a command descriptor referencing the data buffer holding the Type8 port-write descriptor and data. The starting location of the port-write data payload is calculated by adding the data offset to the command descriptor address.

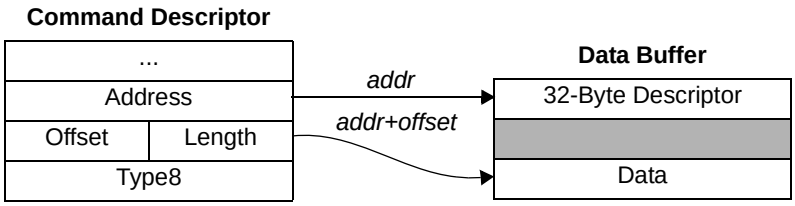


Figure 16-25. Command Descriptor Referencing Outbound Type8 Port-Write Descriptor

16.3.7.2 Type8 Inbound Port-Write Descriptor Format

This section defines the format of the inbound port-write command descriptor. No port-write message descriptor is written, because the payload contains all the necessary information. The command descriptor is then enqueued onto an inbound message queue processed by the consumer.

Figure 16-26 illustrates an example of a command descriptor referencing the data buffer holding the Type8 port-write data payload. The starting location of the port-write data payload is calculated by adding the data offset to the command descriptor address. The data offset is programmable through IB*m*T8C*n*DOR.

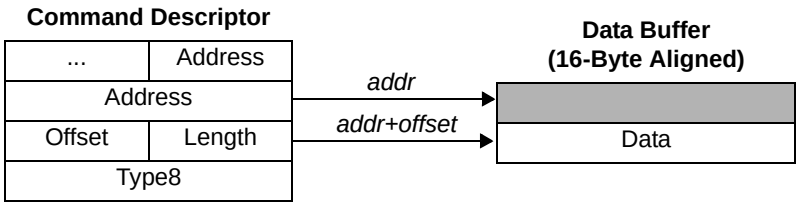


Figure 16-26. Command Descriptor Referencing Inbound Type8 Port-Write Data Payload

16.3.7.3 Type8 Outbound Port-Write Operation

RapidIO supports a maintenance port-write type that contains a data payload from 4 to 64 bytes. The segmentation units are responsible for sending messages enqueued by a producer or producers from one of the outbound frame queues owned by the message unit. Concurrent processing may also occur between any number of message queues as long as the outbound ordering rules apply, see **Section 16.3.14.2**, *Outbound Ordering Rules*.

16.3.7.3.1 Work Scheduling

The message unit receives all work from the outbound message frame queues. A producer or producers enqeueue work onto the outbound message queues owned by the message unit. To maintain any ordering between transactions, a producer must enqueue transactions with dependencies onto the same outbound message queue. The outbound ordering rules are described in **Section 16.3.14.2**, *Outbound Ordering Rules*.

16.3.7.3.2 Adding Port-Writes To A Message Queue

A producer may add new port-writes to any message queue owned by the message unit. The producer needs to allocate buffer space for the port-write descriptor as appropriate and enqueue the command descriptor. The message unit dequeues the command descriptor, determine the type, and proceeds to read the port-write descriptor.

A producer should follow the guidelines below to add a new port-write to a message queue:

- Create the 32-byte outbound port-write descriptor as described in **Section 16.3.7.1**, *Type8 Outbound Port-Write Descriptor Format*.
  - Set the descriptor type select field FTYPE=4b1000 to indicate port-write type.
  - Allocate buffer for the port-write descriptor from one of the buffer pools available, preferably with the smallest buffer size (that is, 64 bytes).
  - For all other fields, see **Section 16.3.7.1**, *Type8 Outbound Port-Write Descriptor Format*.
- Add the Portwrite descriptor to a command descriptor and set FTYPE=0b1000 to indicate port-write type.
- Enqueue the command descriptor onto the message queue.

- The message unit releases buffers used by the producer to hold the port-write descriptor and data payload when the operation completes if the buffer release bit (BR) bit was set in the message descriptor.

16.3.7.3.3 Port-Write Initialization

See **Section 16.3.17.6**, *Initializing Outbound Message Queues*.

16.3.7.3.4 Error Handling

The message unit can report completion status and error information after completing the command descriptor by updating the command descriptor status field.

The following steps should be performed to enable this mode.

- Set the message descriptor completion (CS) or error (ES) status bit.
- Software should observe the status field of the command descriptor after it has been written to the completion queue to determine success.

Table 16-35 indicates the possible error conditions and the outcome when detected. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected, no additional error checking beyond the current error level is performed.

Table 16-35. Outbound Port-Write Hardware Errors

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Port-Write Sent	Comments
Port-Write Request	Command descriptor effective length less than message descriptor payload byte count.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Port-Write Request	Unsupported byte count. Only 4, 8, 16, 32 and 64 bytes allowed.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Port-Write Request	Invalid FTYPE.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.

Table 16-35. Outbound Port-Write Hardware Errors (Continued)

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Port-Write Sent	Comments
Port-Write Request	Invalid target interface specified.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Port-Write Request	Memory transaction error.	2	Message unit error if MUIER[OTE] set.	Transaction error in command descriptor enqueued. MUEDR[OTE]. Command descriptor captured in MUECCDR0–3	No	Port-write aborted. No additional buffers released.

16.3.7.3.4.1 Descriptor Error

When a descriptor error is detected by the message unit the following occurs.

- Completion Message queue
  - The segmentation process is halted and the descriptor error bit DE in the command descriptor is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

16.3.7.3.4.2 Transaction Errors

When an transaction error is detected during a local memory data read by the message unit the following occurs.

- Port-writes that encounter a transaction error are not sent because the port-write data is not available.
- Memory reads that were already generated before the transaction error occurred are also not transferred.
- Command descriptor is captured in MUECCDR0–3.
- MUEDR[OTE] is set.
- Completion message queue
  - After the port-write operation completes the transaction error bit, TE, in the command descriptor is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.
- Buffer release (BR = 1)
  - No further buffers are released once a TE error is detected.

16.3.7.4 Type8 Inbound Port-Write Operation

The port-write is intended as an error reporting mechanism from an end point-less device to a control processor or other system host. The message unit is responsible for receiving RapidIO Type8 port-writes and placing then onto an inbound message queue. Software can specify the destination message queue for the port-write operation utilizing port-write classification, see **Section 16.3.17.2**, *Classification Initialization*. Utilizing the buffer pools, the message unit requests a single buffer to hold the port-write information. Software must provide the required sized buffer pool in IBmT8CnBPR for the message unit to store the entire port-write. The data offset is defined in IB*m*T8C*n*DOR. The message unit generates a frame size error if the inbound port-write cannot be stored in a single buffer due to insufficient buffer sizes available.

16.3.7.4.1 Inbound Port-Write Initialization

To configure the message unit for port-write operations, see **Section 16.3.17.5**, *Initializing Inbound Message Queues*.

16.3.7.4.2 Error Handling

The message unit reports errors, utilizing the inbound message queue, by setting the status field in the command descriptor before enqueueing the descriptor onto the designated inbound message queue. The consumer should read the status bits before processing the port-write to determine the appropriate action.

Table 16-36 indicates the possible error conditions and the outcome when detected. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected, no additional error checking beyond the current error level is performed.

Table 16-36. Inbound Port-Write Hardware Errors

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Local Memory Updated	Comments
Buffer size too small.	1	No	Bit BSE in command descriptor enqueued.	Yes	Port-write is invalid.
Memory transaction error.	3	Message unit error if MUIER[ITEIE] set.	Transaction error in command descriptor enqueued and message unit error detect register MUEDR[ITE]. Address captured in MUECAR.	Possibly	Port-write is invalid.

Table 16-36. Inbound Port-Write Hardware Errors (Continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Local Memory Updated	Comments
Buffer pool exhausted.	3	Message unit error if MUIER[BAEIE] set.	Transaction error in command descriptor enqueued and Message Unit Error Detect Register MUEDR[BAE].	No	Port-write is invalid.
Failure to enqueue the command descriptor onto the inbound message queue.	4	Message unit error if MUIER[IMERIE] set.	MUMEDR[IMER]. Message queue captured in MUECMQR.	Yes	—

16.3.7.4.2.1 Buffer Size Errors

When a buffer size error is detected by the message unit, the following occurs.

- Port-write is discarded.
- The message unit sets the buffer size error bit, BSE in the command descriptor.
- After the port-write operation completes, the command descriptor is enqueued onto the message queue.

16.3.7.4.2.2 Transaction Errors

When an transaction error occurs, due to insufficient buffer resources or local memory write error, by the message unit the following occurs.

- Port-write is defective.
- The message unit sets the transaction error bit, TE, in the command descriptor.
- After the doorbell operation completes, the command descriptor is enqueued onto the message queue.
- MUEDR[ITE] is set for memory access error.
- Address is captured in MUECAR for memory access error.
- MUEDR[BAE] is set if buffer resources depleted.

16.3.8 Type9 Data Streaming Functional Description

Type9 data streaming is supported both inbound and outbound.

16.3.8.1 Type9 Segmentation Descriptor Format

The Type9 outbound descriptor contains information for the message unit to send a PDU. The message unit receives the outbound packet descriptor by dequeueing a command descriptor from

the outbound message queue. The command descriptor points to a scatter/gather table where the first entry points to the outbound descriptor.

Figure 16-27 shows the format of an outbound PDU descriptor. Reserved fields must be cleared.

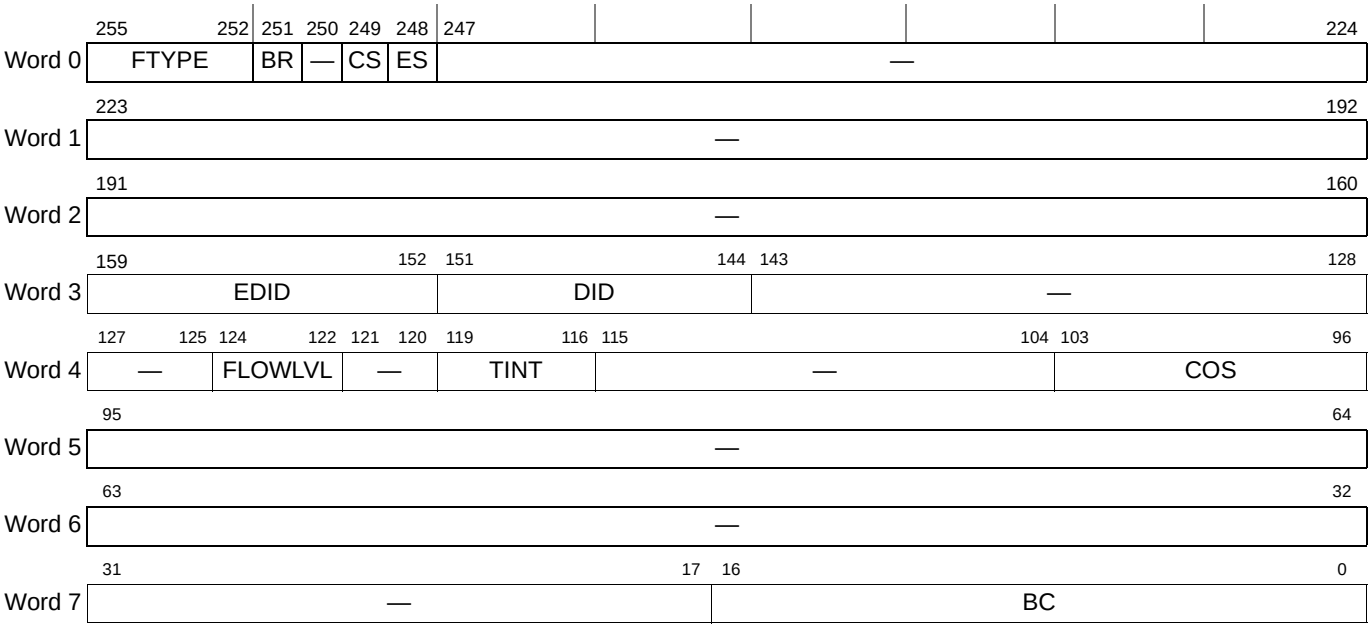


Figure 16-27. Type9 Outbound PDU Descriptor

Table 16-37 describes the Type9 outbound PDU descriptor fields.

Table 16-37. Type9 Outbound PDU Descriptor Field Descriptions

Bits	Name	Description
Word 0—Header		
255–252	FTYPE	Descriptor type select. 1001 Type9 All other values reserved.
251	BR	Buffer release enable. 0 No hardware buffer releases performed. 1 Hardware buffer releases performed. If TE/DE error status was reported, not all buffers may have been released successfully.
250	—	Reserved
249	CS	Completion status. 0 Status message queue is not updated when PDU completes normally. 1 Status message queue is updated when PDU completes normally.
248	ES	Error status. 0 Status message queue is not updated when PDU aborted due to error. 1 Status message queue is updated when PDU aborted due to error.
247–224	—	Reserved
Word 1—Reserved		
223–192	—	Reserved

Table 16-37. Type9 Outbound PDU Descriptor Field Descriptions (Continued)

Bits	Name	Description
Word 2—Reserved		
191–160	—	Reserved
Word 3—Destination Port		
159–152	EDID	Extended destination ID**.** Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
151–144	DID	Destination ID. Contains the destination ID field of the transaction (device ID of the target).
143–128	STREAMID	Value for the “streamID” field in the DATA STREAMING packet.
Word 4—Destination Attributes		
127–125	—	Reserved
124–122	FLOWLVL	Transaction flow level. 000 Lowest priority transaction request flow 001 Next highest priority transaction request flow ... 100 Next highest priority transaction request flow 101 Highest priority transaction request flow 110 Reserved 111 Reserved
121–120	—	Reserved
119–116	TINT	Target interface. 0000 RapidIO Port 1 0001 RapidIO Port 2 All other values reserved.
115–104	—	Reserved
103–96	COS	Value for the “COS” field in the DATA STREAMING packet.
Word 5–6		
95–32	—	Reserved
Word 7—Byte Count		
31–17	—	Reserved
16–0	BC	Byte count. Contains the number of bytes for this segmentation operation. The maximum segmentation operation size is 64 Kbytes and the minimum is 1 byte. 0x00001 1 byte 0x00002 2 bytes 0x00003 3 bytes ... 0x0FFFF 64 Kbytes-1 0x10000 64 Kbytes All other values reserved.

Figure 16-28 shows the format for Type9 scatter/gather tables usage. The first entry of the scatter/gather table holds the 32-byte descriptor. The data offset is specified through the

scatter/gather entry offset field. Non-scatter/gather format is also supported for Type9 if the data payload plus the descriptor offset is less than or equal to 64 Kbytes.

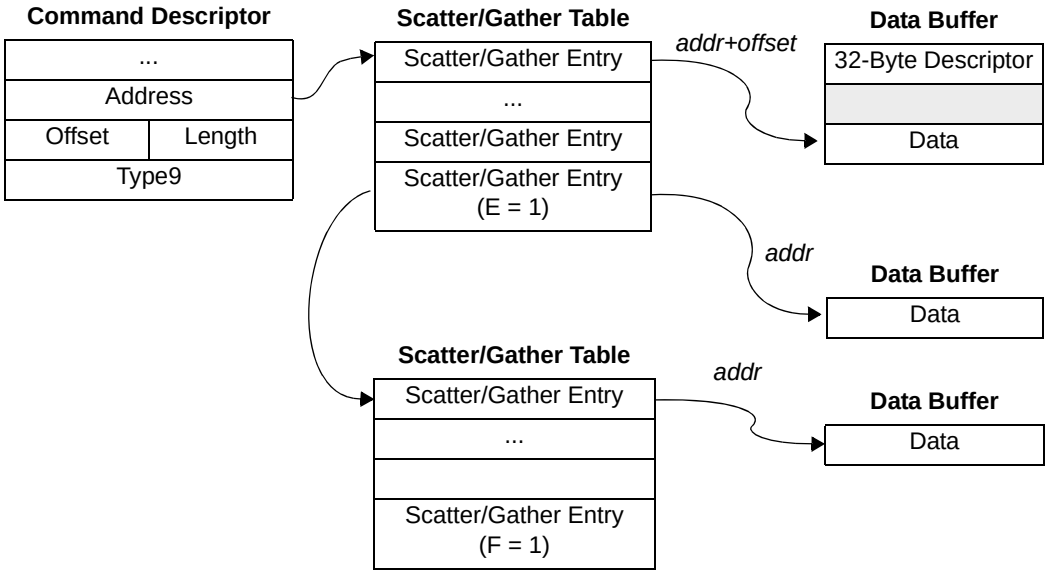


Figure 16-28. Outbound Type9 Scatter/Gather Table Format

16.3.8.2 Type9 Reassembly Descriptor Format

The Type9 inbound descriptor contains information received by the message unit. The message unit fills out the descriptor with the received control parameters before embedding a link to it in a command descriptor. The command descriptor is then enqueued onto an inbound message queue processed by the consumer.

Note: A non-Freescale RapidIO Type9 destination endpoint may be unsuccessful in correctly reassembling two or more PDUs from the same source with the same RapidIO flow when interleaving is done based on CoS. The eMSG segmentation engine includes the RapidIO field CoS (Class-of-Service) field when determining segmentation interleaving dependencies, thus PDUs of the same flow but different CoS may be interleaved. Use one of the following two methods to avoid this problem.

1. Assign outbound Type9 PDUs of the same flow with the same CoS to force a dependency between the PDUs, thus avoiding segmentation interleaving.
2. Set register bit MUMR[OSID] = 1 to disable segmentation interleaving on a destination ID basis. However, this affect all supported types and may affect performance.

Figure 16-29 shows the format of an inbound PDU descriptor. Reserved fields must be cleared.

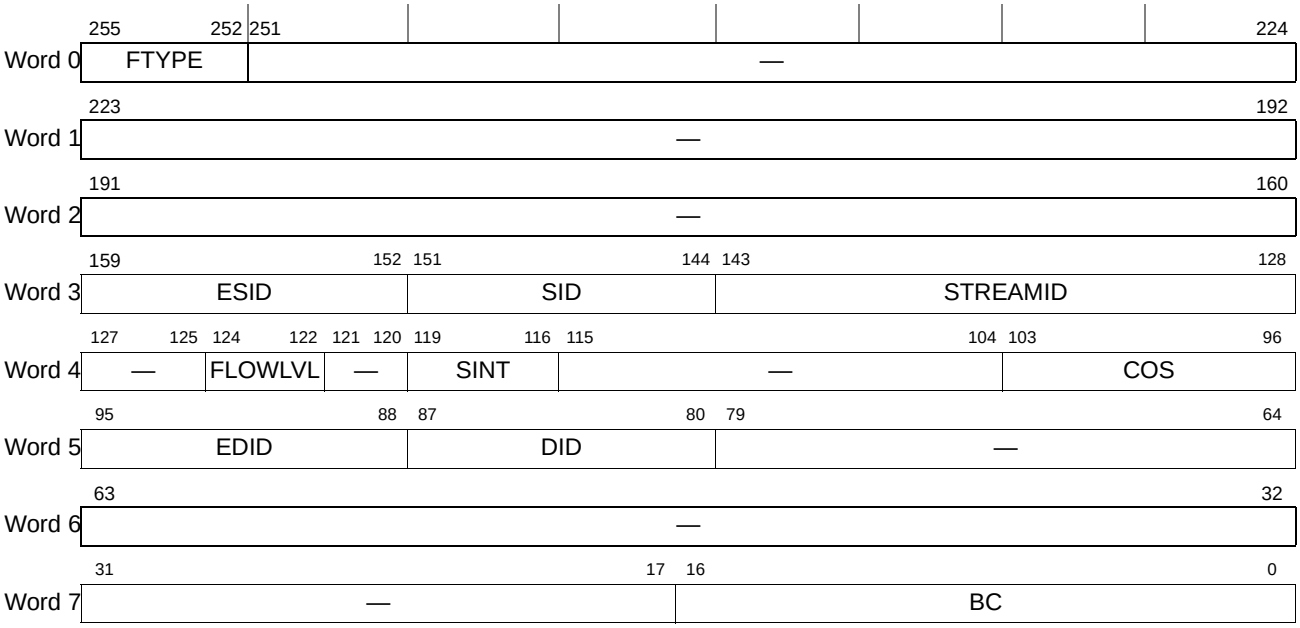


Figure 16-29. Type9 Inbound PDU Descriptor

Table 16-38 describes the Type9 inbound PDU descriptor fields.

Table 16-38. Type9 Inbound PDU Descriptor Field Descriptions

Bits	Name	Description
Word 0—Header		
255–252	FTYPE	Descriptor type select. 1001 Type9 All other values reserved.
251–224	—	Reserved
Word 1		
223–192	—	Reserved
Word 2		
191–160	—	Reserved
Word 3—Source Port		
159–152	ESID	Extended source ID**.** Most significant byte of a 16-bit source ID when operating in large transport mode. Reserved when operating in small transport mode.
151–144	SID	Source ID. Contains the source ID field of the transaction (Device ID of the source).
143–128	STREAMID	Value from the “streamID” field in the DATA STREAMING packets. In case of CSE/ESE, the stream ID is invalid.
Word 4—Source Attributes		
127–125	—	Reserved

Table 16-38. Type9 Inbound PDU Descriptor Field Descriptions (Continued)

Bits	Name	Description
124–122	FLOWLVL	Transaction flow level. 000 Lowest priority transaction request flow 001 Next highest priority transaction request flow ... 110 Next highest priority transaction request flow 111 Highest priority transaction request flow
121–120	—	Reserved
119–116	SINT	Source interface. 0000 RapidIO Port 1 0001 RapidIO Port 2 All other values reserved.
115–104	—	Reserved
103–96	COS	Value from the “COS” field in the DATA STREAMING packet.
Word 5—Destination Port		
95–88	EDID	Extended destination ID. Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
87–80	DID	Destination ID. Contains the destination ID field of the transaction (Device ID of the target).
79–64	—	Reserved
Word 6—Reserved		
63–32	—	Reserved
Word 7—Byte Count		
31–217	—	Reserved
16–0	BC	Byte count. Contains the number of bytes written to memory for the reassembly operation, which may be less than expected if an error was encountered during reassembly. 0x000011 byte 0x000022 bytes 0x000033 bytes ... 0x0FFFF64 Kbytes-1 0x1000064 Kbytes

Figure 16-30 shows the format for Type9 scatter/gather tables usage. The first entry of the scatter/gather table also holds the 32-byte descriptor. The first data offset is programmable

through the scatter/gather Data Offset Register MUSGDOR, but must be greater than or equal to the descriptor size. The data offset is specified through the scatter/gather entry offset field.

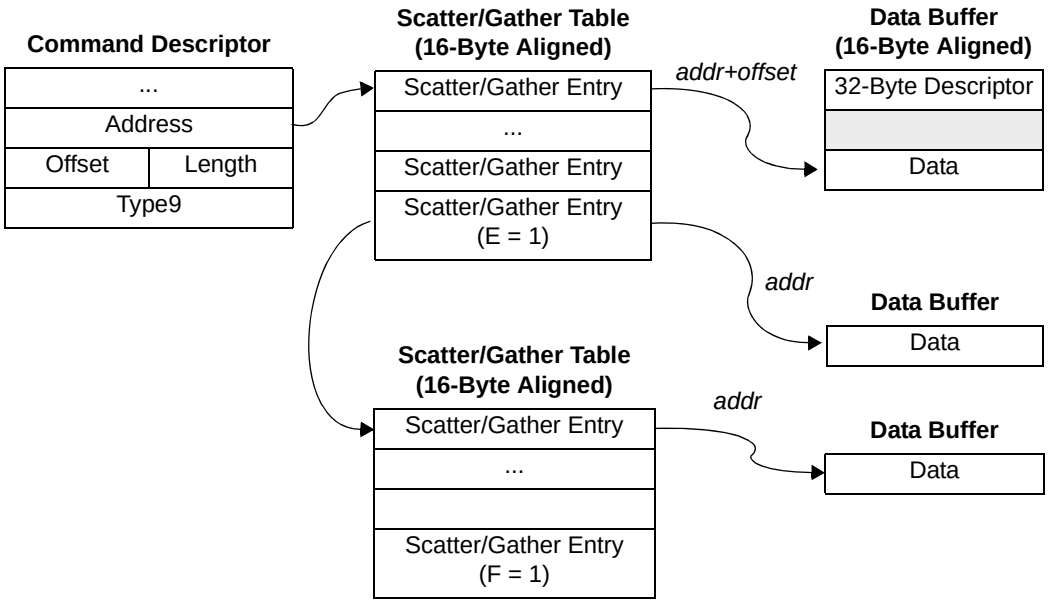


Figure 16-30. Inbound Type9 Scatter/Gather Table Format

16.3.8.3 Type9 Outbound Segmentation Operation

The segmentation units are responsible for sending PDUs enqueued by a producer or producers from one of the outbound frame queues owned by the message unit. Concurrent processing may also occur between any number of message queues as long as the outbound ordering rules apply, see **Section 16.3.14.2**, *Outbound Ordering Rules*.

Note: Do not use the scatter/gather table format for Type9 messages during outbound segmentation. Using this format can result in segmentation unit counters being corrupted by a descriptor error (DE).

16.3.8.3.1 Work Scheduling

The message unit receives all work from the outbound message queues. A producer or producers enqueueue work onto the outbound message queues owned by the message unit. To maintain any ordering between transactions, a producer must enqueue transactions with dependencies onto the same outbound message queue. The outbound ordering rules are described in **Section 16.3.14.2**, *Outbound Ordering Rules*.

16.3.8.3.2 Adding PDUs To A Message Queue

A producer may add new PDUs to any message queue owned by the message unit. The producer needs to allocate buffer space for the PDU descriptor, scatter/gather table(s) and data as appropriate and enqueue the command descriptor. The message unit dequeues the command

descriptor, determines the type, and proceeds to read the first scatter/gather table, message descriptor and data.

A producer should follow the guidelines below to add a new PDU to an outbound message queue:

- Allocate a data buffer to hold the PDU descriptor and all or part of the PDU payload. If buffer pools are used, the smallest buffer size may be preferred to reduce overhead.
- Write the 32-byte outbound PDU descriptor as described in **Section 16.3.8.1**, *Type9 Segmentation Descriptor Format*, at the beginning of the first data buffer.
  - Set the descriptor type select field FTYPE=4b1001 to indicate packet type.
  - For all other fields, see **Section 16.3.8.1**, *Type9 Segmentation Descriptor Format*.
- If multiple data buffers are required to hold the data payload, allocate buffer to hold the scatter/gather table to stitch smaller data buffers together. For best performance, use a small amount of data buffers and avoid linking scatter/gather tables.
- Write the scatter/gather table where the first entry holds a pointer to the first data buffer and the following entries points to additional data buffers or additional scatter/gather tables as needed, see **Section 16.3.4**, *Scatter/Gather Tables*. The first scatter/gather table entry must have an offset to where the data payload starts (greater than the PDU descriptor size).
- Add the first scatter/gather table to a command descriptor and set FTYPE=4b1001 to indicate packet type.
- Enqueue the command descriptor onto the outbound message queue.
- The message unit releases buffers used by the producer to hold the PDU descriptor and data payload when the operation completes if the buffer release bit (BR) is set in the message descriptor.

16.3.8.3.3 Segmentation Initialization

To configure the message unit for data streaming operations, See **Section 16.3.17.6**, *Initializing Outbound Message Queues*.

16.3.8.3.4 Error Handling

The message unit can report completion status and error information after completing the command descriptor by updating the command descriptor status field.

The following steps should be performed to enable this mode.

- Set the message descriptor completion (CS) or error (ES) status bit.
- Software should observe the status field of the command descriptor after it has been written to the completion queue to determine success.

Table 16-39 indicates the possible error conditions and the outcome when detected. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected, no additional error checking beyond the current error level is performed.

Table 16-39. Outbound Segmentation Hardware Errors

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Comments
Data Streaming Request	Command descriptor effective length less than message descriptor payload byte count.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Data Streaming Request	Invalid FTYPE.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Data Streaming Request	Invalid target interface specified.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Data Streaming Request	End of scatter/gather table reached before message descriptor payload byte count accounted for.	2	No	Descriptor error in command descriptor enqueued.	Partially	—
Data Streaming Request	Memory transaction error.	2	Message unit error if MUIER[TE] set	Transaction error in command descriptor enqueued. Message unit error detect register MUEDR[TE]. Command descriptor captured in MUECCDR0–3.	No	Message aborted. No additional buffers released.

16.3.8.3.4.1 Descriptor Error

When a descriptor error is detected by the message unit the following occurs.

- Completion message queue
  - The segmentation process is halted and the descriptor error bit DE in the command descriptor is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

16.3.8.3.4.2 Transaction Errors

When an transaction error occurs during a local memory data read by the message unit the following occurs.

- Segments that encounter a transaction error are not sent because the packet data is not available.
- Memory reads that were already generated before the transaction error occurred are also not transferred.
- Command descriptor captured in MUECCDR0–3.
- Global message unit error detect register MUEDR[OTE] set.
- Completion message queue
  - After the segmentation operation completes, the internal transaction error bit in the TE field, in the command descriptor, is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

16.3.8.4 Type9 Reassembly Operation

The message unit is responsible for receiving packets and placing then onto an inbound message queue. Software can specify the destination message queue for a reassembly operation utilizing packet classification, see **Section 16.3.17.2**, *Classification Initialization*. Utilizing the buffer pools initialized by software, the message unit requests buffers for creating scatter/gather tables, descriptor and data buffers to hold the PDU.

16.3.8.4.1 Reassembly Initialization

To configure the message unit for data streaming operations, see **Section 16.3.17.5**, *Initializing Inbound Message Queues*.

16.3.8.4.2 Packet Steering

Packets are forwarded to a message queue based on the first inbound Type9 classification unit (numbered 1 through 64) to successfully classify the PDU based on virtual stream ID. Message queue specified by IB*m*T9C*n*MQR.

16.3.8.4.3 Packet Drop Condition

The following conditions may result in dropped packets:

- Message unit has detected a resource conflict, that is, a new PDU context was received and hardware context resources were depleted.

16.3.8.4.4 Error Handling

The message unit reports errors, utilizing the inbound message queue, by setting the status field in the command descriptor before enqueueing the descriptor onto the designated inbound message queue. The consumer should read the status bits before processing the packet to determine the appropriate action.

Table 16-40 indicates the possible error conditions and the outcome when detected. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected, no additional error checking beyond the current error level is performed. In general, when an error has been detected the reassembly continues until the end of the PDU is reached.

Table 16-40. Inbound Reassembly Hardware Errors

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Data Segment Written	Message Descriptor Written	Comments
Start segment received for open context.	1	No	Bit SSE in command descriptor enqueued.	Yes. Partial defective PDU received.	Yes	New context is opened for starting segment. Old context is defective.
Segment request timeout.	1	No	Bit SRT in command descriptor enqueued.	No	Yes	PDU is defective.
Continuation segment received for new context.	2	No	Bit CSE in command descriptor enqueued.	No	Yes	PDU is defective. Message descriptor stream ID is invalid.
End segment received for new context.	2	No	Bit ESE in command descriptor enqueued.	No	Yes	PDU is defective. Message descriptor stream ID is invalid.
Source error. PDU was aborted by source.	2	No	Bit SE in command descriptor enqueued.	No	Yes	PDU is defective.
Maximum transmission unit.	2	No	Bit MTU in command descriptor enqueued.	No	Yes	PDU is defective.
Length field for end segment does not match actual byte count received.	2	No	Bit SME in command descriptor enqueued.	No	Yes	PDU is defective.
Transaction error.	3	Message unit error if MUIER[ITE] set.	Transaction error in command descriptor enqueued and Message unit error detect register MUEDR[ITE]. Address captured in MUECAR.	No	No	PDU is defective. Buffer may have been lost if scatter/gather table could not be updated.

Table 16-40. Inbound Reassembly Hardware Errors (Continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Data Segment Written	Message Descriptor Written	Comments
Buffer pool exhausted	3	Message unit error if MUIER[BAEIE] set.	Transaction error in command descriptor enqueued and message unit error detect register MUEDR[BAE].	No	No	PDU is defective.
Failure to enqueue the command descriptor onto the inbound message queue.	4	Message unit error if MUMIER[IMERIE] set.	Message unit error detect register MUMEDR[IMER]. Message queue captured in MUECMQR.	Yes	Yes	—

16.3.8.4.4.1 Single/Start Segment Error

When the single/start segment error is detected by the message unit the following occurs.

- The current open context is considered closed and the PDU is incomplete.
- The received single/start segment represents the first packet in a newly opened PDU context.
- The message unit sets the single/start segment error bit, SSE in the command descriptor.
- The command descriptor is enqueued onto the message queue.

16.3.8.4.4.2 Continuation Segment Error

When the continuation segment error is detected by the message unit the following occurs.

- Segment is discarded.
- The message unit sets the continuation segment error bit, CSE in the command descriptor.
- After the reassembly operation completes, the command descriptor is enqueued onto the message queue.

16.3.8.4.4.3 End Segment Error

When the end segment error is detected by the message unit the following occurs.

- Segment is discarded.
- The message unit sets the end segment error bit, ESE in the command descriptor.

After the reassembly operation completes, the command descriptor is enqueued onto the message queue.

16.3.8.4.4.4 Segment Request Timeout Errors

The segment request time-out counter starts after the first segment of a multi-segment PDU is received and the time-out counter is enabled. When a segment request time-out is detected, as defined by the RapidIO Port Response Time-out Control CSR, the following occurs.

- The current open context is considered closed and the PDU is incomplete.
- All reassembly packets that have not yet been received are considered complete.
- The message unit sets the segment request time-out error bit, SRT in the command descriptor.
- The command descriptor is enqueued onto the message queue.

16.3.8.4.4.5 MTU Violation Errors

When an MTU violation error is detected by the message unit, the following occurs.

- Packet is discarded.
- The message unit sets the MTU violation error bit, MTU in the command descriptor.
- After the reassembly operation completes, the command descriptor is enqueued to the message queue.

16.3.8.4.4.6 Source Errors

When a source error is detected by the message unit, the following occurs.

- The current open context is considered closed and the PDU is incomplete.
- All reassembly segments that have not yet been received are considered complete.
- The message unit sets the source error bit, SE in the command descriptor.
- After the reassembly operation completes, the command descriptor is enqueued to the message queue.

16.3.8.4.4.7 Size Mismatch Errors

When a size mismatch error is detected by the message unit, the following occurs.

- The current open context is considered closed and the PDU is incomplete.
- All reassembly packets that may not yet been received are considered complete.
- The message unit sets the size mismatch error bit, SME in the command descriptor.
- After the reassembly operation completes, the command descriptor is enqueued onto the message queue.

16.3.8.4.4.8 Transaction Errors

When an internal transaction error occurs, due to insufficient buffer resources or local memory write error, by the message unit the following occurs.

- Segment is discarded.
- The message unit sets the transaction error bit, TE, in the command descriptor.
- After the reassembly operation completes, the command descriptor is enqueued onto the message queue.
- Global message unit error detect register MUEDR[ITE] set for memory access error.
- Address captured in MUECAR for memory access error.
- Global message unit error detect register MUEDR[BAE] set if buffer resources depleted.

16.3.8.5 Flow Control Management

The message unit supports only basic stream management flow control (XON/XOFF) using Type9 extended header format. Any other flow control format received is accepted as basic flow control. Flow control messages are received as well as sent by the message unit. Flow control is enabled and controlled through register DSLLCCSR, hosted by the serial RapidIO controller. In addition, inbound queues are flow controlled automatically when reaching a critical level to avoid overflow of a queue.

16.3.8.5.1 Receiving Flow Control Messages

The message unit may receive stream management flow control messages. If flow control is disabled, these messages are ignored. For receiving flow control to work properly one or more classification rules must be configured for Type9 extended flow control targeting the outbound queue as specified in register IB*m*T9C*n*MQR. The block number is determined by the first rule matched. All targets for outbound queue N receiving the flow control should be considered.

Flow control messages must match inbound classification based on the following rules:

- RapidIO type9 extended.
- RapidIO port.
- RapidIO port source ID.
- If message contains wildcarded destination ID, a match always occurs, else message applies to a specific destination and the classification destination mask must be zero.
- If message contains wildcarded class-of-service, a match always occurs, else message applies to a specific set of classes which requires classification class mask to be a sub-set of the message class mask with a match on the remaining unmasked class bits.
- If traffic management stream ID is wildcarded a match always occurs, if not wildcarded an exact rule value match is required and the classification stream ID mask must be zero.

When stream management flow control is enabled, only certain combinations of class-of-service and stream ID rule mask fields are permitted for inbound classification.

- Specific stream in a specific class for specific destination:
  <DestID><CoS><StreamID>

- All streams in a specific class for a specific destination:
  <DestID><CoS>< xx >
- All streams in a group of classes for a specific destination:
  <DestID><CoS>< xx >
  where class mask if not 0x00
- All streams and classes for a specific destination:
  <DestID>< xx >< xx >
  where CoS and stream ID are wildcarded
- All streams and classes for all destinations:
  < xx >< xx >< xx >
  where destination, CoS and stream ID are wildcarded

An accepted flow control message exits (XOFF) or enters (XON) congestion management for the outbound message queue associated with the classification rule.

16.3.8.5.2 Sending Flow Control Messages

The message unit receives a congestion state change notification (CSCN) from an inbound message queue when the queue size has reached a low/high watermark, indicating the need for flow control of the source(s) of traffic. To enable the acceptance of Congestion State Change Notifications (CSCNs) from the inbound message queues perform the following:

1. Enable flow control through DSLLCCSR[TME].
2. Initialize the congestion enter/exit watermarks using IB*m*MQ*n*CMR.
3. Initialize the inbound Type9 classification flow control destination register using IB*m*T9C*n*FCDR.

For each enabled Type9 inbound classification rule targeting the inbound message queue, which is requesting flow control, an XON/XOFF stream management flow control message is generated to the programmed per port flow control destination ID register specified in the rule. An SRIO port which is in a powerdown state must not be included in the classification rule. If the inbound classification rule indicates a wildcarded source ID, the flow control destination ID register should represent a multicast target to reach all possible sources. An XON message is generated only if the queue is already in XOFF state, while an XOFF message is generated each time the enter congestion threshold is crossed.

All flow control requests are issued with the highest RapidIO flow.The traffic management operation is defined by the following fields (no other combination supported):

<DestID><CoS><StreamID> + <wild cards> + <Mask>

The following operands are generated by the message unit:

- Specific stream (classification rule with no stream wildcard):
  <DestID><CoS><StreamID> + <wc=000> + <Mask=0x00>

- All streams in a specific class (classification rule with stream wildcarded):
  <DestID><CoS>< xx > + <wc=001> + <Mask=0x00>
  where StreamID is a don't care.
- All streams in a group of classes (classification rule with class-of-service wildcarded):
  <DestID><CoS>< xx > + <wc=001> + <Mask=0xnn>
  where mask is one of the non-zero values except 0xff defined in **Table 16-132**.
- All streams and classes (classification rule with class-of-service wildcarded):
  <DestID>< xx >< xx > + <wc=011> + <Mask= xx >
  where CoS, StreamID and mask are don't cares.

Figure 16-31 shows an example of four classification rules generating traffic to a single message queue. Based on the value/mask setting of the Type9 rules, a set of flow control messages are generated.

Classification Rule 0	Value	Mask
CoS Stream ID Port	0xAA 0xBBBB —	0x00 0x0000 1
FC Destination ID (Port 1/2)	A/B	

Classification Rule 1	Value	Mask
CoS Stream ID Port	0xCC — 0	0x00 0xFFFF 0
FC Destination ID (port 1/2)	C/D	

Classification Rule 2	Value	Mask
CoS Stream ID Port	0xDD — 1	0x1F 0xFFFF 0
FC Destination ID (port 1/2)	E/F	

Classification Rule 3	Value	Mask
CoS Stream ID Port	— — —	0xFF 0xFFFF 1
FC Destination ID (port 1/2)	G/H	

Outbound Type9 Flow Control Messages

DestID	CoS	StreamID	WC	Mask	Port
A	0xAA	0xBBBB	b000	0x00	0
B	0xAA	0xBBBB	b000	0x00	1
C	0xCC	—	b001	0x00	0
F	0xDD	—	b001	0x1F	1
G	—	—	b011	—	0
H	—	—	b011	—	1

Figure 16-31. Example of Outbound Type9 Flow Control Generation

16.3.9 Type10 Doorbell Functional Description

16.3.9.1 Type10 Outbound Doorbell Descriptor Format

The Type10 outbound descriptor contains information for the message unit to send a doorbell. The message unit receives the outbound doorbell descriptor by dequeueing a command descriptor from the outbound message queue. The command descriptor points to a data buffer containing the descriptor and doorbell data.

Figure 16-32 shows the format of an outbound doorbell descriptor. Reserved fields must be cleared.

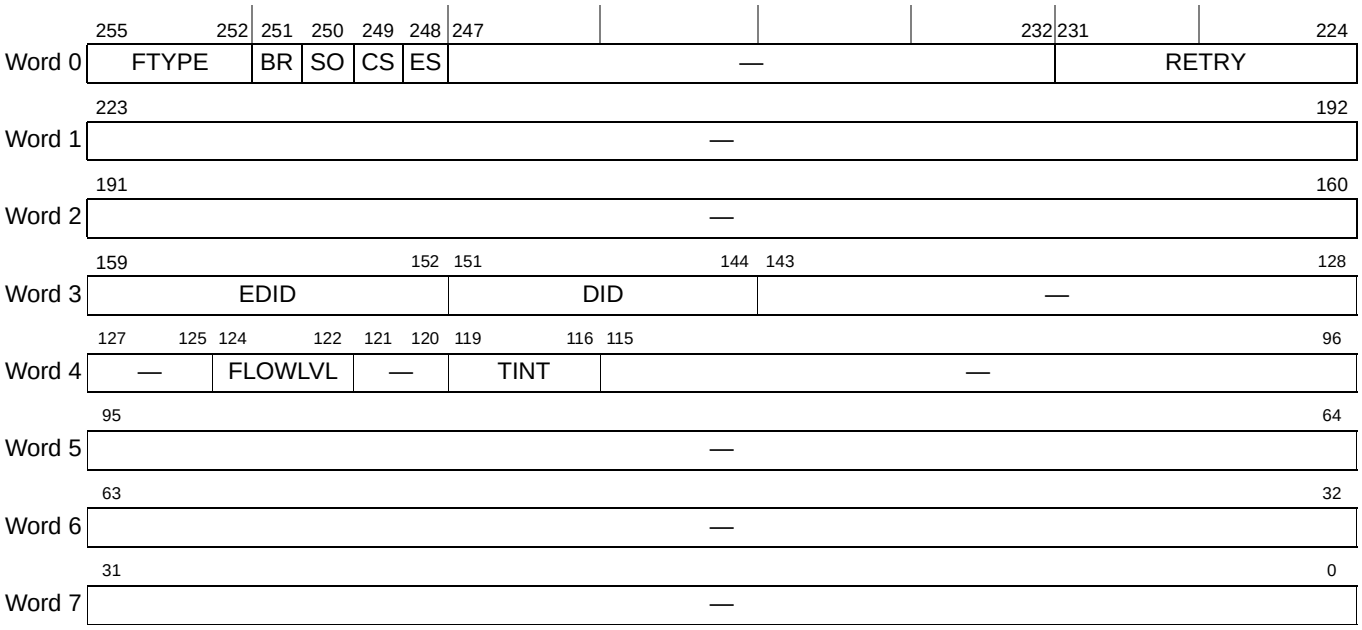


Figure 16-32. Type10 Outbound Doorbell Descriptor

Table 16-41 describes the Type10 outbound doorbell descriptor fields.

Table 16-41. Type10 Outbound Doorbell Descriptor Field Descriptions

Bits	Name	Description
Word 0—Header		
255–252	FTYPE	Descriptor type select. 1010 Type10 All other values reserved.
251	BR	Buffer release enable. 0 No hardware buffer releases performed. 1 Hardware buffer releases performed. If TE/DE error status was reported, not all buffers may have been released successfully.
250	SO	Strict ordering. A Type10 doorbell of same or lower priority than any other transaction is strictly ordered when the destination target ID and Rapid IO port interface is the same. 0 Disabled 1 Enabled
249	CS	Completion status. 0 Status message queue is not updated when doorbell completes normally. 1 Status message queue is updated when doorbell completes normally.
248	ES	Error status 0 Status message queue is not updated when doorbell aborted due to error. 1 Status message queue is updated when doorbell aborted due to error.
247–232	—	Reserved

Table 16-41. Type10 Outbound Doorbell Descriptor Field Descriptions (Continued)

Bits	Name	Description
231–224	RETRY	Retry error threshold. This value is the number of times that the message unit attempts to retransmit the doorbell to a particular target before reporting an error. The doorbell is retransmitted if a RETRY response is received from the target. 0x00 Disabled 0x01 Doorbell retransmitted only 1 time 0x02 Doorbell retransmitted up to 2 times ... 0xFF Doorbell retransmitted up to 255 times
Word 1—Reserved		
223–192	—	Reserved
Word 2—Reserved		
191–160	—	Reserved
Word 3—Destination Port		
159–152	EDID	Extended destination ID Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
151–144	DID	Destination ID Contains the destination ID field of the transaction (device ID of the target).
143–128	—	Reserved
Word 4—Destination Attributes		
127–125	—	Reserved
124–122	FLOWLVL	Transaction flow level 000 Lowest priority transaction request flow 001 Next highest priority transaction request flow ... 100 Next highest priority transaction request flow 101 Highest priority transaction request flow 110 Reserved 111 Reserved
121–120	—	Reserved
119–116	TINT	Target interface 0000 RapidIO Port 1 0001 RapidIO Port 2 All other values reserved.
115–96	—	Reserved
Word 5—Reserved		
95–64	—	Reserved
Word 6—Reserved		
63–32	—	Reserved
Word 7—Reserved		
31–0	—	Reserved

Figure 16-33 shows the command descriptor referencing a single buffer directly. The data payload offset is specified using the command descriptor offset field.

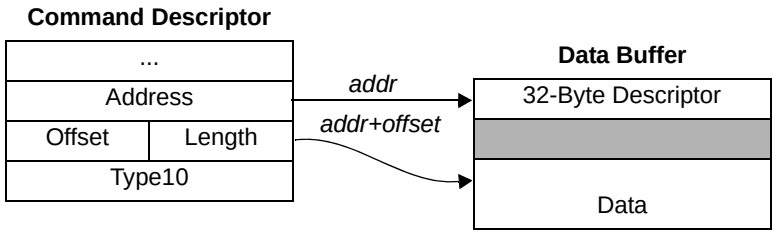


Figure 16-33. CD Using Single Buffer Referencing Type10 Outbound Message

16.3.9.2 Type10 Inbound Doorbell Descriptor Format

This section defines the format of the doorbell descriptor written to memory by the message unit. The message unit fills out the descriptor with the received control parameters before embedding a link to it in a command descriptor. The command descriptor is then enqueued onto an inbound message queue processed by the consumer.

Figure 16-34 shows the format of an inbound doorbell descriptor. Reserved fields must be cleared.

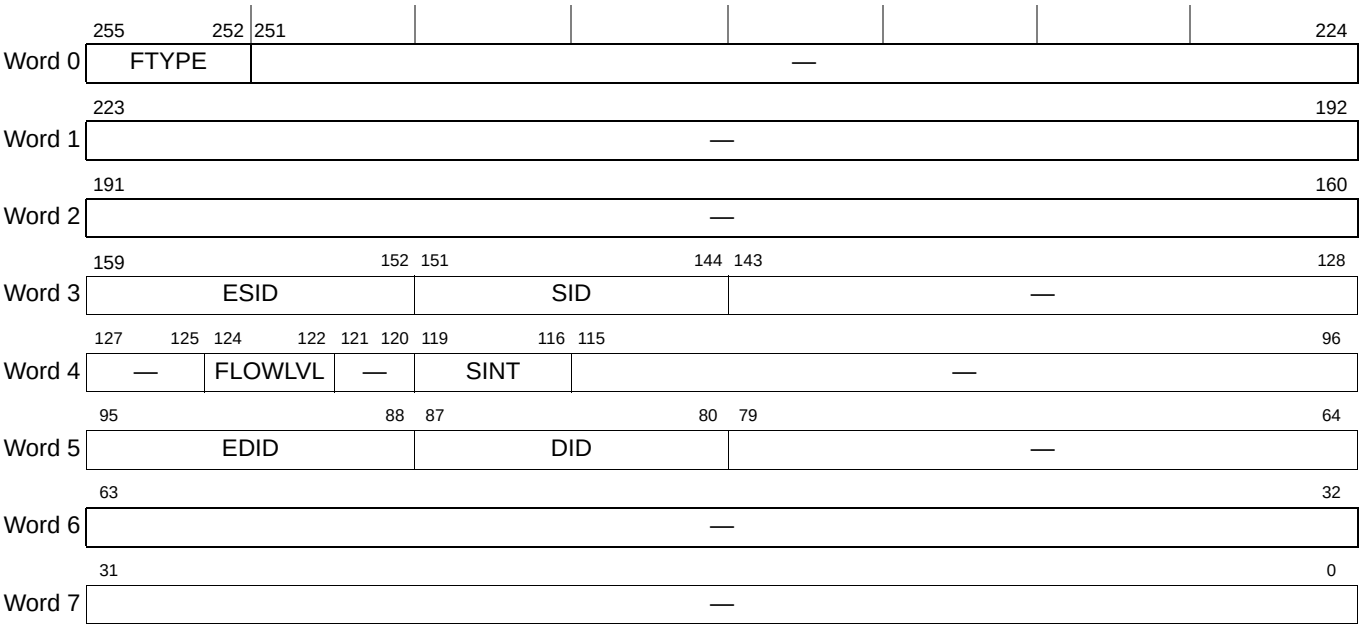


Figure 16-34. Type10 Inbound Doorbell Descriptor

Table 16-42 describes the Type10 inbound doorbell descriptor fields.

Table 16-42. Type10 Inbound Doorbell Descriptor Field Descriptions

Bits	Name	Description
Word 0—Header		
255–252	FTYPE	Descriptor type select. 1010 Type10 All other values reserved.
251–224	—	Reserved
Word 1–2—Reserved		
223–160	—	Reserved
Word 3—Source Port		
159–152	ESID	Extended source ID Most significant byte of a 16-bit source ID when operating in large transport mode. Reserved when operating in small transport mode.
151–144	SID	Source ID Contains the source ID field of the transaction (device ID of the source).
143–128	—	Reserved
Word 4—Source Attributes		
127–125	—	Reserved
124–122	FLOWLVL	Transaction flow level 000 Lowest priority transaction request flow 001 Next highest priority transaction request flow ... 100 Next highest priority transaction request flow 101 Highest priority transaction request flow 110 Reserved 111 Reserved
121–120	—	Reserved
119–116	SINT	Source interface. 0000 RapidIO Port 1 0000 RapidIO Port 2 All other values reserved.
115–96	—	Reserved
Word 5—Destination Port		
95–88	EDID	Extended destination ID Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
87–80	DID	Destination ID Contains the destination ID field of the transaction (Device ID of the destination).
79–64	—	Reserved
Word 6—Reserved		
63–32	—	Reserved

Table 16-42. Type10 Inbound Doorbell Descriptor Field Descriptions (Continued)

Bits	Name	Description
Word 7—Reserved		
31–0	—	Reserved

Figure 16-39 shows the supported inbound format for Type10 data management. The command descriptor address points directly to the start of the message descriptor. The data offset is specified through the command descriptor offset field. The data offset is programmable through IT10C*n*DOR, but must be greater than or equal to the descriptor size.

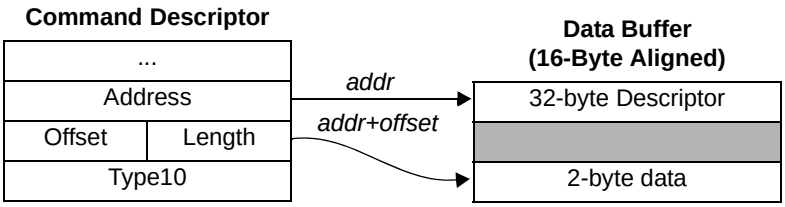


Figure 16-35. Command Descriptor Referencing a Type10 Inbound Doorbell

16.3.9.3 Type10 Outbound Doorbell Operation

RapidIO supports a doorbell type that contains a fixed 2-byte data payload. The segmentation units are responsible for sending messages enqueued by a producer or producers from one of the outbound frame queues owned by the message unit. Concurrent processing may also occur between any number of message queues as long as the outbound ordering rules apply, see **Section 16.3.14.2**, *Outbound Ordering Rules*.

16.3.9.3.1 Work Scheduling

The message unit receives all work from the outbound message queues. A producer or producers enqueueue work onto the outbound message queues owned by the message unit. To maintain any ordering between transactions, a producer must enqueue transactions with dependencies onto the same outbound message queue. The outbound ordering rules are described in **Section 16.3.14.2**, *Outbound Ordering Rules*.

16.3.9.3.2 Adding Doorbells To A Message Queue

A producer may add new doorbells to any message queue owned by the message unit. The producer needs to allocate buffer space for the doorbell descriptor as appropriate and enqueue the command descriptor. The message unit dequeues the command descriptor, determines the type, and proceeds to read the doorbell descriptor.

A producer should follow the guidelines below to add a new doorbell to a message queue:

- Create the 32-byte outbound doorbell descriptor as described in **Section 16.3.9.1**, *Type10 Outbound Doorbell Descriptor Format*.
  - Set the descriptor type select field FTYPE=0b1010 to indicate doorbell type.
  - Set the retry field to indicate number or times the message unit should attempt to retry the doorbell before stopping.
  - Allocate buffer for the doorbell descriptor from one of the buffer pools available, preferable with the smallest buffer size (that is, 64-bytes).
  - For all other fields, see **Section 16.3.9.1**, *Type10 Outbound Doorbell Descriptor Format*.
- Add the doorbell descriptor to a command descriptor and set FTYPE=0b1010 to indicate doorbell type.
- Enqueue the command descriptor onto the message queue.
- The message unit releases buffers used by the producer to hold the doorbell descriptor and data payload when the operation completes; if the buffer release bit (BR) is set in the message descriptor.

16.3.9.3.3 Doorbell Initialization

See **Section 16.3.17.6**, *Initializing Outbound Message Queues*.

16.3.9.3.4 Error Handling

The message unit can report completion status and error information after completing the command descriptor by updating the command descriptor status field.

The following steps should be performed to enable this mode.

- Set the message descriptor completion (CS) or error (ES) status bit.
- Software should observe the status field of the command descriptor after it has been written to the completion queue to determine success.

Table 16-43 indicates the possible error conditions and the outcome when detected. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected, no additional error checking beyond the current error level is performed.

Table 16-43. Outbound Doorbell Hardware Errors

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Doorbell Sent	Comments
Doorbell Request	Command descriptor effective length less than message descriptor payload byte count.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Doorbell Request	Invalid FTYPE.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Doorbell Request	Data format (scatter/gather) not supported.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Doorbell Request	Invalid flow level specified.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Doorbell Request	Invalid target interface specified.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Doorbell Request	Memory transaction error.	2	Message unit error if MUIER[OTE] set	Transaction error in command descriptor enqueued. Message unit error detect register MUEDR[OTE]. Command descriptor captured in MUECCDR0–3.	No	Doorbell aborted. No additional buffers released.
Doorbell Response	Number of retries exceeded.	3	No	Retry threshold exceeded in command descriptor enqueued.	Yes	Doorbell aborted.
Doorbell Response	Doorbell response timeout.	3	No	Segment response timeout in command descriptor enqueued.	Yes	Doorbell aborted.

16.3.9.3.4.1 Descriptor Error

When a descriptor error is detected by the message unit the following occurs.

- Completion message queue
  - The segmentation process is halted and the descriptor error bit DE in the command descriptor is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

16.3.9.3.4.2 Doorbell Error Response Errors

When a doorbell error response is received by the message unit the following occurs.

- Completion message queue
  - After the doorbell operation completes, the error response received bit, ERR, in the command descriptor, is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

16.3.9.3.4.3 Doorbell Response Time-out Errors

When a doorbell response time-out is detected, as defined by the RapidIO Port Response Time-out Control CSR, the following occurs.

- Completion message queue
  - After the doorbell operation completes the doorbell response time-out error bit, SRT, in the command descriptor, is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

16.3.9.3.4.4 Retry Threshold Exceeded Errors

When a retry threshold exceeded error occurs for a doorbell the following occurs.

- Completion message queue
  - After the doorbell operation completes the retry threshold exceeded bit in the RTE field, in the command descriptor, is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

16.3.9.3.4.5 Transaction Errors

When an transaction error is detected during a local memory data read by the message unit the following occurs.

- Doorbells that encounter a transaction error are not sent because the doorbell data is not available.
- Memory reads that were already generated before the transaction error occurred are also not transferred. This includes retried message segments.
- Command descriptor captured in MUECCDR0–3.
- Global message unit error detect register MUEDR[OTE] set.
- Completion message Queue
  - After the doorbell operation completes the internal transaction error bit, TE, in the command descriptor is set.

— The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

- Buffer Release (BR=1)
  — No further buffers are released once a TE error is detected.

16.3.9.4 Type10 Inbound Doorbell Operation

The message unit is responsible for receiving RapidIO Type10 doorbells and placing then onto an inbound message queue. Software can specify the destination message queue for the doorbell operation utilizing doorbell classification, see **Section 16.3.17.2**, *Classification Initialization*. Utilizing the buffer pools initialized by software, the message unit requests buffers for creating the descriptors, which also holds the doorbell data.

Utilizing the buffer pools, the message unit requests a single buffer to hold the message descriptor and data. Software must provide the required sized buffer pool in IB*m*T10C*n*BPR for the message unit to store the entire message plus descriptor. The data offset is defined in IB*m*T10C*n*DOR. The message unit generated a frame size error if the inbound doorbell cannot be stored due to insufficient buffer sizes available.

16.3.9.4.1 Doorbell Initialization

To configure the message unit for doorbell operations, see **Section 16.3.17.5**, *Initializing Inbound Message Queues*.

16.3.9.4.2 Doorbell Steering

Doorbells are forwarded to a message queue based on the first classification unit (numbered 1 through 64) to successfully classify the doorbell based on source ID, and flow attributes. Message queue specified by IB*m*T10C*n*MQR.

16.3.9.4.3 Retry Response Condition

The conditions to generate a logical layer retry response are:

- Message unit has detected a resource conflict, that is, a doorbell was received and hardware context resources were depleted.

All responses are upgraded to priority 3.

16.3.9.4.4 Error Response Condition

The conditions to generate a logical layer error response are:

- Message unit did not have a matching classification rule.
- Message unit has detected a buffer size error.
- Message unit has detected a transaction error when allocating buffer resources or performing a local memory write.

All responses are upgraded to priority 3.

16.3.9.4.5 Error Handling

The message unit reports errors, utilizing the inbound message queue, by setting the status field in the command descriptor before enqueueing the descriptor onto the designated inbound message queue. The consumer should read the status bits before processing the doorbell to determine the appropriate action.

Table 16-44 indicates the possible error conditions and the outcome when detected. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected, no additional error checking beyond the current error level is performed.

Table 16-44. Inbound Doorbell Hardware Errors

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Data Segment Written	Message Descriptor Written	Response	Comments
Illegal RapidIO request priority 3.	1	Message unit error if MUIER[MFE] set.	Message unit error detect register MUEDR[MFE].	N/A	N/A	None	Transaction dropped.
Buffer size too small (buffer size error).	2	No	Buffer size error in command descriptor enqueued.	No	Yes	Error	Doorbell is defective.
Memory transaction error.	3	Message unit error if MUMIER[ITE] set.	Transaction error in command descriptor enqueued and message unit error detect register MUEDR[ITE]. Address captured in MUECAR.	Possibly	No	Error	Doorbell is defective.
Buffer pool exhausted.	3	Message unit error if MUIER[BAEIE] set.	Transaction error in command descriptor enqueued and Message Unit Error Detect Register MUEDR[BAE].	No	No	Error	Doorbell is defective.
Failure to enqueue the command descriptor onto the inbound message queue.	4	Message unit error if MUIER[IMERIE] set.	MUEDR[IMER]. Message queue captured in MUECMQR.	Yes	Yes	None	—

16.3.9.4.5.1 Buffer Size Errors

When a buffer size error is detected by the message unit, the following occurs.

- Doorbell is discarded and an error response is returned for the segment.

- The message unit sets the buffer size error bit, BSE in the command descriptor.
- After the doorbell operation completes, the command descriptor is enqueued onto the message queue.

16.3.9.4.5.2 Transaction Errors

When an transaction error occurs, due to insufficient buffer resources or local memory write error, by the message unit the following occurs.

- Doorbell payload and message descriptor are defective and an error response is returned.
- The message unit sets the transaction error bit, TE, in the command descriptor.
- After the doorbell operation completes, the command descriptor is enqueued onto the message queue.
- Global message unit error detect register MUEDR[ITE] set for memory access error.
- Address captured in MUECAR for memory access error.
- Global message unit error detect register MUEDR[BAE] set if buffer resources depleted.

16.3.10 Type11 Message Functional Description

16.3.10.1 Type11 Outbound Message Descriptor Format

The Type11 outbound descriptor contains information for the message unit to send a messages. The message unit receives the outbound message descriptor by dequeueing a command descriptor from the outbound message queue. The command descriptor points to a data buffer containing the descriptor and message data.

Figure 16-36 shows the format of an outbound message descriptor. Reserved fields must be cleared.

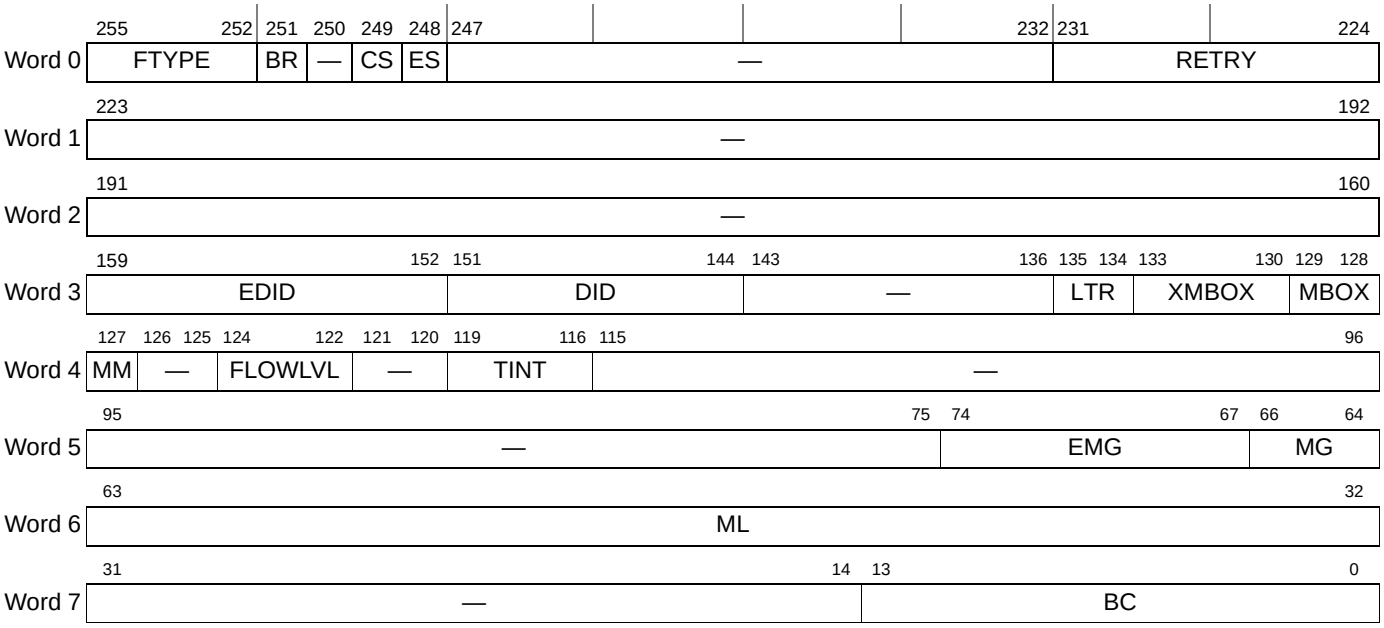


Figure 16-36. Type11 Outbound Message Descriptor

Table 16-45 describes the Type11 outbound message descriptor fields.

Table 16-45. Type11 Outbound Message Descriptor Field Descriptions

Bits	Name	Description
Word 0—Header		
255–252	FTYPE	Descriptor type select. 1011 Type11 All other values reserved.
251	BR	Buffer release enable. 0 No hardware buffer releases performed. 1 Hardware buffer releases performed. If TE/DE error status was reported, not all buffers may have been released successfully.
250	—	Reserved
249	CS	Completion status. 0 Status message queue is not updated when message completes normally. 1 Status message queue is updated when message completes normally.
248	ES	Error status 0 Status message queue is not updated when message aborted due to error. 1 Status message queue is updated when message aborted due to error.
247–232	—	Reserved
231–224	RETRY	Retry error threshold. This value is the number of times that the message unit attempts to retransmit a message segment to a particular target before reporting an error. A message segment is retransmitted if a RETRY response is received from the target. 0x00 Disabled 0x01 Message segment retransmitted only 1 time 0x02 Message segment retransmitted up to 2 times ... 0xFF Message segment retransmitted up to 255 times
Word 1—Reserved		

Table 16-45. Type11 Outbound Message Descriptor Field Descriptions (Continued)

Bits	Name	Description
223–192	—	Reserved
Word 2—Reserved		
191–160	—	Reserved
Word 3—Destination Port		
159–152	EDID	Extended destination ID. Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
151–144	DID	Destination ID. Contains the destination ID field of the transaction (Device ID of the target). This value is overridden by the multicast group and list if multicast mode is enabled. On error, if multicast mode is enabled, this field is loaded with the destination of the failed operation.
143–136	—	Reserved
135–134	LTR	Value for “letter” field in MESSAGE packet.
133–130	XMBOX	Value for “xmbox” field in MESSAGE packet. Valid only for single segment messages. Software must set this field to zero for a multi-segment message, that is, larger than 256 bytes.
129–128	MBOX	Value for “mbox” field in MESSAGE packet.
Word 4—Destination Attributes		
127	MM	Multicast mode. 0 Disabled. 1 Message operation is sent to all of the targets indicated by the multicast group and list. Messages are limited to one segment and 256 bytes or less when this mode is enabled.
126–125	—	Reserved
124–122	FLOWLVL	Transaction flow level. 000 Lowest priority transaction request flow 001 Next highest priority transaction request flow ... 100 Next highest priority transaction request flow 101 Highest priority transaction request flow 110 Reserved 111 Reserved
121–120	—	Reserved
119–116	TINT	Target interface. 0000 RapidIO Port 1 0000 RapidIO Port 2 All other values reserved.
115–96	—	Reserved
Word 5—Multicast Group		

Table 16-45. Type11 Outbound Message Descriptor Field Descriptions (Continued)

Bits	Name	Description
The multicast group (MG) value and extended multicast group number (EMG) which, in combination with the multicast list (ML) and the multicast enable (MM), indicates which device IDs are targets of a multicast operation. The multicast group represents the most significant three bits (bits[0–2]) of the RapidIO deviceIDs that are destinations of the message operation, whereas the multicast list indicates a list of targets within that group. Each individual set bit, when encoded, determines the least significant five bits of the RapidIO device IDs (bits[3–7]) that are targets of the message operation. Therefore, multicast group 0 (MG=0) contains target deviceIDs (0,1,...,31), multicast group 1 (MG=1) contains target deviceIDs (32,33,...63), etc. If in large transport mode, the extended multicast group represents the eight most significant bits (bits[0–7]), the multicast group represents the next three bits (bits[8–10]) 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 DID and EDID fields.		
95–75	—	Reserved
74–67	EMG	Extended multicast group. This is the most significant eight bits of the target device IDs for the multicast operation when operating in large transport mode.
66–64	MG	Multicast group. This is the most significant three bits of the target device IDs for the multicast operation.
Word 6—Multicast List		
63–32	ML	Multicast list. This is the group target list 0–31 for the message operation. Depending upon the value of the multicast group value, target 0 corresponds to device ID 0, 32, 64, 96, e.t.c., target 1 corresponds to deviceID 1, 33, 65, 97, e.t.c. and so on. If none of the bits are set, target 0 is assumed to be set.
Word 7—Byte Count		
31–14	—	Reserved
13–0	BC	Byte count. Contains the number of bytes for the message operation. 0x0001 1 byte 0x0002 2 bytes 0x0003 3 bytes ... 0x0FFF 4095 bytes 0X1000 4096 bytes (4 Kbytes) Byte counts greater than 256 are valid for multi-segment mode only. The maximum message operation size is 4 Kbytes and the minimum is 1 byte. Hardware performs automatic padding to double-word size. Number of segments is determined by the byte count divided by the segment size (256 bytes).

Figure 16-37 shows the command descriptor referencing a single buffer directly. The data payload offset is specified using the command descriptor offset field.

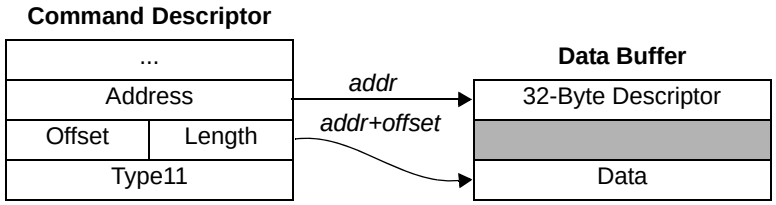


Figure 16-37. Command Descriptor Using Single Buffer Referencing Type11 Outbound Message

16.3.10.2 Type11 Inbound Message Descriptor Format

The Type11 inbound descriptor contains information received by the message unit. The message unit fills out the descriptor with the received control parameters before embedding a link to it using the predefined command descriptor format. The command descriptor is then enqueued onto an inbound message queue.

Figure 16-38 shows the format of an inbound message descriptor. Reserved fields must be cleared.

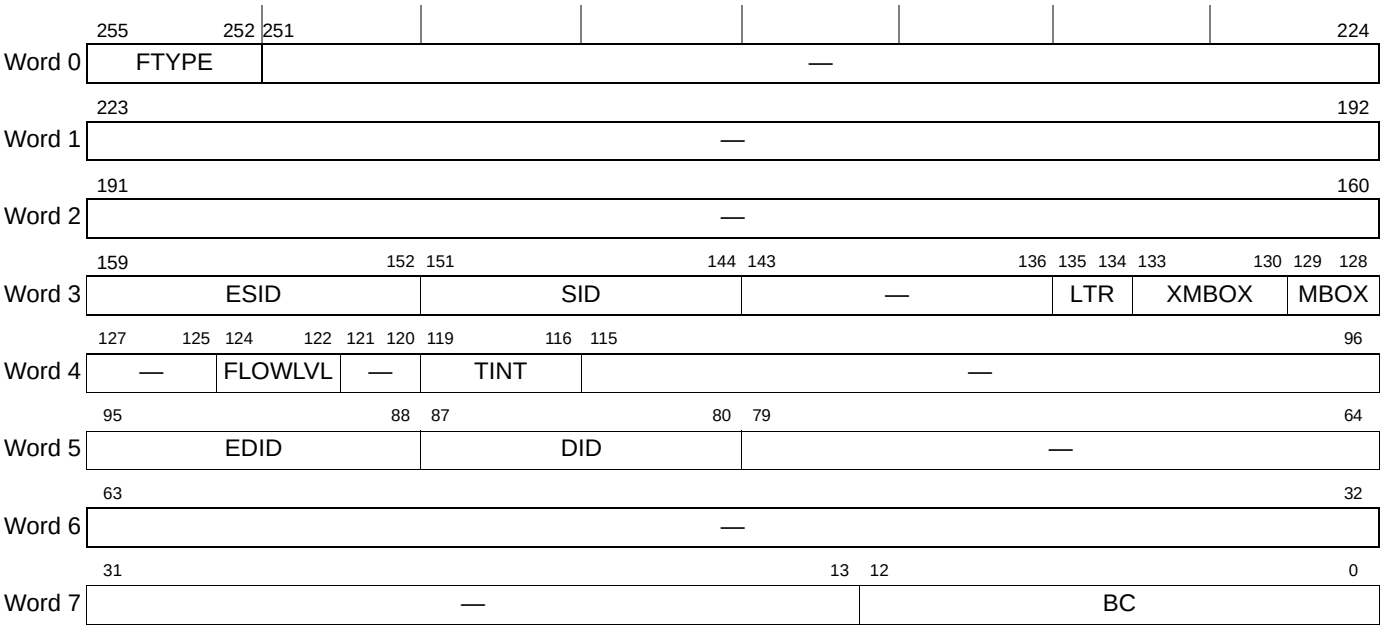


Figure 16-38. Type11 Inbound Message Descriptor

Table 16-46 describes the Type11 inbound message descriptor fields.

Table 16-46. Type11 Inbound Message Descriptor Field Descriptions

Bits	Name	Description
Word 0—Header		
255–252	FTYPE	Descriptor type select. 1011 Type11 All other values reserved.

Table 16-46. Type11 Inbound Message Descriptor Field Descriptions (Continued)

Bits	Name	Description
251–224	—	Reserved
Word 1–2—Reserved		
223–160	—	Reserved
Word 3—Source Port		
159–152	ESID	Extended source ID. Most significant byte of a 16-bit source ID when operating in large transport mode. Reserved when operating in small transport mode.
151–144	SID	Source ID. Contains the source ID field of the transaction (Device ID of the source).
143–136	—	Reserved
135–134	LTR	Value of “letter” field in MESSAGE packet.
133–130	XMBOX	Value of “xmbox” field in MESSAGE packet for single segment messages. For multi-segment messages, that is, greater than 256 bytes, this field is unused.
129–128	MBOX	Value from “mbox” field in MESSAGE packet.
Word 4—Source Attributes		
127–125	—	Reserved
124–122	FLOWLVL	Transaction flow level. 000 Lowest priority transaction request flow 001 Next highest priority transaction request flow ... 100 Next highest priority transaction request flow 101 Highest priority transaction request flow 110 Reserved 111 Reserved
121–120	—	Reserved
119–116	SINT	Source interface. 0000 RapidIO Port 1 0000 RapidIO Port 2 All other values reserved.
115–96	—	Reserved
Word 5—Destination Port		
95–88	EDID	Extended destination ID. Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
87–80	DID	Destination ID. Contains the destination ID field of the transaction (Device ID of the destination).
79–64	—	Reserved
Word 6—Reserved		
63–32	—	Reserved
Word 7—Byte Count		

Table 16-46. Type11 Inbound Message Descriptor Field Descriptions (Continued)

Bits	Name	Description
31–13	—	Reserved
12–0	BC	Byte count. Contains the number of bytes written to memory for the message operation, which may be less than expected if an error was encountered during reassembly. Only multiples of 8 bytes are valid. 0x0008 8 bytes ... 0x0010 16 bytes ... 0x0018 24 bytes ... 0x0FF8 4088 bytes 0x1000 4096 bytes Byte counts greater than 256 are valid for multi-segment mode only. Number of segments is determined by the byte count divided by the segment size.

Figure 16-39 shows the supported inbound format for Type11 data management. The command descriptor address points directly to the start of the message descriptor. The data offset is specified through the command descriptor offset field. The data offset is programmable through the Type11 data offset register, IT11C*n*DOR, but must be greater than or equal to the descriptor size.

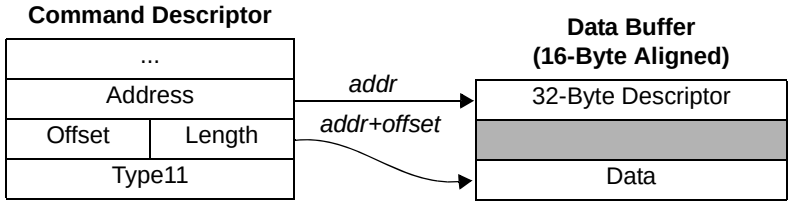


Figure 16-39. Command Descriptor Referencing a Type11 Inbound Message

16.3.10.3 Type11 Outbound Message Operation

The segmentation units are responsible for sending messages enqueued by a producer or producers from one of the outbound message queues owned by the message unit. Concurrent processing may also occur between any number of message queues as long as the outbound ordering rules apply, see **Section 16.3.14.2**, *Outbound Ordering Rules*.

An outbound segmentation unit can operate in multicast mode. In multicast mode, a single segment can be sent to multiple destinations.

16.3.10.3.1 Work Scheduling

The message unit receives all work from the outbound message queues. A producer or producers enqueue work onto the outbound message queues owned by the message unit. To maintain any

ordering between transactions, a producer must enqueue transactions with dependencies onto the same outbound message queue. The outbound ordering rules are described in **Section 16.3.14.2**, *Outbound Ordering Rules*.

16.3.10.3.2 Adding Messages to a Message Queue

A producer may add new messages to any message queue owned by the message unit. The producer needs to allocate buffer space for the message descriptor and data as appropriate and enqueue the command descriptor. The message unit dequeues the command descriptor, determines the type, and proceeds to read the message descriptor and data.

A producer should follow the guidelines below to add a new message to a message queue:

- Create the 32-byte outbound message descriptor as described in **Section 16.3.10.1**, *Type11 Outbound Message Descriptor Format*.
  - Set the descriptor type select field FTYPE=4b1011 to indicate message type.
  - Set the retry field to indicate number or times the message unit should attempt to retry the message segments before stopping.
  - For all other fields, see **Section 16.3.10.1**, *Type11 Outbound Message Descriptor Format*.
- Allocate a single buffer to hold the message descriptor and data.
- Add the buffer to a command descriptor and set FTYPE=4b1011 to indicate message type.
- Enqueue the command descriptor onto the message queue.
- The message unit releases buffers used by the producer to hold the message descriptor and data payload when the operation completes if the buffer release bit (BR) is set in the message descriptor.

16.3.10.3.3 Message Unit Initialization

To configure the message unit for message operations, see **Section 16.3.17.6**, *Initializing Outbound Message Queues*.

16.3.10.3.4 Error Handling

The message unit can report completion status and error information after completing the command descriptor by updating the command descriptor status field.

The following steps should be performed to enable this mode.

- Set the message descriptor completion (CS) or error (ES) status bit.
- Software should observe the status field of the command descriptor after it has been written to the completion queue to determine success.

Table 16-47 indicates the possible error conditions and the outcome when detected. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at

an error checking level. Once an error is detected, no additional error checking beyond the current error level is performed.

Table 16-47. Outbound Message Hardware Errors

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Comments
Message Request	Command descriptor effective length less than message descriptor payload byte count.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Message Request	Message descriptor byte count exceeds 256B when multicast mode specified (MM=1).	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Message Request	Invalid FTYPE.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Message Request	Data format (scatter/gather) not supported.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Message Request	Invalid flow level specified.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Message Request	Invalid target interface specified.	1	No	Descriptor error in command descriptor enqueued.	No	No buffers released regardless of BR setting.
Message Request	Memory transaction error.	1/2	Message unit error if MUIER[OTE] set	Transaction error in command descriptor enqueued. Message unit error detect register MUEDR[OTE]. Command descriptor captured in MUECCDR0–3.	No	Message aborted. No buffers released regardless of BR setting.
Message Response	Message error response.	2	No	Message response error in command descriptor enqueued. If MM=1, multicast error also set.	Yes	Message incomplete.

Table 16-47. Outbound Message Hardware Errors (Continued)

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Comments
Message Response	Number of segment retries exceeded.	2	No	Retry threshold exceeded in command descriptor enqueued. If MM=1, multicast error also set.	Yes	Message incomplete.
Message Response	Segment response timeout.	3	No	Segment response timeout in command descriptor enqueued. If MM=1, multicast error also set.	Yes	Message aborted.

16.3.10.3.4.1 Descriptor Error

When a descriptor error is detected by the message unit the following occurs.

- Completion message queue
  - The segmentation process is halted and the descriptor error bit DE in the command descriptor is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

16.3.10.3.4.2 Message Error Response Errors

When a message error response is received by the message unit the following occurs:

- Completion message queue
  - After the message operation completes, the error response received bit, ERR, in the command descriptor is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

16.3.10.3.4.3 Segment Response Time-Out Errors

When a segment response time-out is detected, as defined by the RapidIO port response time-out control CSR, for a message segment the following occurs.

- Completion message queue
  - After the message operation completes the segment response time-out error bit, SRT, in the command descriptor is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

16.3.10.3.4.4 Retry Threshold Exceeded Errors

When a retry threshold exceeded error is detected for a message segment the following occurs.

- Completion message queue
  - After the message operation completes the retry threshold exceeded bit, RTE, in the command descriptor is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

All segments, or targets for multicast, are transmitted even if the error is detected. Segments receiving retry are not retransmitted when the threshold is exceeded.

16.3.10.3.4.5 Multicast Errors

A multicast error is reported when operating in multicast mode, by setting bit MM=1 in the Destination Attribute descriptor field, and one of the following errors is reported by one or more destinations:

- Message Error Response (ERR)—one or more multicast destinations responded with error.
- Segment response time-out error (SRT)—one or more multicast destinations did not respond.
- Retry Threshold Exceeded (RTE)—one or more multicast destinations responded with retry.

When a multicast error is detected the following occurs.

- Completion message queue
  - After the message operation completes the multicast error bit, MCE, in the command descriptor is set.
  - The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

16.3.10.3.4.6 Transaction Errors

When an transaction error is detected during a local memory data read by the message unit the following occurs.

- Message segments that encounter a transaction error are not sent because the message data is not available.
- Memory reads that were already generated before the transaction error occurred are also not transferred. This includes retried message segments.
- Command descriptor captured in MUECCDR0–3.
- Global message unit error detect register MUEDR[OTE] set.
- Completion message queue
  - After the message operation completes the internal transaction error bit, TE, in the command descriptor is set.

— The message descriptor is enqueued onto the completion message queue if the message descriptor error status (ES) bit is set.

- Buffer release (BR=1)
  — No further buffers are released once a TE error is detected.

16.3.10.4 Type11 Inbound Message Operation

The inbound message unit is responsible for receiving messages and placing then onto an inbound message queue. Software can specify the destination message queue for an inbound message utilizing inbound message classification, see **Section 16.3.17.2**, *Classification Initialization*. The inbound message unit can receive segments of a message in any order.

Utilizing the buffer pools, the message unit requests a single buffer to hold the message descriptor and data. Software must provide the required sized buffer pool in IB*m*T11C*n*BPR for the message unit to store the entire message plus descriptor. The data offset is defined in IB*m*T11C*n*DOR. The message unit generates a frame size error if the inbound message can not be stored due to insufficient buffer sizes available.

16.3.10.4.1 Inbound Message Initialization

To configure an inbound classification unit for inbound message transactions, see **Section 16.3.17.5**, *Initializing Inbound Message Queues*.

16.3.10.4.2 Message Steering

Messages are forwarded to an inbound message queue based on the first message unit (numbered 1 through 64) to successfully classify the message based on source, mailbox, letter and extended mailbox fields. Message queue is specified by IB*m*T11C*n*MQR.

16.3.10.4.3 Retry Response Condition

The conditions to generate a logical layer retry response are:

- Message unit has detected a resource conflict, that is, a new message context was received and hardware context resources were depleted.

All responses are upgraded to priority 3.

16.3.10.4.4 Error Response Condition

The conditions to generate a logical layer error response are:

- Message unit did not have a matching classification rule.
- Message unit has detected a buffer size error.
- Message unit has detected a message format error.

- Message unit has detected a transaction error when allocating buffer resources or performing a local memory write.

All responses are upgraded to priority 3.

16.3.10.4.5 Error Handling

The message unit reports errors utilizing the inbound message queue, by setting the status field in the command descriptor before enqueueing the descriptor onto the designated inbound message queue. The consumer should read the status bits before processing the message to determine the appropriate action.

Table 16-48 indicates the possible error conditions and the outcome when detected. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected, no additional error checking beyond the current error level is performed. In general, when an error has been detected the reassembly continues until the end of the message is reached.

Table 16-48. Inbound Message Hardware Errors

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Data Segment Written	Message Descriptor Written	Response	Comments
Illegal RapidIO request priority 3.	1	Message unit error if MUIER[MFE] set.	Message unit error detect register MUEDR[MFE].	N/A	N/A	None	Transaction dropped.
Segment request timeout.	1	No	Segment Response Time-out in command descriptor enqueued.	No	Yes	None	Message is defective.
The RapidIO priority is not consistent for all message segments of a message.	1	No	Message format error in command descriptor enqueued.	No	Yes	Error	Message is defective.
Insufficient buffer pool size.	2	No	Buffer size error in command descriptor enqueued.	No	Yes	Error	Message is defective.
Duplicate segment received.	2	No	Message format error in command descriptor enqueued.	No	Yes	Error	Message is defective. Segment counted.

Table 16-48. Inbound Message Hardware Errors (Continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Data Segment Written	Message Descriptor Written	Response	Comments
Illegal segment size.	2	No	Message format error in command descriptor enqueued.	No	Yes	Error	Message is defective.
Segment size not consistent for all message segments of a message.	2	No	Message format error in command descriptor enqueued.	No	Yes	Error	Message is defective.
Memory transaction error.	3	Message unit error if MUIER[ITE] set.	Transaction error in command descriptor enqueued and message unit error detect register MUEDR[ITE]. Address captured in MUECAR.	No	No	Error	Message is defective.
Buffer pool exhausted.	3	Message unit error if MUIER[BAEIE] set.	Transaction error in command descriptor enqueued and message unit error detect register MUEDR[BAE].	No	No	Error	Message is defective.
Failure to enqueue the command descriptor onto the inbound message queue.	4	Message unit error if MUIER[IMERIE] set.	Message unit error detect register MUEDR[IMER]. Message queue captured in MUECMQR.	Yes	Yes	None	—

16.3.10.4.5.1 Segment Request Time-Out Errors

The segment request time-out counter starts after the first valid segment of a multi-segment message is received and the time-out counter is enabled. When a segment request time-out is detected, as defined by the RapidIO Port Response Time-out Control CSR, the following occurs.

- The message unit sets the segment request time-out error bit, SRT in the command descriptor.
- All message segments that have not yet been received are considered complete.
- The command descriptor is enqueued onto the message queue.

16.3.10.4.5.2 Buffer Size Errors

When a buffer size error is detected by the message unit, due to insufficient buffer pool size, the following occurs.

- Segment is discarded and an error response is returned for the segment.
- The message unit sets the buffer size error bit, BSE in the command descriptor.
- After the message operation completes, the command descriptor is enqueued onto the message queue.

The required buffer size for a message is based on the maximum byte count allowed for the given segment size, plus the descriptor size.

16.3.10.4.5.3 Message Format Errors

When a message format error is detected by the message unit the following occurs.

- Segment is discarded and an error response is returned for the segment.
- The message unit sets the message format error bit, MFE in the command descriptor.
- After the message operation completes, the command descriptor is enqueued to the message queue.

A duplicate segment counts towards the total segments for the message transaction. This results in the context closing when the total number of segments are received.

16.3.10.4.5.4 Transaction Errors

When an transaction error occurs, due to insufficient buffer resources or local memory write error, by the message unit the following occurs.

- Segment is discarded and an error response is returned for the segment.
- The message unit sets the transaction error bit, TE, in the command descriptor.
- After the message operation completes, the command descriptor is enqueued onto the message queue.
- Global message unit error detect register MUEDR[ITE] is set for memory access error.
- Address captured in MUECAR for memory access error.
- Global message unit error detect register MUEDR[BAE] is set if buffer resources depleted.

16.3.11 Session Management

The RapidIO Session Management Protocol Specification permits a session manager to establish, manage and remove connections between two end points. **Section 16.3.11.1**, *Classification*, describes how a session manager can establish connections with other end-points.

16.3.11.1 Classification

Classification is the process of filtering inbound traffic to a programmable message queue target. Each inbound classification unit can be configured to handle a specific type. The classification value/mask register pairs are used as rule filters for accepting or rejecting incoming traffic of the programmed RapidIO type. Flow specific reassembly context resources can be allocated for session management to avoid retries in a loaded system due to exhausted hardware resources.

The Session Management Protocol may use the classification feature to establish communication with other selected end-points and to manage communication channels. The protocol allows an end-point to advertise the conveyance used for establishing a connection through the Session Management Advertising CSR. The classification rule filters should then be initialized to match the CSR conveyance type and attributes.

Figure 16-15 shows an example of traffic classification, where packets of Type8–11 are steered to the correct inbound unit for processing. If there are any aliasing between units, the lower numbered unit wins. A transaction not classified is subject to error response or dropped.

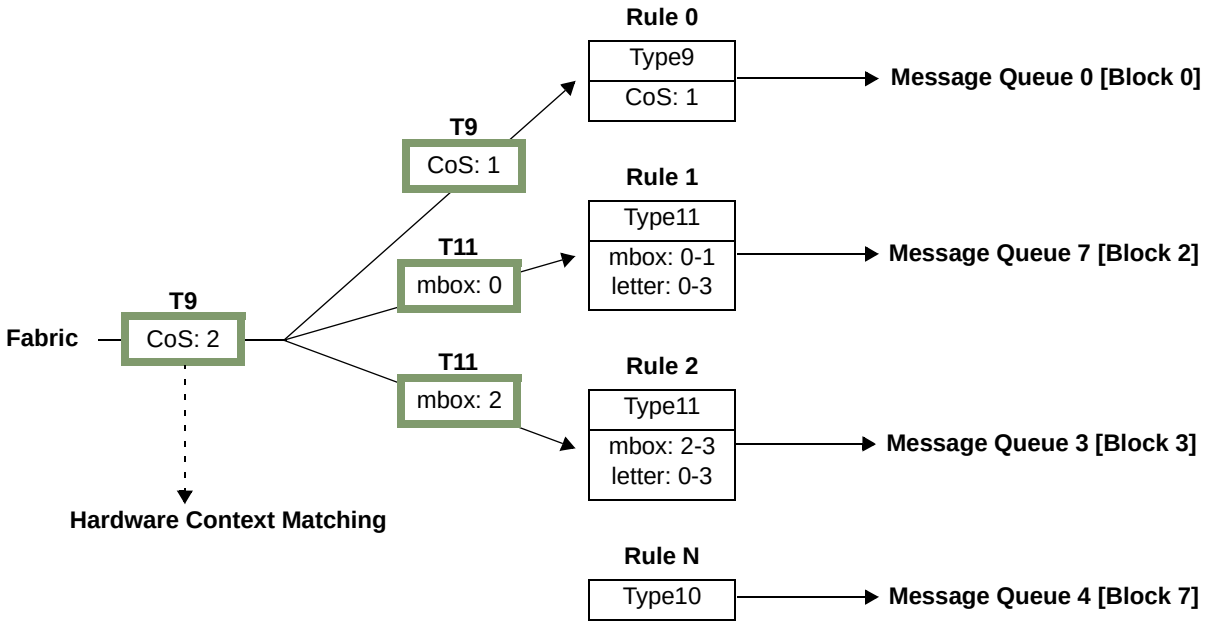


Figure 16-40. Inbound Traffic Classification

16.3.12 Hardware Context Management

The RapidIO specification defines three types of messages: Type9 data streaming, Type10 doorbells and Type11 data messages. A Type10 doorbell carries a 16-bit info field but no data and is always a single RapidIO packet. When a packet carries a data payload, the maximum size of this payload is 256 bytes.

Type9 and Type11 RapidIO transaction types define a user PDU larger than 256 bytes. They support messages with maximum PDUs of 64 Kbytes and 4 Kbytes respectively. To carry larger payloads, the protocol supports hardware segmentation and reassembly of the message payload. The RapidIO specification requires that Type10 doorbell messages and each Type11 message segment be logically acknowledged (ACK) by the receiver. When the receiver cannot temporarily accept the packet, the endpoint may send a retry ACK to the sender causing the segment to be resent again. If an error occurs, an error ACK may be sent. This logical ACK is distinct from the link-level acknowledge that occurs as the transaction crosses each link in the network.

16.3.12.1 Segmentation and Reassembly

Segmenting outbound messages is relatively straightforward in hardware. However, inbound reassembly is significantly more complex. As each segment arrives, it must be identified as either the first segment of a new message or a segment from an existing one. When the first segment of a new message arrives, hardware resources must be allocated to track and reassemble the message. Because Type11 message segments can arrive out of order, the hardware must also be capable of reordering the segments in memory back to their original order.

To maximize system performance, inbound messaging hardware should allow multiple messages from one or more sources to be received at the same time. This implies that individual segments from multiple messages are interleaved in arbitrary ways. RapidIO packet headers allow hardware to distinguish one message segment from another and reassemble similar segments into their respective messages. When the first segment of a new message arrives, a new reassembly context begins that is different from all other messages being reassembled.

As a message is transmitted segment by segment, a corresponding hardware reassembly resource (or context) is used at the destination. Due to the complexity of this hardware, the number of contexts that can be implemented is usually limited. This limitation defines the maximum number of simultaneous outstanding messages that may be in-flight to a given destination endpoint. Software can determine the number of contexts supported by any endpoint by accessing the data streaming information CAR register.

eMSG implements 24 reassembly contexts (or hardware contexts). These contexts are used by Type9–11 transaction types under various circumstances. The message unit also allows reassembly contexts to be allocated to specific flows using appropriate threshold registers. Messages assigned to flows without an allocation, draw from a generic pool set by the generic threshold register.

If the first segment of a new message arrives and is properly classified (that is, a classification rule matches) but all allocated hardware contexts are busy reassembling other messages, the behavior of the inbound hardware depends on the transaction type. Type11 transactions cause hardware to issue a logical retry, with the expectation that resources will become available soon.

Type9 transactions do not require logical ACKs and hence there is no mechanism for communicating an error or retry back to the receiver. As a result, Type9 packets are dropped.

Hardware contexts within eMSG are shared by Type9, 10 and 11 messages and allocation can not be made by transaction type. To avoid Type9 packet loss, system designers must ensure that for each endpoint the maximum number of simultaneous inbound interleaved messages of all types never exceed the pre-allocated hardware contexts.In the eMSG, each hardware context entry has the following states:

- Invalid
  - Empty entry
- Open
  - Multi-segment Type9 or 11 message
  - Not all message segments have been received
- Closed
  - Single-segment Type9–11 message
  - Multi-segment Type9 or 11 message after all segments of the message have been received
  - Message is now guaranteed to complete
  - Associated command descriptor update has not yet been issued to memory

When a single-segment Type9–11 message arrives and is classified, an invalid hardware context is allocated and its state is moved from Invalid to Closed. When hardware issues the command descriptor to memory, the hardware context state moves from Closed to Invalid.

When a new Type9 or 11 multi-segment message first arrives, an invalid hardware context is allocated and its state is moved from Invalid to Open. The entry is moved from Open to Closed when all segments of a message have been classified. Because single-segment messages move directly from Invalid to Closed, the number of hardware contexts in the Open state equals the total number of outstanding multi-segment messages.

Open hardware contexts may be allocated to specific RapidIO flow levels. A flow with no hardware contexts allocated shares the generic pool. hardware contexts remaining after per flow and generic pool are in the unassigned pool. Hardware maintains a counter per flow as well as a generic counter to track the number of Open and unassigned contexts. Counters are incremented when the first segment of a multi-segment Type9 or 11 message is classified and decremented when the last segment of a message is classified. If a counter reaches a configurable watermark threshold, no further new single- or multi-segment messages at that flow level are accepted, and subsequent Type11 messages are logically retried and Type9 messages are dropped. In all situations, Type9 packets are dropped only if the number of interleaved messages exceeds the allocation budgeted by the system.

Should the eMSG encounter unusual system congestion that significantly increases the latency of memory, inbound message segments begin to accumulate in internal buffers and eventually result in link-level retries as back-pressure is applied. Middle segments are buffered until space is no longer available after which link-level retries occur. End segments are also buffered and eventually cause Open hardware contexts to transition to Closed. First segments of messages are accepted while there are Invalid hardware contexts and their associated flow/generic counter has not reached the programmed threshold.

Under normal operating conditions, the eMSG hardware operates such that when the final segment of a message is followed by the first segment of a new one, the hardware context used by the first message is reused for the second. This is because the last (or only) segment causes the hardware context to transition quickly to Invalid as memory writes are issued. As a result of this reuse, one or more streams of single or multi-segment messages whose message segments are never interleaved require one reassembly context to be allocated.

For example, an inbound sequence of multiple single segment Type9 or 11 message streams must be allocated only a single reassembly context. However, each single-segment message still constitutes a new hardware context in terms of context interleaving and per-flow thresholds. As a result, hardware still checks the flow level of the message against the associated open thresholds. If that threshold was reached before it arrives, the single-segment message represents a new interleaved context beyond the designated allocation. As a result, a Type11 packet is retried and a Type9 packet is dropped.

An accepted single segment message never increments Open counters and hence cannot push Open counters to a threshold. Therefore, single-segment messages which are not part of flows that have already reached threshold are accepted until Invalid hardware contexts are exhausted or internal buffers fill. Once filled, link-level retry is invoked as back-pressure to avoid Type9 packet loss.

The unassigned pool makes available hardware contexts even when all allocated ones are in use. When single segment Type9 messages arrive while memory congestion prevents release of previously Closed contexts, hardware contexts from the unassigned pool are used instead. In this manner, the unassigned pool serves to manage congestion by delaying the invocation of link-level retry during short memory congestion events.

To ensure maximum performance when using Type9 traffic, it is recommended that eight hardware contexts be allocated to the congestion management (unassigned) pool to cover rare worst-case short-term congestion events. This leaves 16 entries for general system use. Failure to allocate eight for this purpose does not result in Type9 packet loss, but may cause more frequent link-level retries. Note that these retries can cause brief performance loss for higher priority streams carried by the link.

16.3.12.2 Examples

The following are some examples of system configurations and the resulting behavior. A stream as used below is defined as a sequence of messages sharing the same tuple, as follows:

- Type9: <SrcID, Destid, CoS, StreamID>
- Type11: <SrcID, DestID, MBox/XMBox, Letter>

Example 16-1.

- Inbound Traffic Types
  - 23 segment interleaved multi-segment Type11 streams over Flow A
- Context allocation
  - 23 for Flow A
  - 1 for congestion management
- Thresholds
  - Flow A = 23
  - Generic = 0
- Endpoint behavior
  - All segments of Type11 message streams 1–23 always accepted
  - Even though only 1 context is allocated for congestion management, configuration is tolerant of memory congestion because Type11 multi-segment streams are more tolerant of congestion
  - Memory congestion beyond tolerance could eventually result in Type11 logical retries, link-level retries or both depending on sequence of events and timing

Example 16-2.

- Inbound Traffic Types
  - 30 segment interleaved multi-segment Type11 streams over Flow A
- Context allocation
  - 23 for Flow A
  - 1 for congestion management
- Thresholds
  - Flow A = 23
  - Generic = 0
- Endpoint behavior
  - Messages from streams 1–30 may be accepted depending on the order of the message and segments
  - Segments of only 23 messages are accepted at any given moment.
  - All others are logically retried.
  - No control at the receive endpoint over which streams are accepted and which are retried is possible.

- Even though only 1 context is allocated for congestion management, configuration is tolerant of memory congestion because Type11 multi-segment streams are more tolerant of congestion
- Memory congestion beyond tolerance eventually results in Type11 logical retries, link-level retries or both.

Example 16-3.

- Inbound Traffic Types
  - 23 segment interleaved multi-segment Type9 streams over Flow A
- Context allocation
  - 23 for Flow A
  - 1 for congestion management
- Thresholds
  - Flow A = 23
  - Generic = 0
- Endpoint behavior
  - All segments of Type9 message streams 1–23 are always accepted.
  - Even though only 1 context is allocated for congestion management, configuration is tolerant of memory congestion because multi-segment streams are more tolerant of congestion.
  - Memory congestion beyond tolerance eventually results in link-level retries.

Example 16-4.

- Inbound Traffic Types
  - 30 segment interleaved multi-segment Type9 streams over Flow A
- Context allocation
  - 23 for Flow A
  - 1 for congestion management
- Thresholds
  - Flow A = 23
  - Generic = 0
- Endpoint behavior
  - Various messages from Type9 streams 1–30 may be accepted depending on the order of the message and segments
  - No more than 23 messages are reassembled at one time.
  - All others messages are dropped.
  - No control at the receive endpoint over which streams are accepted and which are dropped is possible.

Example 16-5.

- Inbound Traffic Types
  - 16 segment interleaved multi-segment Type11 streams over Flow A
  - 1 single-segment Type9 stream over Flow A
- Context allocation
  - 16 for Flow A
  - 8 for congestion management
- Thresholds
  - Flow A = 16
  - Generic = 0
- Endpoint behavior
  - All segments of Type11 message streams 1–16 always accepted
  - Type9 segments are dropped if they arrive when all 16 Type11 streams have outstanding segments (that is, all 16 contexts are in Open state thus causing the threshold for flow A to be reached)

Example 16-6.

- Inbound Traffic Types
  - 8 interleaved single-segment Type9 streams over Flow A
  - 8 interleaved single-segment Type9 streams over Flow B
- Context allocation
  - 8 for Flow A
  - 8 for Flow B
  - 8 for congestion management
- Thresholds
  - Flow A = 8
  - Flow B = 8
  - Generic = 0
- Endpoint behavior
  - All segments of Type9 message streams 1–8 always accepted over Flow A
  - All segments of Type9 message streams 1–8 always accepted over Flow B
  - Tolerant of memory congestion because recommended 8 entries are allocated for congestion management
  - Memory congestion beyond tolerance eventually results in link-level retries.

Example 16-7.

- Inbound Traffic Types
  - 15 segment interleaved multi-segment Type11 streams over Flow A
  - 16 single segment Type9 streams over Flow B

- Context allocation
  - 15 for Flow A
  - 1 for Flow B
  - 8 for congestion management
- Thresholds
  - Flow A = 15
  - Flow B = 1
  - Generic = 0
- Endpoint behavior
  - All segments of Type11 message streams 1–15 always accepted
  - All segments of Type9 message streams 1–16 always accepted
  - Only a single context is required for one or more streams of single segment messages
  - Tolerant of memory congestion because recommended 8 entries are allocated for congestion management
  - Memory congestion beyond tolerance eventually results in Type11 logical retries, link-level retries or both.

Example 16-8.

- Inbound Traffic Types
  - 11 segment interleaved Type11 streams over Flow A where each stream is a mix of single and multi-segment messages
  - 5 segment interleaved Type9 streams over Flow B where each stream is a mix of single and multi-segment messages
- Context allocation
  - 11 for Flow A
  - 5 for Flow B
  - 8 for congestion management
- Thresholds
  - Flow A = 11
  - Flow B = 5
  - Generic = 0
- Endpoint behavior
  - All segments of Type11 message streams 1–11 always accepted
- All segments of Type9 message streams 1–5 always accepted
  - Tolerant of memory congestion because recommended 8 entries are allocated for congestion management
  - Memory congestion beyond tolerance eventually results in Type11 logical retries, link-level retries or both.

16.3.13 Address Alignment Requirements

The inbound and outbound address alignment requirements are as follows:

- Inbound—Software must align scatter/gather tables and message descriptors to a 16 byte address boundary. This requirement forces all buffer pools programmed for inbound reassembly to supply 16 byte aligned buffer addresses.
- Outbound—No address alignment restrictions exist for scatter/gather table, message descriptor or data payload.

16.3.14 Ordering Rules

The RapidIO fabric implies no ordering rules between different types of transactions such as doorbells and messages. The transaction source must use priorities to ensure in-order delivery in these cases. The following sections describe the ordering rules applied by the message unit.

16.3.14.1 Inbound Ordering Rules

- Inbound Type9 Packets
  - Type9 packets (PDUs) with the same Virtual Stream ID (Port + Source ID + Destination ID + Class-of-Service + Flow) are enqueued to the assigned message queue in order of delivery.
  - Type9 packets (PDUs) with different Virtual Stream IDs may be enqueue out of order if assigned to the same message queue.
  - There are no ordering rules between PDUs assigned to different message queues.
- Inbound Type10 Doorbells
  - There are no implied ordering rules between doorbells.
- Inbound Type11 Messages
  - There are no implied ordering rules between messages.
- Inbound mixed
  - A Type10 doorbell following a Type9 PDU or Type11 message from the same source ID, same destination ID, same or lower priority and assigned to the same message queue, is guaranteed to be enqueued in order.
  - There are no implied ordering rules between messages and PDUs.

16.3.14.2 Outbound Ordering Rules

- Outbound Type5 writes without response
  - NWrites with the same Source ID + Destination ID + Flow are transmitted in order, if from the same message queue.
- Outbound Type6 streaming writes
  - SWrites with the same Source ID + Destination ID + Flow are transmitted in order, if from the same message queue.

- Outbound Type8 port-writes
  - There are no implied ordering rules between port-writes.
- Outbound Type9 PDUs
  - Type9 packets (PDUs) with the same Virtual Stream ID (Source ID + Destination ID + Class-of-Service + Flow) are transmitted in order if from the same message queue. PDUs of the same VSID should not be occupying two different transmit queues to avoid reassembly errors at the destination.
  - Type9 packets (PDUs) with different Virtual Stream IDs may be transmitted out of order regardless of the message queue from which it was dequeued.
- Outbound Type10 doorbells
  - Doorbells with the same Source ID + Destination ID + Flow are transmitted in order, regardless of the message queue.
- Outbound Type11 messages
  - Type11 messages with the same Source ID + Destination ID + Mailbox + Letter are transmitted in order, regardless of the message queue.
  - For multicast messages, all possible destination IDs are considered for dependency.
  - A single segment message have no ordering requirements on another single segment message when the extended mailbox number is unique.
- Outbound mixed
  - A Type10 doorbell following a Type9–11 transaction with same or lower priority and same destination ID, are transmitted in order if the doorbell descriptor SO (strict ordering) bit is set.
  - There are no implied ordering rules between messages and PDUs.
  - NWrites and SWrites of the same Source ID + Destination ID + Flow are transmitted in order, if from the same message queue.

16.3.14.3 Outbound Segmentation Interleaving

Assigning the same RapidIO flow level to transactions in different arbitration groups is not allowed. All transactions in a particular arbitration group should use the same RapidIO flow level and each arbitration group should use a unique flow level. Failure to follow these rules may result in illegal interleaving of Type9 messages whose VSID differs only in streamID and result in undefined reassembly at the destination.

To achieve a higher transmit rate, the outbound segmentation units may interleave segments when no order dependencies exists, as defined by the outbound ordering rules, between two or more transactions. While this may increase the performance on the transmit side, additional hardware reassembly resources may be needed on the receive side. Outbound segmentation interleaving can be controlled per destination by setting the register bit MUMR[OSI].

16.3.14.4 Transaction Priorities

The message unit uses the following transaction priorities for read and write transactions to and from the memory sub-system, see **Table 16-49**.

Table 16-49. Memory Read/Write Priorities

Memory Transaction	Priority
Memory write of data payload	RapidIO priority
Memory write of message descriptor	RapidIO priority
Memory read of scatter/gather table	2
Memory read of message descriptor	2
Memory read of data payload	Arbitration group bits [0:1]

16.3.15 Congestion Management

Congestion can occur within the message unit for several reason, this includes:

- Congestion due to slow memory subsystem resulting in the inbound dispatch queue to fill.
  - Physical retry is performed
- Congestion due to context availability, or lack thereof.
  - Transactions are accepted speculatively, meaning without a hardware context, up to a certain threshold after which logical retry occurs for Type10/11 and Type9 is dropped.
- Congestion due to buffer availability
  - Reassembly terminated with error, subsequent segments dropped.
- Congestion due to target message queue full.
  - Critical watermark stops further delivery of traffic to the queue.

16.3.15.1 Critical Flow Control

The message unit supports a built-in critical flow control mechanism that guarantees no inbound queue overflows. Operating at a near full queue (30 entries remaining) triggers critical flow control, disabling the receipt of new transactions. This flow control mechanism results in physical retry for subsequent transactions when the enter watermark has been reached as indicated by the classification unit status bit IBmTtCnSR[XOFF] = 1. To exit the critical flow level condition, software should process transactions in the inbound queue to bring the level below the exit watermark as specified by IB*m*MQ*n*CMR[EXIT_WM]. The classification rule could also be disabled to clear the condition, but it is not recommended unless the queue level is also dealt with.

16.3.16 Interrupts

The message unit only generates one global error interrupt. The interrupt error detect from register MUEDR is qualified with its corresponding interrupt enable from register MUIER. The result of each is combined to form one global error interrupt. **Figure 16-41** shows the interrupt structure.

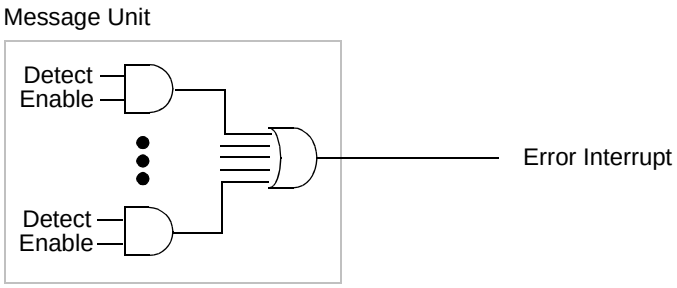


Figure 16-41. Interrupt Structure

16.3.17 Initialization Information

The message unit needs to be initialized properly to be able start processing transactions. The following sections describe the setup for:

- Global register setup—error detection, interrupts and flow control
- Inbound classification rules and inbound queues
- Outbound queues

16.3.17.1 Initializing the Global Registers

The following steps are recommended for correct device initialization:

1. Clear the error detect register, see **Section 16.4.2.96**, *Message Unit Error Detect Registers (MUEDR).*, by writing a 1 to the fields. If not cleared, incorrect error status information is indicated.
2. Initialize stream management flow control through register DSLLCCSR for Type9 data streaming.

16.3.17.2 Classification Initialization

There are many ways in which software can interact with the inbound classification units. One method to initialize the message unit to use one or more inbound units for classification is as follows:

- Disable the unit, if enabled. Any pending transactions are completed. Poll the status register busy bit to make sure the unit is not busy with any previously initiated transactions.
- Initialize the inbound Buffer Pool ID and data offset registers, for all types, used to build the scatter/gather tables and data structures in memory for inbound transactions.

- Type8—Port-Write
  - Set the classification value/mask registers IB*m*T8C*n*RVR0 and IB*m*T8C*n*RMR0 to accept the selected inbound port-write.
  - Set the message queue target register, IB*m*T8C*n*MQR.
  - Set the inbound unit type, IB*m*T9C*n*MR[FTYPE]=4b1000.
  - Enable the classification rule, IB*m*T8C*n*MR[CUE]=1 after the target queue has been initialized.
- Type9—Data Streaming Reassembly
  - Set the classification value/mask registers IB*m*T9C*n*RVR0/1 and IB*m*T9C*n*RMR0/1 to accept the selected inbound PDUs.
  - Set the message queue target register, IB*m*T9C*n*MQR.
  - Set the inbound unit type, IB*m*T9C*n*MR[FTYPE]=4b1001 and IB*m*T9C*n*MR[EXT]=0.
  - Set the inbound flow control destination register, IB*m*T9C*n*FCDR, if flow control is enabled DSLLCCSR[TME]=1
  - Enable the classification rule, IB*m*T9C*n*MR[CUE]=1 after the target queue has been initialized.
- Type9 Extended—Outbound Traffic Management
  - Set the classification value/mask registers IB*m*T9C*n*RVR0/1 and IB*m*T9C*n*RMR0/1 to accept the required data streaming flow control messages.
  - Set the message queue target register, IB*m*T9C*n*MQR.
  - Set the inbound unit type, IB*m*T9C*n*MR[FTYPE]=4b1001 and IB*m*T9C*n*MR[EXT]=1.
  - Enable the classification rule, IB*m*T9C*n*MR[CUE]=1.
- Type10—Doorbell
  - Set the classification value/mask registers IB*m*T10C*n*RVR0/1 and IB*m*T10C*n*RMR0/1 to accept the selected inbound doorbells.
  - Set the message queue target register IB*m*T10C*n*MQR.
  - Set the inbound unit type, IB*m*T10C*n*MR[FTYPE]=4b1010.
  - Enable the classification rule, IB*m*T10C*n*MR[CUE]=1 after the target queue has been initialized.
- Type11—Message
  - Set the classification value/mask registers IB*m*T11C*n*RVR0/1 and IB*m*T11C*n*RMR0/1 to accept the selected inbound messages
  - Set the message queue target register IB*m*T11C*n*MQR.
  - Set the inbound unit type, IB*m*T11C*n*MR[FTYPE]=4b1011.
  - Enable the classification rule, IB*m*T11C*n*MR[CUE]=1 after the target queue has been initialized.

16.3.17.3 Dynamically Changing Rules

Software may wish to dynamically reassign a classification rule to a previously initialized inbound target queue. If the rule is currently not in use, it may be enabled without further action.

To modify an active inbound rule, software should perform the following actions:

1. Stop all sources of traffic before disabling the classification rule to avoid loss of transactions. Software must also ensure that any pending transaction associated with the rule has completed. This should be handled by the session management protocol and is beyond the scope of this discussion.
2. Disable the rule. For Type9 rules with congestion management enabled, software should make sure that both the source and destination reflect the same consistent flow control setting. It is recommended that the inbound target queue be in an XON state before disabling the rule.
3. Modify and then enable the rule for an existing active inbound queue.
4. Software communicates to the source that traffic can start for the new rule using the session management protocol.

16.3.17.4 Mixing Classification Rule Types

Multiple classification rules of a different RapidIO type may target the same inbound queue. It should be noted that only classification rules of Type9 extended can generate flow control. The source must be able to respond to a single Type9 flow control for congestion management to work properly, or the inbound queue may fill with non-Type9 transactions. The source endpoint could have a single mixed mode outbound queue generating all traffic.

16.3.17.5 Initializing Inbound Message Queues

There are many ways in which software can interact with the inbound message queues. One method to initialize the message queues is as follows:

- Disable all classification rules targeting the inbound queue, IB*m*T[8–11]C*n*MR[CUE], if enabled. See **Section 16.3.17.3**, *Dynamically Changing Rules*.
- Disable the inbound message queue, IB*m*MQ*n*MR[EN]=0.
- Clear the error detect registers, IB*m*MQ*n*EDR and IB*m*MQ*n*MEDR.
- Set the inbound message queue size, IB*m*MQ*n*MR[CQ_SIZE], field.
- Initialize the inbound message queue pointers, IB*m*MQ*n*(E)DPAR and IB*m*MQ*n*(E)EPAR to match.
- Initialize the flow control watermark levels, IB*m*MQ*n*CMR. For data streaming, flow control must be enabled (DSLLCCSR[TME] = 1) to generate outbound flow control messages.

- Enable the inbound message queue, IB*m*MQ*n*MR[EN] = 1.
- Follow the classification initialization sequence, see **Section 16.3.17.2**, *Classification Initialization*.

16.3.17.6 Initializing Outbound Message Queues

There are many ways in which software can interact with an outbound message queue. One method to initialize an outbound message queue is as follows:

- Disable the outbound message queue, OB*m*MQ*n*MR[EN]=0. Poll the status register busy bit, OB*m*MQ*n*SR[MQB] to make sure all completion results have been received.
- Clear the error detect registers OB*m*MQ*n*(M)EDR.
- Configure the arbitration group number of message queues using configuration register OMQDSCR1. Queue arbitration weight per AG is configured using OMQDSCR2.
- Set the message queue size, OB*m*MQ*n*MR[CQ_SIZE], field.
- Initialize the outbound message queue pointers, OB*m*MQ*n*(E)DPAR and OB*m*MQ*n*(E)EPAR to match.
  - The outbound message queue detects address pointer wrap conditions only when enabled. If the pointers are unequal when the queue is enabled, no wrap condition must exist.
- Enable the outbound message queue, OB*m*MQ*n*MR[EN] = 1.
- Enable the message unit, OB*m*MU*n*MR[MUE] = 1. This enables the unit to start dequeueing from the outbound message queue as soon as work is available (message queue pointers not equal).

16.4 RapidIO/eMSG Programming Model

The RapidIO registers include:

- Architectural Registers:
  - Device Identity Capability Register (DIDCAR), **page 16-181**
  - Device Information Capability Register (DICAR), **page 16-182**
  - Assembly Identity Capability Register (AIDCAR), **page 16-182**
  - Assembly Information Capability Register (AICAR), **page 16-183**
  - Processing Element Features Capability Register (PEFCAR), **page 16-184**
  - Source Operations Capability Register (SOCAR), **page 16-185**
  - Destination Operations Capability Register (DOCAR), **page 16-186**
  - Data Streaming Information Capability Register (DSICAR), **page 16-188**
  - Data Streaming Logical Layer Control Command and Status Register (DSLLCCSR), **page 16-189**
  - Processing Element Logical Layer Control Command and Status Register (PELLCCSR), **page 16-190**

- Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR), **page 16-190**
- Base Device ID Command and Status Register (BDIDCSR), **page 16-191**
- Host Base Device ID Lock Command and Status Register (HBDIDLCSR), **page 16-192**
- Component Tag Command and Status Register (CTCSR), **page 16-192**

- Extended Features Space: x1/x2/x4 LP-Serial
  - Port Maintenance Block Header 0 (PMBH0), **page 16-193**
  - Port Link Time-Out Control Command and Status Register (PLTOCCSR), **page 16-194**
  - Port Response Time-out Control Command and Status Register (PRTOCCSR), **page 16-194**
  - Port General Control Command and Status Register (PGCCSR), **page 16-195**
  - Port 1–2 Link Maintenance Request Command and Status Register (PnLMREQCSR), **page 16-196**
  - Port 1–2 Link Maintenance Response Command and Status Register (PnLMRESPCSR), **page 16-197**
  - Port 1–2 Local ackID Status Command and Status Register (PnLASCR), **page 16-198**
  - Port 1–2 Error and Status Command and Status Register (PnESCSR), **page 16-199**
  - Port 1–2 Control Command and Status Register (PnCCSR), **page 16-200**
- Extended Features Space: Error Reporting, Logical
  - Error Reporting Block Header (ERBH), **page 16-203**
  - Logical/Transport Layer Error Detect Command and Status Register (LTLEDCSR), **page 16-203**
  - Logical/Transport Layer Error Enable Command and Status Register (LTLEECSR), **page 16-205**
  - Logical/Transport Layer Address Capture Command and Status Register (LTLACCSR), **page 16-207**
  - Logical/Transport Layer Device ID Capture Command and Status Register (LTLDIDCCSR), **page 16-208**
  - Logical/Transport Layer Control Capture Command and Status Register (LTLCCCSR), **page 16-209**
- Extended Features Space: Error Reporting, Physical
  - Port 1–2 Error Detect Command and Status Register (PnEDCSR), **page 16-210**
  - Port 1–2 Error Rate Enable Command and Status Register (PnERECSR), **page 16-211**
  - Port 1–2 Error Capture Attributes Command and Status Register (PnECACSR), **page 16-212**
  - Port 1–2 Packet/Control Symbol Error Capture Command and Status Register (PnPCSECCSR0), **page 16-213**

- ATMU
  - Port 1–2 RapidIO Outbound Window Translation Address Register (PnROWTARx), **page 16-238**
  - Port 1–2 RapidIO Outbound Window Translation Extended Address Register (PnROWTEARx), **page 16-239**
  - Port 1–2 RapidIO Outbound Window Base Address Register (PnROWBARx), **page 16-240**
  - Port 1–2 RapidIO Outbound Window Attributes Register (PnROWARx), **page 16-241**
  - Port 1–2 RapidIO Outbound Window Segment 1–3 Registers (PnROWSxRy), **page 16-243**
  - Port 1–2 RapidIO Inbound Window Translation Address Registers (PnRIWTARx); **page 16-244**
  - Port 1–2 RapidIO Inbound Window Base Address Registers (PnRIWBARx), **page 16-244**
  - Port 1–2 RapidIO Inbound Window Attributes Registers (PnRIWARx), **page 16-245**

The eMSG registers (including the BMLite and QMLite registers) include the following:

- Inbound Classification Unit Registers (for Block m [0–7] Unit n [0–7])
  - Type8 Registers (IB*m*T8C*n*MR[FTYPE]=0b1000)
    - Inbound Block m Type8 Classification Unit n Mode Register (IBmT8CnMR), **page 16-247**
    - Inbound Block m Type8 Classification Unit n Status Register (IBmT8CnSR), **page 16-248**
    - Inbound Block m Type8 Classification Unit n Message Queue Register (IBmT8CnMQR), **page 16-248**
    - Inbound Block m Type8 Classification Unit n Rule Value Register 0 (IBmT8CnRVR0), **page 16-249**
    - Inbound Block m Type8 Classification Unit n Rule Value Register 1 (IBmT8CnRVR1), **page 16-249**
    - Inbound Block m Type8 Classification Unit n Rule Mask Register 0 (IBmT8CnRMR0), **page 16-251**
    - Inbound Block m Type8 Classification Unit n Rule Mask Register 1 (IBmT8CnRMR1), **page 16-251**
    - Inbound Block m Type8 Classification Unit n Data Buffer Pool Register (IBmT8CnDBPR), **page 16-253**
    - Inbound Block m Type8 Classification Unit n Data Offset Register (IBmT8CnDOR), **page 16-255**
  - Type9 Registers (IB*m*T9C*n*MR[FTYPE]=0b1001

- Inbound Block m Type9 Classification Unit n Mode Register (IBmT9CnMR), **page 16-255**
- Inbound Block m Type9 Classification Unit n Status Register (IBmT9CnSR), **page 16-256**
- Inbound Block m Type9 Classification Unit n Message Queue Register (IBmT9CnMQR, **page 16-257**
- Inbound Block m Type9 Classification Unit n Rule Value Register 0 (IBmT9CnRVR0), **page 16-258**
- Inbound Block m Type9 Classification Unit n Rule Value Register 1 (IBmT9CnRVR1), **page 16-258**
- Inbound Block m Type9 Classification Unit n Rule Mask Register 0 (IBmT9CnRMR0), **page 16-259**
- Inbound Block m Type9 Classification Unit n Rule Mask Register 1 (IBmT9CnRMR1), **page 16-259**
- Inbound Block m Type9 Classification Unit n Flow Control Destination Register (IBmT9CnFCDR), **page 16-261**
- Inbound Block m Type9 Classification Unit n Data Buffer Pool Register (IBmT9CnDBPR), **page 16-262**
- Inbound Block m Type9 Classification Unit n Data Offset Register (IBmT9CnDOR), **page 16-263**
- Inbound Block m Type9 Classification Unit n Scatter/Gather Buffer Pool Register (IBmT9CnGBPR), **page 16-264**

— Type10 Registers (IB*m*T10C*n*MR[FTYPE]=0b1010)

- Inbound Block m Type10 Classification Unit n Mode Register (IBmT10CnMR), **page 16-265**
- Inbound Block m Type10 Classification Unit n Status Register (IBmT10CnSR), **page 16-266**
- Inbound Block m Type10 Classification Unit n Message Queue Register (IBmT10CnMQR, **page 16-267**
- Inbound Block m Type10 Classification Unit n Rule Value Register 0 (IBmT10CnRVR0), **page 16-267**
- Inbound Block m Type10 Classification Unit n Rule Value Register 1 (IBmT10CnRVR1), **page 16-267**
- Inbound Block m Type10 Classification Unit n Rule Mask Register 0 (IBmT10CnRMR0), **page 16-269**
- Inbound Block m Type10 Classification Unit n Rule Mask Register 1 (IBmT10CnRMR1), **page 16-269**

- Inbound Block m Type10 Classification Unit n Data Buffer Pool Register (IBmT10CnDBPR), **page 16-271**
- Inbound Block m Type10 Classification Unit n Data Offset Register (IBmT10CnDOR), **page 16-272**

— Type11 Registers (IB*m*T11C*n*MR[FTYPE]=0b1011)

- Inbound Block m Type11 Classification Unit n Mode Register (IBmT11CnMR), **page 16-273**
- Inbound Block m Type11 Classification Unit n Status Register (IBmT11CnSR), **page 16-275**
- Inbound Block m Type11 Classification Unit n Message Queue Register (IBmT11CnMQR, **page 16-275**
- Inbound Block m Type11 Classification Unit n Rule Value Register 0 (IBmT11CnRVR0), **page 16-276**
- Inbound Block m Type11 Classification Unit n Rule Value Register 1 (IBmT11CnRVR1), **page 16-276**
- Inbound Block m Type11 Classification Unit n Rule Mask Register 0 (IBmT11CnRMR0), **page 16-278**
- Inbound Block m Type11 Classification Unit n Rule Mask Register 1 (IBmT11CnRMR1), **page 16-278**
- Inbound Block m Type11 Classification Unit n Data Buffer Pool Register (IBmT11CnDBPR), **page 16-280**
- Inbound Block m Type11 Classification Unit n Data Offset Register (IBmT11CnDOR), **page 16-281**

■ Queue Manager Inbound Block Message Queue Registers (for Block m [0–7] Queue n [0–7])

— Inbound Block m Message Queue n Mode Register (IBmMQnMR), **page 16-281**
— Inbound Block m Message Queue n Status Register (IBmMQnSR), **page 16-282**
— Inbound Block m Message Queue n Dequeue Pointer Address Register (IBmMQnDPAR), **page 16-283**
— Inbound Block m Message Queue n Enqueue Pointer Address Register (IBmMQnEPAR), **page 16-284**
— Inbound Block m Message Queue n Congestion Management Register (IBmMQnCMR), **page 16-285**
— Inbound Block m Message Queue n Interrupt Enable Register (IBmMQnIER), **page 16-286**
— Inbound Block m Message Queue n Interrupt Detect Register (IBmMQnIDR), **page 16-288**
— Inbound Block m Message Queue n Interrupt Coalescing Register (IBmMQnICR), **page 16-289**

- Queue Manager Outbound Block Message Queue Registers (for Block m [0–7] Queue n [0–7])
  - Outbound Block m Message Queue n Mode Register (OBmMQnMR), **page 16-290**
  - Outbound Block m Message Queue n Status Register (OBmMQnSR), **page 16-292**
  - Outbound Block m Message Queue n Dequeue Pointer Address Register (OBmMQnDPAR), **page 16-292**
  - Outbound Block m Message Queue n Enqueue Pointer Address Register (OBmMQnEPAR), **page 16-293**
  - Outbound Block m Message Queue n Interrupt Enable Register (OBmMQnIER), **page 16-294**
  - Outbound Block m Message Queue n Interrupt Detect Register (OBmMQnIDR), **page 16-296**
- Queue Manager Outbound Block m Completion Queue Registers (for Block m [0–7])
  - Outbound Block m Completion Queue Mode Register (OBmCQMR), **page 16-297**
  - Outbound Block m Completion Queue Status Register (OBmCQSR), **page 16-298**
  - Outbound Block m Completion Queue Dequeue Pointer Address Register (OBmCQDPAR), **page 16-298**
  - Outbound Block m Completion Queue Enqueue Pointer Address Register (OBmCQEPAR), **page 16-299**
  - Outbound Block m Completion Queue Interrupt Coalescing Register (OBmCQICR), **page 16-300**
- Buffer Manager Software Portal n Registers (for Portal n [0–7]; index k [A–H])
  - Software Portal n Interrupt Status Register (SWPn_ISR), **page 16-301**
  - Software Portal n Interrupt Enable Register (SWPn_IER), **page 16-302**
  - Software Portal n Interrupt Status Disable Register (SWPn_ISDR), **page 16-303**
  - Software Portal n Interrupt Inhibit Register (SWPn_IIR), **page 16-304**
  - Software Portal n Interrupt Force Register (SWPn_IFR), **page 16-304**
  - Software Portal n Configuration (SWPn_CONFIG), **page 16-305**
  - Software Portal n Ring Buffer Assignment Register (SWPn_ASSIGN), **page 16-305**
  - Software Portal n Ring Buffer Enable Register (SWPn_EN), **page 16-305**
  - Software Portal n Base Address Register 1 (SWPn_BAR1), **page 16-305**
  - Software Portal n Base Address Register 2 (SWPn_BAR2), **page 16-305**
  - Software Portal n Producer/Consumer Index Base Address Register (SWPn_PICI_BAR), **page 16-305**
  - Software Portal n Acquire Consumer Index k Register (SWPn_ACQ_CI_RINGk), **page 16-309**
  - Software Portal n Release Producer Index k Register (SWPn_REL_PI_RINGk), **page 16-310**
  - Software Portal n Acquire Producer Index k Register (SWPn_ACQ_PI_RINGk), **page 16-311**

- External Memory Access Error Address Register (EMAI_EADR), **page 16-335**
- External Memory Corruption Error Capture Register (EMCI_ECR), **page 16-336**
- External Memory Corruption Error Address Register (EMCI_EADR), **page 16-337**
- IP Block Revision 1 Register (IP_REV_1), **page 16-337**
- IP Block Revision 2 Register (IP_REV_2), **page 16-338**

■ RapidIO Message Unit Global Registers

- Message Unit Mode Register (MUMR), **page 16-339**
- Message Unit Status Register (MUSR), **page 16-339**
- Message Unit Interrupt Enable Register (MUIER), **page 16-340**
- Message Unit Error Detect Register (MUEDR), **page 16-341**
- Message Unit Interrupt Coalescing Register (MUICR), **page 16-342**
- Message Unit T8 Drop Counter Register (MUT8DCR), **page 16-343**
- Message Unit T9 Drop Counter Register (MUT9DCR), **page 16-343**
- Message Unit Error Capture Message Queue Register (MUECMQR), **page 16-344**
- Message Unit Error Capture CD Registers 0–3 (MUECCDR[0–3]), **page 16-344**
- Message Unit Error Capture Address Register (MUECAR), **page 16-346**
- Message Unit Arbitration Weight Register (MUAWR), **page 16-347**
- Message Unit Outbound Interleaving Mask Register (MUOIMR), **page 16-348**
- Message Unit Segmentation Execution Privilege Register 0–1 (MUSEPR[0–1], **page 16-348**
- Message Unit Reassembly Context Assignment Registers 0–2 (MURCAR[0–2], **page 16-350**
- IP Block Revision Register 0 for eMSG (IPBRR0), **page 16-352**
- IP Block Revision Register 1 for eMSG (IPBRR1), **page 16-352**

Note: The base address for the RapidIO registers is: 0xFFF80000.
The base address for the eMSG, BMLite, and QMLite registers is: 0xFFF60000.

16.4.1 RapidIO Registers

16.4.1.1 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	1	0	1	0	0	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

DIDCAR is a read-only register. The device vendor identity field (DVI) identifies the vendor that manufactured the device containing the processing element. A value for DVI is uniquely assigned to a device vendor by the registration authority of the RapidIO Trade Association. The device identity field (DI) is intended to uniquely identify the type of device from the vendor specified by DVI. The values for DI are assigned and managed by the respective vendor.

Table 16-50. DIDCAR Field Descriptions

Bits	Reset	Description
DI 31–16	0x1828	**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 MSC8157E = 0x1828.
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.

16.4.1.2 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

DICAR is a read-only register. The device revision field (DR) identifies the revision level of the Serial RapidIO controller. The value for DR is assigned and managed by the vendor specified by DIDCAR[DVI]. DICAR represents a copy of the device system version register (SVR).

Table 16-51. 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].

16.4.1.3 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

AIDCAR is a read-only register. The assembly vendor identity field (AVI) identifies the vendor that manufactured the assembly or subsystem containing the device. A value for AVI is uniquely assigned to an assembly vendor by the registration authority of the RapidIO Trade Association.

The assembly identity field (AI) is intended to uniquely identify the type of assembly from the vendor specified by AVI. The values for AI are assigned and managed by the respective vendor.

Table 16-52. 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.

16.4.1.4 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 16-53. AICAR Field Descriptions

Bit	Reset	Description
AR 31–16	0x0000	**Assembly Revision**
EFP 15–0	0x0100	**Extended Features Pointer**

16.4.1.5 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	—											
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**
	—										CRF	CTLS	EF	EAS		
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1

PEFCAR identifies the major functionality provided by the processing element.

Table 16-54. PEFCAR Field Descriptions

Bit	Reset	Description	Settings	
BR 31	1	**Bridge** Specifies whether the MSC8157E can bridge to another interface.		
MEM 30	1	**Memory** Specifies whether the MSC8157E has physically addressable local address space and can be accessed as an endpoint through non-maintenance operations.		
PROC 29	1	**Processor** Specifies whether the MSC8157E contains a local processor that executes code.		
SW 28	0	**Switch** The MSC8157E does not bridge to another external RapidIO interface. (SW=0)		
— 27–6	0	Reserved. Write to zero for future compatibility.		
CRF 5	1	**Critical Request Flow** Specifies whether the RapidIO Controller supports the critical request flow.	0 1	Critical request flow not supported. Critical request flow supported.
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 1	Only common transport small systems (up to 256 devices). Large systems up to 65,536 devices.
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.	000 010 001	Reserved. Reserved. 34-bit addresses.

16.4.1.6 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	—		DSTM	DS	—	
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	1	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 MSC8157E.

Table 16-55. SOCAR Field Descriptions

Bit	Reset	Description
R 31	0	**Read Operation** MSC8157E does not support.
IR 30	0	**IRead Operation** MSC8157E does not support.
RO 29	0	**Read-to-Own Operation** MSC8157E does not support.
DCI 28	0	**Data Cache Invalidate Operation** MSC8157E does not support.
C 27	0	**Castout Operation** MSC8157E does not support.
F 26	0	**Flush Operation** MSC8157E does not support a flush operation.
IOR 25	0	**I/O-Read Operation** MSC8157E does not support an I/O read operation.
ICI 24	0	**Instruction Cache Invalidate Operation** MSC8157E does not support.
TIE 23	0	**TLBIE Operation** MSC8157E does not support.
TIES 22	0	**TLBSYNC Operation** MSC8157E does not support.
— 21–20	0	Reserved. Write to zero for future compatibility.
DSTM 19	1	**Data Streaming Traffic Management Operation** MSC8157E supports operation.
DS 18	1	**Data Streaming Operation** MSC8157E supports operation.
— 17–16	0	Reserved. Write to zero for future compatibility.
NR 15	1	**NREAD Operation** MSC8157E supports operation.
NW 14	1	**NWRITE Operation** MSC8157E supports operation.
SW 13	1	**SWRITE Operation** MSC8157E supports operation.

Table 16-55. SOCAR Field Descriptions (Continued)

Bit	Reset	Description
NWR 12	1	**NWRITE_R Operation** MSC8157E supports operation.
M 11	1	**Message Operation** MSC8157E supports message operation.
D 10	1	**Doorbell Operation** MSC8157E supports doorbell operation.
— 9	—	Reserved. Write to zero for future compatibility.
ATS 8	0	**Atomic-Test-and-Swap Operation** MSC8157E does not support.
AI 7	0	**Atomic-Inc Operation** MSC8157E does not support.
AD 6	0	**Atomic-Dec Operation** MSC8157E does not support.
AS 5	0	**Atomic-Set Operation** MSC8157E does not support.
AC 4	0	**Atomic-Clr Operation** MSC8157E does not support.
— 3	0	Reserved. Write to zero for future compatibility.
PW 2	0	**Port-Write Operation** MSC8157E does not support.
— 1–0	0	Reserved. Write to zero for future compatibility.

16.4.1.7 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	—		DSTM	DS	—	
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	1	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 MSC8157E can support as a target.

Table 16-56. DOCAR Field Descriptions

Bit	Reset	Description
R 31	0	**Read Operation** MSC8157E does not support.
IR 30	0	**IRead Operation** MSC8157E does not support.
RO 29	0	**Read-to-Own Operation** MSC8157E does not support.

Table 16-56. DOCAR Field Descriptions (Continued)

Bit	Reset	Description
DCI 28	0	**Data Cache Invalidate Operation** MSC8157E does not support.
C 27	0	**Castout Operation** MSC8157E does not support.
F 26	0	**Flush Operation** MSC8157E does not support.
IOR 25	0	**I/O-Read Operation** MSC8157E does not support.
ICI 24	0	**Instruction Cache Invalidate Operation** MSC8157E does not support.
TIE 23	0	**TLBIE Operation** MSC8157E does not support.
TIES 22	0	**TLBSYNC Operation** MSC8157E does not support.
— 21–20	0	Reserved. Write to zero for future compatibility.
DSTM 19	1	**Data Streaming Traffic Management Operation** Supported
DS 18	1	**Data Streaming Operation** Supported
— 17–16	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.
— 9	—	Reserved. Write to zero for future compatibility.
ATS 8	0	**Atomic-Test-and-Swap Operation** MSC8157E does not support.
AI 7	0	**Atomic-Inc Operation** MSC8157E does not support.
AD 6	0	**Atomic-Dec Operation** MSC8157E does not support.
AS 5	0	**Atomic-Set Operation** MSC8157E does not support.
AC 4	0	**Atomic-Clr Operation** MSC8157E does not support.
— 3	0	Reserved. Write to zero for future compatibility.

Table 16-56. DOCAR Field Descriptions (Continued)

Bit	Reset	Description
PW 2	1	**Port-Write Operation** Supported.
— 1–0	0	Reserved. Write to zero for future compatibility.

16.4.1.8 Data Streaming Information Capability Register (DSICAR)

DSICAR Data Streaming Information Capability Register 0x0003C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MaxPDU															
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**
	SegSupport															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

DSICAR defines the data streaming capabilities of a processing element. It is required for destination end point devices. DSICAR is a read-only register.

Table 16-57. DSICAR Field Definitions

Bits	Reset	Description	Settings
MaxPDU 31–16	0x0000	**Maximum PDU** Specifies the maximum PDU size supported by the destination end point (MaxPDU = 0x0000)	0x0000 64 Kbytes
SegSupport 15–0	0x0018	**Segment Support** Indicates the number of segmentation contexts supported by the destination end point.	0x0018 24 segmentation contexts.

16.4.1.9 Data Streaming Logical Layer Control Command and Status Register (DSLLCCSR)

DSLLCCSR Data Streaming Logical Layer Control Command and Status Register — Offset 0x00048

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TMTypesSupport				TMModes				—							
TYPE	R				R/W				R							
RESET	1	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**
	—								MTU							
TYPE	R								R/W							
RESET	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

DSLLCCSR is used for general command and status information for the logical interface.:

Table 16-58. DSLLCCSR Field Descriptions

Bits	Reset	Description	Settings
TMTypesSupport 31–28	0	**Traffic Management Types Supported** This read-only field supports only the basic type.	0x8 Basic type.
TMMode 27–24	0	**Traffic Management Mode** Specifies the traffic management mode.	0x0 Traffic management disabled. 0x1 Basic. All other values reserved.
— 25–8	0	Reserved. Write to zero for future compatibility.	
MTU 7–0	0x40	**Maximum Transmission Unit** Controls the data payload size for segments of an encapsulated PDU. Only single segment PDUs and end segments are permitted to have a data payload less than this value. The MTU can be specified in increments of 4 bytes. Support for the entire range is required.	0x00– 0x07 Reserved 0x08 32 byte block size 0x09 36 byte block size 0x0A 40 byte block size 0x0B 44 byte block size 0x0C 48 byte block size ... 0x3F 252 byte block size 0x40 256 byte block size 0x41– 0xFF Reserved.

16.4.1.10 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 16-59. 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.

16.4.1.11 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	0	0	0	0	0	0	0	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

LCSBA1CSR specifies the least significant bits of the local physical address double-word offset for the RapidIO configuration register space allowing the configuration register space to be physically mapped in the processing element. This register allows configuration and maintenance of an RapidIO block 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 16-60. 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.

16.4.1.12 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 16-61. 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 5-20). 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 5-20). If the RapidIO controller is configured as an agent, LBDID = 0xFFFF.

16.4.1.13 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 16-62. 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.

16.4.1.14 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 16-63. CTCSR Field Descriptions

Bit	Reset	Description
CT 31–0	0	**Component Tag**

16.4.1.15 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 16-64. PMBH0 Field Descriptions

Bit	Reset	Description
EFPTR 31–15	0x0600	**Extended Features Pointer**
EFID 16–31	0x0001	**Extended Features ID**

16.4.1.16 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.

Table 16-65. 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.

16.4.1.17 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.

Table 16-66. 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.

16.4.1.18 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.

The user must initialize the value of M to 1. Otherwise, no outbound transactions are initiated by the MSC8157E, including messages and doorbells.

Table 16-67. 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.	

16.4.1.19 Port 1–2 Link Maintenance Request Command and Status Register (PnLMREQCSR)

P1LMREQCSR Port 1–2 Link Maintenance Request Offset 0x00140
P2LMREQCSR 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 1–2 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 16-68** lists PnLMREQCSR fields.

Table 16-68. 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.

16.4.1.20 Port 1–2 Link Maintenance Response Command and Status Register (PnLMRESPCSR)

P1LMRESPCSR Port 1–2 Link Maintenance Response Offset 0x00144
P2LMRESPCSR 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 1–2 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 16-69** lists PnLMRESPCSR fields.

Table 16-69. 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.	

16.4.1.21 Port 1–2 Local ackID Command and Status Register (PnLASCR)

P1LASCSR Port 1–2 Local ackID Command and Status Register Offset 0x00148
P2LASCSR 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 1–2 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 16-70** lists PnLASCSR fields.

Table 16-70. 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

16.4.1.22 Port 1–2 Error and Status Command and Status Register (PnESCSR)

P1ESCSR Port n Error Command and Status Register Offset 0x00158
P2ESCSR 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 16-71. 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.
ORS 18	0	**Output Stop** The output port stops due to a retry. Read only.

Table 16-71. PnESCSR Field Descriptions (Continued)

Bit	Reset	Description
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 PnCCSR[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.

16.4.1.23 Port 1–2 Control Command and Status Register (PnCCSR)

P1CCSR Port 1–2 Control Command and Status Register Offset 0x0015C
P2CCSR 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	—	EB	—
TYPE	R					R/W						R			R/W	R
RESET	1	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 16-72. PnCCSR Field Descriptions

Bit	Reset	Description	Settings
PW 31–30	0b11	**Port Width** Specifies the supported hardware port widths. Read only.	00 Single-lane port supported. 01 Single and four-lane port supported 10 Single and two-lane port supported. 11 Single-, two-, and four-lane port supported (default)
IPW 29–27	0b010	**Initialized Port Width** Specifies the width of the ports after they are initialized. Read only. If the port degrades to x1, 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. 001 Single-lane port, lane 2. 010 Four-lane port, lanes 0–3. 011 Two-lane port, lanes 0–1. 100–111 Reserved.
PWO 26–24	0b000	**Port Width Override** Soft port configuration to control the width modes available for port initialization. The meaning of bits 25–24 is determined by the value of bit 26. When bit 26 = 0b1, bit 25 controls enabling x4 mode and bit 24 controls enabling of x2 mode. **Note:** This field should be changed only when the port is uninitialized. To achieve this, first set PnCCSR[PD] to 1 (port disabled). Then change PWO to any legal value. Finally, set PD back to 0 (enabled)	000 No override (the default). 001 Reserved. 010 Force single lane, lane reversal not forced. 011 Force single lane and lane reversal. 100 x2 and x4 mode disabled. 101 x2 mode enabled, x4 mode disabled. 110 x2 mode disabled, x4 mode enabled. 111 x2 mode enabled, x4 mode enabled.
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. If OPE = 0, the following conditions apply: • Inbound non-maintenance requests packets can generate responses. • If the device configuration allows outbound requests to be pending in the controller, inbound maintenance requests may not generate responses. • Outbound maintenance requests are not be transmitted. Set the bit (1) to allow transmission of any request or response packet. To allow responses to inbound maintenance requests when OPE = 0, disable any LAWs with a RapidIO target to prevent outbound request packets from being sent to the RapidIO controller. The initial value of OPE is read from configuration pins.	0 Port is stopped and not enabled to issue packets. Port may generate responses to inbound maintenance and to other types of transactions. 1 Port is enabled to issue packets.

Table 16-72. PnCCSR Field Descriptions (Continued)

Bit	Reset	Description	Settings
IPE 21	1	**Input Port Receive Enable** Specifies whether the port is enabled 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 signalled 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 MSC8157E does not participate in multicast events.	
— 18	0	Reserved. Write to zero for future compatibility.	
EB 17	0	**Enumeration Boundary** An enumeration boundary aware system algorithm honors this flag. The algorithm, at either the ingress or the egress port, will not count past a port with this bit set.	0 No enumeration boundary checking. 1 Enumeration boundary checking enabled.
— 16–4	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.	0 Drop packet response disabled. 1 Drop packet response enabled.
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.

16.4.1.24 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 16-73. 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.

16.4.1.25 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	—	GER	—	ITD	ITTE	—	PRT	UR	UT	MDSC	OE DSC	LDSS	SDSS	DSP DUL	—
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**
	—								IACB	OACB	—	RETE	TSE	PTTL	—	
TYPE	R/W															
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. LTLEDCSR is stored in each RapidIO Endpoint port and the Message Unit, although the values in this register can differ for each port/Message Unit. A port's LTLEDCSR cannot detect any errors if the port or the Message Unit has captured an enabled Logical/Transport layer error until the detected error is cleared, and likewise, the Message Unit's LTLEDCSR cannot detect any errors if the Message Unit or any port has captured an enabled Logical/Transport layer error.

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. If more than one port or Message Unit detects one or more enabled errors in the same cycle, the capture registers will be saved in the top port / Message Unit in the SBus daisy chain that detected an enabled error, and the set and enabled detect bits of the port(s)/Message Unit below will be masked from the SBus daisy chain. This means that a read of LTLEDCSR will only return the disabled set bits from any port/Message Unit and enabled set bits from the top port / Message Unit in the daisy chain with a set enabled error, and that a read of the capture registers will return the values in the top port / Message Unit in the daisy chain with a set enabled error; that is, the set enabled detect bits will correspond to the capture registers.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 16-74. LTLEDCSR Field Descriptions

Bit	Name	Description
IER 31	0	**I/O Error Response** Indicates an error response for an I/O logical layer request.
— 30	0	Reserved. Write to zero for future compatibility.
GER 29	0	**GSM Error Response** Indicates a response of ERROR for a GSM logical layer request.
— 28	0	Reserved. Write to zero for future compatibility.
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.
— 25	0	Reserved. Write to zero for future compatibility.
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.
MDSC 21	0	**Missing Data Streaming Context** A continuation or end data streaming segment was received for a closed or non-existent segmentation context.
OEDSC 20	0	**Open Existing Data Streaming Context** A start or single data streaming segment was received for an already open segmentation context.
LDSS 19	0	**Long Data Streaming Segment** Received a data streaming segment with a payload size greater than the MTU.

Table 16-74. LTLEDCSR Field Descriptions (Continued)

Bit	Name	Description
SDSS 18	0	**Short Data Streaming Segment** Received a start or continuation data streaming segment with a payload size less than the MTU.
DSPDUL 17	0	**Data Streaming PDU Length** The length of a reassembled PDU differs from the PDU length carried in the end data streaming segment. packet header
— 16–8	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.

16.4.1.26 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	—	GER	—	ITD	ITTE	—	PRT	UR	UT	MDSC	OE DSC	LDSS	SDSS	DSP DUL	—
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**
	—								IACB	OACB	—	RETE	TSE	PTTL	—	
TYPE	R/W															
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 16-75. 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.
— 30	0	Reserved. Write to zero for future compatibility.
GER 29	0	**GSM Error Response Enable** Enables reporting of a GSM error response. It captures and locks the error.
— 28	0	Reserved. Write to zero for future compatibility.
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.
— 25	0	Reserved. Write to zero for future compatibility.
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.
MDSC 21	0	**Missing Data Streaming Context** Enables reporting of an unsupported transaction error. Capture and lock the error.
OEDSC 20	0	**Open Existing Data Streaming Context** Enable reporting of an unsupported transaction error. Capture and lock the error.
LDSS 19	0	**Long Data Streaming Segment** Enable reporting of an unsupported transaction error. Capture and lock the error.
SDSS 18	0	**Short Data Streaming Segment** Enable reporting of an unsupported transaction error. Capture and lock the error.
DSPDUL 17	0	**Data Streaming PDU Length** Enable reporting of an unsupported transaction error. Capture and lock the error.
— 16–8	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.
— 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.

16.4.1.27 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 16-76. 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 16.2.14.3**, *Logical Layer RapidIO Errors*, on page 16-64.
— 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 16.2.14.3**, *Logical Layer RapidIO Errors,* on page 16-64.

16.4.1.28 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 16-77. 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 16.2.14.3**, *Logical Layer RapidIO Errors,* on page 16-64.
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 16.2.14.3**, *Logical Layer RapidIO Errors,* on page 16-64.
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 16.2.14.3**, *Logical Layer RapidIO Errors,* on page 16-64.
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 16.2.14.3**, *Logical Layer RapidIO Errors,* on page 16-64.

16.4.1.29 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**
	EPN				—											
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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 16-78. LTLCCCSR Field Descriptions

Bit	Reset	Description
FT 31–28	0	**Format Type** Normally, the format type associated with the error. For details, see **Section 16.2.14.3**, *Logical Layer RapidIO Errors,* on page 16-64.
TT 27–24	0	**Transaction Type** Normally, the transaction type associated with the error. For details, see **Section 16.2.14.3**, *Logical Layer RapidIO Errors,* on page 16-64.
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 16.2.14.3**, *Logical Layer RapidIO Errors,* on page 16-64.
EPN 15–12	0	**Encoded Port Number** Indicates the port from which the error information was captured.
— 11–0	0	Reserved. Write to zero for future compatibility.

16.4.1.30 Port 1–2 Error Detect Command and Status Register (PnEDCSR)

P1EDCSR Port 1–2 Error Detect Command and Status Register Offset 0x00640
P2EDCSR 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 16-79. 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.

16.4.1.31 Port 1–2 Error Rate Enable Command and Status Register (PnERECSR)

P1ERECSR Port 1–2 Error Rate Enable Command and Status Register Offset 0x00644
P2ERECSR 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 1–2 error rate threshold register and lock the Port 1–2 error capture registers.

Table 16-80. 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.
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.

16.4.1.32 Port 1–2 Error Capture Attributes Command and Status Register (PnECACSR)

P1ECACSR Port 1–2 Error Capture Attributes Command and Status Register Offset 0x00648
P2ECACSR 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															
RESET	0	0	0	0	0	0	0	0	0	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															
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 1–2 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 16-81. 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.	
ET 28–24	0	**Error** The encoded value of the bit in the Port 1–2 error detect CSR that describes the error captured in the Port 1–2 error capture CSRs.	

Table 16-81. PnECACSR Field Descriptions

Bit	Reset	Description	Settings
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.	

16.4.1.33 Port 1–2 Packet/Control Symbol Error Capture Command and Status Register (PnPCSECCSR)

P1PCSECCSR Port 1–2 Packet/Control Symbol Error Capture Command and Status Register Offset 0x0064C
P2PCSECCSR 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 16-82. 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.

16.4.1.34 Port 1–2 Packet Error Capture Command and Status Register 1 (PnPECCSR1)

P1PECCSR1 Port 1–2 Packet Error Capture Command and Status Register 1 Offset 0x00650
P2PECCSR1 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 16-83. PnPECCSR1 Field Descriptions

Bit	Reset	Description
C1 31–0	0	**Capture 1** Contains bytes 4–7 of the packet header.

16.4.1.35 Port 1–2 Packet Error Capture Command and Status Register 2 (PnPECCSR2)

P1PECCSR2 Port 1–2 Packet Error Capture Command and Status Register 2 Offset 0x00654
P2PECCSR2 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 16-84. PECCSR2 Field Descriptions

Bit	Reset	Description
C2 31–0	0	**Capture 2** Bytes 8 to 11 of the packet header

16.4.1.36 Port 1–2 Packet Error Capture Command and Status Register 3 (PnPECCSR3)

P1PECCSR3 Port 1–2 Packet Error Capture Command and Status Register 3 Offset 0x00658
P2PECCSR3 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 16-85. PnPECCSR3 Field Descriptions

Bit	Reset	Description
C3 31–0	0	**Capture 3** Bytes 12–15 of the packet header.

16.4.1.37 Port 1–2 Error Rate Command and Status Register (PnERCSR)

P1ERCSR Port 1–2 Error Rate Command and Status Register Offset 0x00668
P2ERCSR 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 Portn Error Rate Threshold Command and Status Register (PnERTCSR) to monitor and control the reporting of transmission errors.

Table 16-86. 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.	

Table 16-86. PnERCSR Field Descriptions (Continued)

Bit	Reset	Description	Setting
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).	

16.4.1.38 Port 1–2 Error Rate Threshold Command and Status Register (PnERTCSR)

P1ERTCSR Port 1–2 Error Rate Threshold Command and Status 'Register — Offset 0x0066C
P2ERTCSR — 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 16-87. 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] exceeds the ERFTT value.	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] exceeds the ERDTT value.	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.	

16.4.1.39 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	NPUE	—											
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

LLCR contains general port-common logical layer mode enables.

Table 16-88. 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.
NPUE 28	0	**NWRITE_R Priority Update Enable** When set, all inbound NWRITE_R transactions return a response with priority set to 3. When cleared, normal functionality of returning a response with Priority + 1 is assumed.
— 27–0	0	Reserved. Write to zero for future compatibility.

16.4.1.40 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**
	—															
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 16-89. EPWISR Field Descriptions

Bit	Reset	Description
PINT0 31	0	**Physical or Logical/Transport Error Interrupt Port 1** Indicates a physical or logical transport error interrupt was generated by port 1.
PINT1 30	0	**Physical or Logical/Transport Error Interrupt Port 2** Indicates a physical or logical transport error interrupt was generated by port 2.
— 29–0	0	Reserved. Can be used to indicate errors on more ports if they exist.

16.4.1.41 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 16-90. 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.

16.4.1.42 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 16-91. 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.

16.4.1.43 Port 1–2 Alternate Device ID Command and Status Register (PnADIDCSR)

P1ADIDCSR Port 1–2 Alternate Device ID Command and Status Register Offset 0x10100
P2ADIDCSR 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 san alternate device ID. This register should be enabled before the initiator enabled bit of the PGCCSR is set (see **Table 16-67*PGCCSR Field Descriptions,*** on page 16>-195). 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 16-92. 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.

16.4.1.44 Port 1–2 Pass-Through Accept-All Configuration Register (PnPTAACR)

P1PTAACR Port 1–2 Pass-Through Accept-All Configuration Register Offset 0x10120
P2PTAACR 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 16-93. PnPTAACR Field Descriptions

Bit	Reset	Description	Settings
PTPN 31–28	Port 1= 1 Port 2 = 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 (1 or 2) to the OCN port number (0 or 4).	**RapidIO Port1** 0 The pass-through packet will be looped back from RapidIO Port1 to same RapidIO Port1 1 The pass-through packet will be passed from RapidIO Port1 to RapidIO Port2 **RapidIO Port2** 1 The pass-through packet will be looped back from RapidIO Port2 to same RapidIO Port2 0 The pass-through packet will be passed from RapidIO Port2 to RapidIO Port1
— 27–2	0	Reserved. Write to zero for future compatibility.	

Table 16-93. PnPTAACR Field Descriptions

Bit	Reset	Description	Settings
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 1 = RCWHR[R1A] Port 2 = 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.

16.4.1.45 Port 1–2 Logical Outbound Packet Time-to-Live Configuration Register (PnLOPTTLCR)

P1LOPTTLCR Port 1–2 Logical Outbound Packet Time-to-Live Configuration Register Offset 0x10124
P2LOPTTLCR 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 1–2 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.

Table 16-94. 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.

16.4.1.46 Port 1–2 Implementation Error Command and Status Register (PnIECSR)

P1IECSR Port 1–2 Implementation Error Command and Status Register Offset 0x10130
P2IECSR 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 16-95. 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.

16.4.1.47 Port 1–2 Physical Configuration Register (PnPCR)

P1PCR Port 1–2 Physical Configuration Register Offset 0x10140
P2PCR Offset 0x101C0

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**
	CCC	—										CCP	—	OBDEN	—	
TYPE	R/W															
RESET	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

PnPCR contains general physical layer protocol and link mode enables.

Table 16-97. PnPCR Field Descriptions

Bit	Reset	Description
— 31–16	0	Reserved. Write to zero for future compatibility.
CCC 15	1	**CRC Checking Enable for Control Symbol** CRC is checked on received control symbols. When CCC is cleared, no CRC is checked on received control symbols.
— 14–13	0	Reserved. Write to zero for future compatibility.
— 12–5	0	Reserved. Write to zero for future compatibility.
CCP 4	1	**CRC Checking Enable for Packets** CRC is checked on received packets. When CCP is cleared, no CRC is checked on received packets.
— 3	0	Reserved. Write to zero for future compatibility.
OBDEN 2	0	**Output Buffer Drain Enable** When this bit is set, the output drains packets from the outbound buffer and does not send them out, intentionally causing the inbound to time-out when a response on a drained request is expected and to send an error response.
— 1–0	0	Reserved. Write to zero for future compatibility.

16.4.1.48 Port 1–2 Serial Link Command and Status Register (PnSLCSR)

P1SLCSR Port 1–2 Serial Link Command and Status Register Offset 0x10158
P2SLCSR 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/W				W1C	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

PnSLCSR contains status of the of the serial physical link.

Table 16-98. 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.

16.4.1.49 Port 1–2 Serial Link Error Injection Configuration Register (PnSLEICR)

P1SLEICR Port 1–2 Serial Link Error Injection Configuration Register Offset 0x10160
P2SLEICR Offset 0x101E0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EIC					—							EIR			
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**
	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 16-99. 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.	

16.4.1.50 Port *n* Arbitration 0 Tx Configuration Register (P*n*A0TxCR)

P1A0TxCR Port 1–2 Arbitration 0 Tx Configuration Register Offset 0x10164
P2A0TxCR Offset 0x101E4

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**
	—						SPSA_TH_MSCREQ[9–0]									
TYPE	R/W															
RESET	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

PnA0TxCR is used to control the arbitration of packets within the serial RapidIO controller transmit logic (Tx), specifically at arbitration point 0. Please see

Section 16.2.8.3.3, *Arbitration*, on page 16-23 for details on using this register. **Table 16-100** contains detailed descriptions of the fields within PnA0TxCR.

Table 16-100. P*n*A0TxCR Field Descriptions

Bit	Reset	Description	Settings
— 31–10	0	Reserved. Write to zero for future compatibility.	
SPSA_TH_ MSGREQ 9–0	0x3FF	**Message Arbitration Threshold for Point 0** Sets a strict priority with starvation avoidance (arbitration point 0 Tx) threshold (count) unit request packets.	0x000 Disabled. A low priority message unit request will never promote to the highest priority. Use at own risk; may cause deadlocks. 0x001 Strict priority with starvation avoidance enabled. A low priority message unit request packet may lose strict priority arbitration 1 consecutive time before it promotes to highest priority (starvation avoidance). ... 0x3FF Count of 1023. Strict priority with starvation avoidance enabled. A low priority message unit request packet may lose strict priority arbitration 1023 consecutive times before it promotes to highest priority (starvation avoidance).

16.4.1.51 Port *n* Arbitration 1 Tx Configuration Register (P*n*A1TxCR)

P1A1TxCR Port 1–2 Arbitration 1 Tx Configuration Register Offset 0x10168
P2A1TxCR Offset 0x101E8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WRR_SEL	—			WRR_TH_MSGFC[3–0]				WRR_TH_MSGRSP[3–0]				WRR_TH_MSGREQ[3–0]			
TYPE	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**
	—											SPSA_TH_MSG[4–0]				
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

P*n*A1TxCR is used to control the arbitration of packets within the serial RapidIO controller transmit logic (Tx), specifically at arbitration point 1. Please see **Section 16.2.8.3.3**, *Arbitration*, on page 16-23 for details on using this register. **Table 16-101** contains detailed descriptions of the fields within PnA1TxCR.

Table 16-101. P*n*A1TxCR Field Descriptions

Bit	Reset	Description	Settings
WRR_SEL 31	1	**Weighted Round Robin Select** Indicates the arbitration type to use at arbitration point 1 Tx.	0 Strict priority with starvation avoidance: SPSA_TH_MSG[4–0] is valid. 1 Weighted-round-robin: WRR_TH_MSGFC[3–0], WRR_TH_MSGRSP[3–0], and WRR_TH_MSGREQ[3–0] are valid.
— 30–28	0	Reserved. Write to zero for future compatibility.	
WRR_TH_MSGFC 27–24	0	**Weighted Round Robin Threshold (Arbitration Point 1) Message Unit Flow Control Packets** Sets the threshold (weight) for message unit flow control packets.	0x0 Weight of 1. 1 consecutive message unit flow control packet before considering other packets. 0x1 Weight of 2. 2 consecutive message unit flow control packets before considering other packets. ... 0xF Weight of 16: 16 consecutive message unit flow control packets before considering other packets.
WRR_TH_MSGRSP 23–20	0x3FF	**Weighted Round Robin Threshold (Arbitration Point 1) Message Unit Response Packets** Sets the threshold (weight) for message unit response packets.	0x0 Weight of 1. 1 consecutive message unit response packet before considering other packets. 0x1 Weight of 2. 2 consecutive message unit response packets before considering other packets. ... 0xF Weight of 16: 16 consecutive message unit response packets before considering other packets.

Table 16-101. P*n*A1TxCR Field Descriptions (Continued)

Bit	Reset	Description	Settings
WRR_TH_ MSGREQ 19–16	0x3FF	**Weighted Round Robin Threshold (Arbitration Point 1) Message Unit Request Packets** Sets the threshold (weight) for message unit request packets.	0x0 Weight of 1. 1 consecutive message unit request packet before considering other packets. 0x1 Weight of 2. 2 consecutive message unit request packets before considering other packets. ... 0xF Weight of 16: 16 consecutive message unit request packets before considering other packets.
— 15–5	0	Reserved. Write to zero for future compatibility.	
SPSA_TH_MSG 4–0	0	**Strict Priority with Starvation Avoidance (Arbitration Point 1) Threshold for Message Unit Packets.** Sets the strict priority for message unit flow control packets.	0x00 Disabled. A low priority message unit request will never promote to the highest priority. Use at own risk; may cause deadlocks. 0x01 Strict priority with starvation avoidance enabled. A low priority message unit request packet may lose strict priority arbitration 1 consecutive time before it promotes to highest priority (starvation avoidance). ... 0x1F Count of 31. Strict priority with starvation avoidance enabled. A low priority message unit request packet may lose strict priority arbitration 31 consecutive times before it promotes to highest priority (starvation avoidance).

16.4.1.52 Port *n* Arbitration 2 Tx Configuration Register (P*n*A2TxCR)

P1A2TxCR Port 1–2 Arbitration 2 Tx Configuration Register Offset 0x1016C
P2A2TxCR Offset 0x101EC

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WRR_SEL	—			WRR_TH_MSG[3–0]				—				WRR_TH_SYS[3–0]			
TYPE	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**
	—											SPSA_TH_SRIO[4–0]				
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

P*n*A2TxCR is used to control the arbitration of packets within the serial RapidIO controller transmit logic (Tx), specifically at arbitration point 2. Please see

Section 16.2.8.3.3, *Arbitration*, on page 16-23 for details on using this register. **Table 16-102** contains detailed descriptions of the fields within PnA2TxCR.

Table 16-102. P*n*A2TxCR Field Descriptions

Bit	Reset	Description	Settings
WRR_SEL 31	1	**Weighted Round Robin Select** Indicates the arbitration type to use at arbitration point 2 Tx.	0 Strict priority with starvation avoidance: SPSA_TH_MSG[4–0] is valid. 1 Weighted-round-robin: WRR_TH_MSG[3–0] and WRR_TH_SYS[3–0] are valid.
— 30–28	0	Reserved. Write to zero for future compatibility.	
WRR_TH_MSG 27–24	0	**Weighted Round Robin Threshold (Arbitration Point 1) Message Unit Packets** Sets the threshold (weight) for message unit packets.	0x0 Weight of 1. 1 consecutive message unit flow control packet before considering other packets. 0x1 Weight of 2. 2 consecutive message unit flow control packets before considering other packets. ... 0xF Weight of 16: 16 consecutive message unit flow control packets before considering other packets.
— 23–20	0	Reserved. Write to zero for future compatibility.	
WRR_TH_SYS 19–16	0x3FF	**Weighted Round Robin Threshold (Arbitration Point 1) System Packets** Sets the threshold (weight) for system packets.	0x0 Weight of 1. 1 consecutive system packet before considering other packets. 0x1 Weight of 2. 2 consecutive system packets before considering other packets. ... 0xF Weight of 16: 16 consecutive system packets before considering other packets.
— 15–5	0	Reserved. Write to zero for future compatibility.	
SPSA_TH_SRIO 4–0	0	**Strict Priority with Starvation Avoidance (Arbitration Point 1) Threshold for RapidIO Packets.** Sets the strict priority for RapidIO packets.	0x00 Disabled. A low priority RapidIO packet will never promote to the highest priority. Use at own risk; may cause deadlocks. 0x01 Strict priority with starvation avoidance enabled. A low priority RapidIO packet may lose strict priority arbitration 1 consecutive time before it promotes to highest priority (starvation avoidance). ... 0x1F Count of 31. Strict priority with starvation avoidance enabled. A low priority RapidIO packet may lose strict priority arbitration 31 consecutive times before it promotes to highest priority (starvation avoidance).

16.4.1.53 Port *n* Message Request Tx Buffer Allocation Configuration Register 0 (P*n*MReqTxBACR0)

P1MReqTxBACR0	Port 1–2 Message Request Tx Buffer	Offset 0x10170
P2MReqTxBACR0	Allocation Configuration Register 0	Offset 0x101F0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—				DedAG0 [3–0]				—				DedAG1 [3–0]			
TYPE	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**
	—				DedAG2 [3–0]				—				DedAG3 [3–0]			
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

The Port *n* Message Request Tx Buffer Allocation Configuration Register 0 (P*n*MReqTxBACR0) is used to control the buffer allocation of message unit request packets within the serial RapidIO controller transmit logic (Tx).

Note: There are only 15 buffers that hold message unit request packets. The total value of the fields within this register and the registers described in **Section 16.4.1.54**, *Port n Message Request Tx Buffer Allocation Configuration Register 1 (PnMReqTxBACR1)*, on page 16-233 and **Section 16.4.1.55**, *Port n Message Request Tx Buffer Allocation Configuration Register 2 (PnMReqTxBACR2)*, on page 16-235 must not exceed 15 (undefined behavior will result).

Please see **Section 16.2.8.3**, *Buffer Allocation*, on page 16-20 for details on the use of this register. **Table 16-103** contains detailed descriptions of the fields within PnMReqTxBACR0.

Table 16-103. P*n*MReqTxBACR0 Field Descriptions

Bit	Reset	Description	Settings
— 31–28	0	Reserved. Write to zero for future compatibility.	
DedAG0[3–0] 27–24	1	**Dedicated Arbitration Group 0** Controls the buffer allocation of arbitration group 0 message unit request packets.	0x0 No dedicated buffer allocated for Group 0 message unit request packets. 0x1 One dedicated buffer allocated for Group 0 message unit request packets. ... 0xF Fifteen dedicated buffers allocated for Group 0 message unit request packets.
— 23–20	0	Reserved. Write to zero for future compatibility.	

Table 16-103. P*n*MReqTxBACR0 Field Descriptions (Continued)

Bit	Reset	Description	Settings
DedAG1[3–0] 19–16	1	**Dedicated Arbitration Group 1** Controls the buffer allocation of arbitration group 0 message unit request packets.	0x0 No dedicated buffer allocated for Group 1 message unit request packets. 0x1 One dedicated buffer allocated for Group 1 message unit request packets. ... 0xF Fifteen dedicated buffers allocated for Group 1 message unit request packets.
— 15–12	0	Reserved. Write to zero for future compatibility.	
DedAG2[3–0] 11–8	1	**Dedicated Arbitration Group 2** Controls the buffer allocation of arbitration group 0 message unit request packets.	0x0 No dedicated buffer allocated for Group 2 message unit request packets. 0x1 One dedicated buffer allocated for Group 2 message unit request packets. ... 0xF Fifteen dedicated buffers allocated for Group 2 message unit request packets.
— 7–4	0	Reserved. Write to zero for future compatibility.	
DedAG3[3–0] 3–0	1	**Dedicated Arbitration Group 3** Controls the buffer allocation of arbitration group 0 message unit request packets.	0x0 No dedicated buffer allocated for Group 3 message unit request packets. 0x1 One dedicated buffer allocated for Group 3 message unit request packets. ... 0xF Fifteen dedicated buffers allocated for Group 3 message unit request packets.

16.4.1.54 Port *n* Message Request Tx Buffer Allocation Configuration Register 1 (P*n*MReqTxBACR1)

P1MReqTxBACR1 Port 1–2 Message Request Tx Buffer Offset 0x10174
P2MReqTxBACR1 Allocation Configuration Register 1 Offset 0x101F4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—				DedAG4 [3–0]				—				DedAG5 [3–0]			
TYPE	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**
	—				DedAG6 [3–0]				—				DedAG7 [3–0]			
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

The Port *n* Message Request Tx Buffer Allocation Configuration Register 1 (P*n*MReqTxBACR1) is used to control the buffer allocation of message unit request packets within the serial RapidIO controller transmit logic (Tx).

Note: There are only 15 buffers that hold message unit request packets. The total value of the fields within this register and the registers described in **Section 16.4.1.53**, *Port n Message Request Tx Buffer Allocation Configuration Register 0 (PnMReqTxBACR0)*, on page 16-232 and **Section 16.4.1.55**, *Port n Message Request Tx Buffer Allocation Configuration Register 2 (PnMReqTxBACR2)*, on page 16-235 must not exceed 15 (undefined behavior will result).

Please see **Section 16.2.8.3**, *Buffer Allocation*, on page 16-20 for details on the use of this register. **Table 16-104** contains detailed descriptions of the fields within PnMReqTxBACR1.

Table 16-104. P*n*MReqTxBACR1 Field Descriptions

Bit	Reset	Description	Settings
— 31–28	0	Reserved. Write to zero for future compatibility.	
DedAG4[3–0] 27–24	1	**Dedicated Arbitration Group 4** Controls the buffer allocation of arbitration group 0 message unit request packets.	0x0 No dedicated buffer allocated for Group 4 message unit request packets. 0x1 One dedicated buffer allocated for Group 4 message unit request packets. ... 0xF Fifteen dedicated buffers allocated for Group 4 message unit request packets.
— 23–20	0	Reserved. Write to zero for future compatibility.	
DedAG5[3–0] 19–16	1	**Dedicated Arbitration Group 5** Controls the buffer allocation of arbitration group 0 message unit request packets.	0x0 No dedicated buffer allocated for Group 5 message unit request packets. 0x1 One dedicated buffer allocated for Group 5 message unit request packets. ... 0xF Fifteen dedicated buffers allocated for Group 5 message unit request packets.
— 15–12	0	Reserved. Write to zero for future compatibility.	
DedAG6[3–0] 11–8	1	**Dedicated Arbitration Group 6** Controls the buffer allocation of arbitration group 0 message unit request packets.	0x0 No dedicated buffer allocated for Group 6 message unit request packets. 0x1 One dedicated buffer allocated for Group 6 message unit request packets. ... 0xF Fifteen dedicated buffers allocated for Group 6 message unit request packets.
— 7–4	0	Reserved. Write to zero for future compatibility.	
DedAG7[3–0] 3–0	1	**Dedicated Arbitration Group 7** Controls the buffer allocation of arbitration group 0 message unit request packets.	0x0 No dedicated buffer allocated for Group 7 message unit request packets. 0x1 One dedicated buffer allocated for Group 7 message unit request packets. ... 0xF Fifteen dedicated buffers allocated for Group 7 message unit request packets.

16.4.1.55 Port *n* Message Request Tx Buffer Allocation Configuration Register 2 (P*n*MReqTxBACR2)

P1MReqTxBACR2 Port 1–2 Message Request Tx Buffer Offset 0x10178
P2MReqTxBACR2 Allocation Configuration Register 2 Offset 0x101F8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—				GenAG [3–0]				—							
TYPE	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**
	—															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

The Port *n* Message Request Tx Buffer Allocation Configuration Register 2 (P*n*MReqTxBACR2) is used to control the buffer allocation of message unit request packets within the serial RapidIO controller transmit logic (Tx).

Note: There are only 15 buffers that hold message unit request packets. The total value of the fields within this register and the registers described in **Section 16.4.1.53**, *Port n Message Request Tx Buffer Allocation Configuration Register 0 (PnMReqTxBACR0)*, on page 16-232 and **Section 16.4.1.54**, *Port n Message Request Tx Buffer Allocation Configuration Register 1 (PnMReqTxBACR1)*, on page 16-233 must not exceed 15 (undefined behavior will result).

Please see **Section 16.2.8.3**, *Buffer Allocation*, on page 16-20 for details on the use of this register. **Table 16-105** contains detailed descriptions of the fields within PnMReqTxBACR2.

Table 16-105. P*n*MReqTxBACR2 Field Descriptions

Bit	Reset	Description	Settings
— 31–28	0	Reserved. Write to zero for future compatibility.	
GenAG[3–0] 27–24	1	**Generic Arbitration Group** Controls the buffer allocation of arbitration group 0 message unit request packets.	0x0 No generic buffers allocated for any arbitration group message unit request packets. Use at own risk; this may cause deadlocks. 0x1 One generic buffer allocated for any group message unit request packets. ... 0xF Fifteen generic buffers allocated for any group message unit request packets.
— 23–0	0	Reserved. Write to zero for future compatibility.	

16.4.1.56 Port *n* Message Response/Flow Control Tx Buffer Allocation Configuration Register (P*n*MRspFcTxBACR)

P1MRspFcTxBACR Port 1–2 Message Response/Flow Control Tx Buffer Allocation Configuration Register — Offset 0x1017C
P2MRspFcTxBACR — Offset 0x101FC

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—			DedRsp [4–0]					—			DedFc [4–0]				
TYPE	R/W															
RESET	1	0	0	0	0	0	0	1	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**
	—			GenRspFc [4–0]					—							
TYPE	R/W															
RESET	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0

The Port *n* Message Response/Flow Control Tx Buffer Allocation Configuration Register 1 (P*n*MRspFcTxBACR) is used to control the buffer allocation of message unit response and flow control packets within the serial RapidIO controller transmit logic (Tx).

Note: There are only 16 buffers that hold message unit response and flow control packets. The total value of the fields within this register must not exceed 16 (undefined behavior will result).

Please see **Section 16.2.8.3**, *Buffer Allocation*, on page 16-20 for details on the use of this register. **Table 16-106** contains detailed descriptions of the fields within PnMRspFcTxBACR.

Table 16-106. P*n*MRspFcTxBACR Field Descriptions

Bit	Reset	Description	Settings
— 31–29	0	Reserved. Write to zero for future compatibility.	
DedRsp[4–0] 28–24	0x01	**Dedicated Response** Controls the buffer allocation of message unit response packets.	0x00 No dedicated buffer allocated for message unit response packets. Use at own risk; this may cause deadlocks. 0x01 One dedicated buffer allocated for message unit response packets. ... 0x10 Sixteen dedicated buffers allocated for message unit response packets.
— 23–21	0	Reserved. Write to zero for future compatibility.	
DedFc[4–0] 20–16	0x01	**Dedicated Flow Control** Controls the buffer allocation of message unit flow control packets.	0x00 No dedicated buffer allocated for message unit flow control packets. Use at own risk; this may cause deadlocks. 0x01 One dedicated buffer allocated for message unit flow control packets. ... 0x10 Sixteen dedicated buffers allocated for message unit flow control packets.

Table 16-106. P*n*MRspFcTxBACR Field Descriptions (Continued)

Bit	Reset	Description	Settings
— 15–13	0	Reserved. Write to zero for future compatibility.	
GenRspFc[3–0] 12–8	1	**Generic Response and Flow Control** Controls the buffer allocation of generic message unit response and flow control packets.	0x00 No generic buffer allocated for message unit response and flow control packets. Use at own risk; this may cause deadlocks 0x01 One dedicated buffer allocated for message unit response and flow control packets. ... 0x10 Sixteen dedicated buffers allocated for message unit response and flow control packets.
— 7–0	0	Reserved. Write to zero for future compatibility.	

16.4.1.57 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 16-107. IPBRR1 Field Descriptions

Bit	Reset	Description
IPID 31–16	0x01C0	IP block ID = 0x01C0
IPMJ 15–8	0x01	Major revision of the IP block = 0x02
IPMN 7–0	0x02	Minor revision of the IP block = 0x00

16.4.1.58 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	0

IPBRR2 track changes and revisions of the RapidIO endpoint.

Table 16-108. 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	0	IP Block Configuration Options = 0x00.

16.4.1.59 Port 1–2 RapidIO Outbound Window Translation Address Registers x (PnROWTARx)

P1ROWTAR[0–8] Port 1–2 RapidIO Outbound Window Offset 0x10C00 + x*0x20
P2ROWTAR[0–8] Translation Address Registers 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	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

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 16-109. 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 16.4.1.5**, *Processing Element Features Capability Register (PEFCAR),* on page 16-184). Bits 0–5 of the target ID are specified in the PnROWTEARX (see **Section 16.4.1.60**, *Port 1–2 RapidIO Outbound Window Translation Extended Address Registers x (PnROWTEARx),* on page 16-239
TREXAD 29–20	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.

16.4.1.60 Port 1–2 RapidIO Outbound Window Translation Extended Address Registers x (PnROWTEARx)

P1ROWTEAR[0–8] Port 1–2 RapidIO Outbound Window Translation Extended Address Registers 0–8 Offset 0x10C04 + x*0x20
P2ROWTEAR[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
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	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 16-110. PnROWTAR*x* Field Descriptions

Bit	Reset	Description
— 31–6	0	Reserved. Write to zero for future compatibility.
LTGTID 5–0	0	**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 16.4.1.5**, *Processing Element Features Capability Register (PEFCAR),* on page 16-184). Bits 6–7 of the target ID are specified in the corresponding PnROWTARx.

16.4.1.61 Port 1–2 RapidIO Outbound Window Base Address Registers x (PnROWBARx)

P1ROWBAR[1–8] Port 1–2 RapidIO Outbound Offset 0x10C08 + x*0x20
P2ROWBAR[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 16.2.9.4.2**, *Window Boundary Crossing Errors*, on page 16-56.

Table 16-111. 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.

16.4.1.62 Port 1–2 RapidIO Outbound Window Attributes Registers x (PnROWARx)

P1ROWARx Port 1–2 RapidIO Outbound Offset 0x10C10 + x*0x20
P2ROWARx 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	—		CRF	TFLOWLV		PCI	—	NSEG		NSSEG		RDTYP			
TYPE	R/W	R		R/W				R	R/W							
RESET	*	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	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1

Note: For x = 0, the reset value for the EN field = 1. For all other registers (x = 1 to 8), the reset value of the EN field = 0.

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 16-112. PnROWAR*x* Field Descriptions

Bit	Reset	Description	Settings
EN 31	0	**Enable Address Translation** For default window 0 only, this bit is set to 1 and read-only.	0 Window disabled. 1 Window enabled.
— 30–29	0	Reserved. Write to zero for future compatibility.	
CRF 28	0	**Critical Request Flow** For packets that are destined for the same output port of a switch, this bit is used to raise the precedence of the request above requests of other packets at the same flow level that have a CRF value of 0.	0 Do not raise flow precedence. 1 Raise flow precedence.
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 16-112. 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	0	Reserved. Write to zero for future compatibility.	
SIZE 5–0	100011	**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.

16.4.1.63 Port 1–2 RapidIO Outbound Window Segment 1–3 Registers 1–8 (PnROWSxRy)

Port 1–2 RapidIO Outbound Window Segment 1–3 Registers 1–8

P1ROWS[1–3]R[1–8] Offset 0x10C34 + (x–1)*0x4 + (y–1)*0x20

P2ROWS[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	0	0	0	0	0	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 16-113. 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.	

16.4.1.64 Port 1–2 RapidIO Inbound Window Translation Address Registers x (PnRIWTARx)

P1RIWTAR[0–4] Port 1–2 RapidIO Inbound Window Translation Address Registers 0–4 Offset 0x10D60 + (4–x)*0x20
P2RIWTAR[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 16-114. 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.

16.4.1.65 Port 1–2 RapidIO Inbound Window Base Address Registers x (PnRIWBARx)

P1RIWBAR[1–4] Port 1–2 RapidIO Inbound Window Base Address Registers 1–4 Offset 0x10D68 + (4–x)*0x20
P2RIWBAR[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 16.2.9.4.2**, *Window Boundary Crossing Errors*, on page 16-56.

Table 16-115. 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.

16.4.1.66 Port 1–2 RapidIO Inbound Window Attributes Registers x (PnRIWARx)

P1RIWARx Port 1–2 RapidIO Inbound Offset 0x10D70 + (4–x)*0x20
P2RIWARx Window Attributes Offset 0x10F70 + (4–x)*0x20
Registers 0–4

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 portn 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 16-116. 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.	

Table 16-116. PnRIWAR*x* Field Descriptions (Continued)

Bit	Description	Settings
— 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 1:* 0000 OCN to MBus 0 0001 OCN to MBus 1 0010 SRIO Port 1 0011 SRIO Port 2 0100–1111 Reserved. *For Port 2:* 0000 OCN to MBus 1 0001 OCN to MBus 0 0010 SRIO Port 1 0011 SRIO Port 2 0100–1111 Reserved.
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.

Note: When using the same OCN-to-MBus (O2M) for Read and Write transactions, the write transactions may write incorrect data for specific access sequences. To preclude this

scenario, use one bridge for Write transactions and the other bridge for read transactions.

16.4.2 eMSG, BMLite, and QMLite Registers

16.4.2.1 Inbound Block m Type8 Classification Unit n Mode Register (IBmT8CnMR)

The doorbell mode register allows software to enable the doorbell unit to control various doorbell operation characteristics. The IB*m*T8C*n*MR is shown in **Figure 16-42**.

Offset: B0C0: 0000; B0C1: 0100;
B0C2: 0200; B0C3: 0300;
B0C4: 0400; B0C5: 0500;
B0C6: 0600; B0C7: 0700;
B1C0–7: 1*n*00;
B2C0–7: 2*n*00;
B3C0–7:3*n*00;
B4C0–7:4*n*00;
B5C0–7:5*n*00;
B6C0–7:6*n*00;
B7C0–7:7*n*00

Access: Read/Write

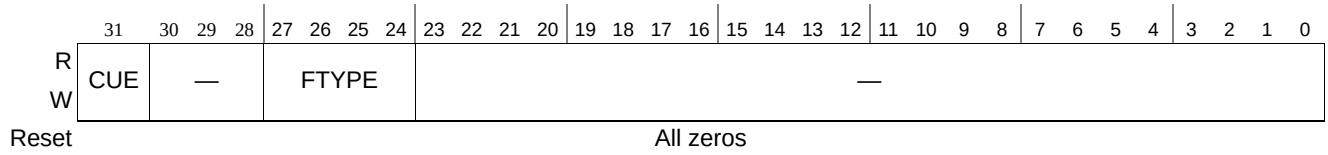


Figure 16-42. Inbound Block *m* Type8 Classification *n* Mode Register (IB*m*T8C*n*MR)

Table 16-117. IB*m*T8C*n*MR Field Descriptions

Bits	Name	Description
31	CUE	Inbound Type8 classification unit enable. 0 Disabled. 1 Inbound Type8 classification unit has been initialized and can service incoming reassembly operations.
30–28	—	Reserved
27–24	FTYPE	Type select. Determines the type of operation for this message unit. 1000 Type8 All other values reserved.
23–0	—	Reserved

16.4.2.2 Inbound Block *m* Type8 Classification *n* Status Register (IB*m*T8C*n*SR)

The status register indicates current status of the classification rule.

Offset: B0C0: 0x1E_0004; B0C1: 0x1E_0104; Access: Read-Only
B0C2: 0x1E_0204; B0C3: 0x1E_0304;
B0C4: 0x1E_0404; B0C5: 0x1E_0504;
B0C6: 0x1E_0604; B0C7: 0x1E_0704;
B1C0-7: 0x1E_1*n*04;
B2C0-7: 0x1E_2*n*04;
B3C0-7:0x1E_3*n*04;
B4C0-7:0x1E_4*n*04;
B5C0-7:0x1E_5*n*04;
B6C0-7:0x1E_6*n*04;
B7C0-7:0x1E_7*n*04

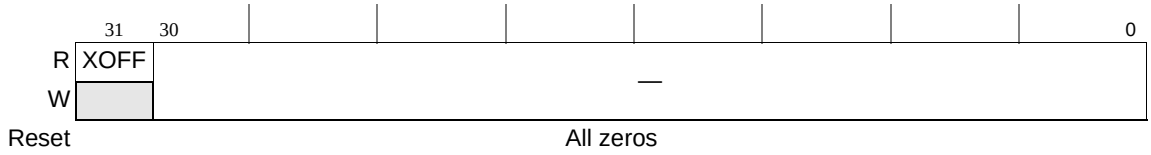


Figure 16-43. Inbound Block *m* Type8 Classification *n* Status Register (IB*m*T8C*n*SR)

Table 16-118. IB*m*T8C*n*SR Field Descriptions

Bits	Name	Description
31	XOFF	Critical flow control. 0 Classification rule not flow controlled. 1 Classification has received a critical flow control by an inbound queue, targeted by the rule, due to a queue near full condition. This bit is automatically cleared when the exit watermark is reached as defined by IB*m*MQ*n*CMR[EXIT_WM], or when the rule is disabled. Further transactions matching this rule will results in physical retry.
30–0	—	Reserved

16.4.2.3 Inbound Block *m* Type8 Classification *n* Message Queue Register

This register specifies the target message queue ID.

Offset: B0C0: 0008; B0C1: 0108;
B0C2: 0208; B0C3: 0308;
B0C4: 0408; B0C5: 0508;
B0C6: 0608; B0C7: 0708;
B1C0–7: 1*n*08;
B2C0–7: 2*n*08;
B3C0–7: 3*n*08;
B4C0–7: 4*n*08;
B5C0–7: 5*n*08;
B6C0–7: 6*n*08;
B7C0–7: 7*n*08

Access: Read/Write

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

Figure 16-44. Inbound Block *m* Type8 Classification *n* Message Queue Register (IB*m*T8C*n*MQR)

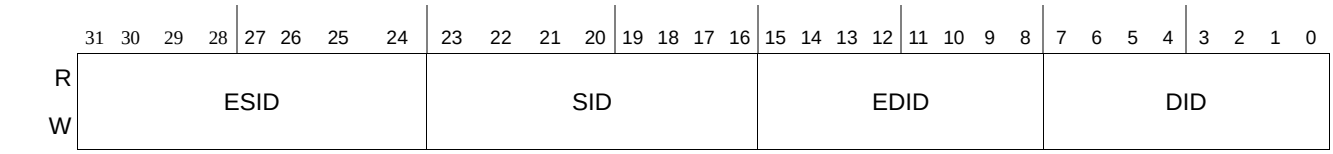
Table 16-119. IB*m*T8C*n*MQR Field Descriptions

Bits	Name	Description
31–3	—	Reserved
2–0	MQID	Message queue ID. For proper operation, this field should only be modified when the inbound classification unit is not enabled.

16.4.2.4 Inbound Block *m* Type8 Classification *n* Rule Value Register 0 (IB*m*T8C*n*RVR0) and Inbound Block *m* Type8 Classification *n* Rule Value Register 1 (IB*m*T8C*n*RVR1)

The inbound Type8 classification rule value registers are used to configure a port-write rule directing inbound port-writes to a programmable message queue. The value registers are used in conjunction with the mask registers to create the rule. For example if the source field is set to 0x07 and the corresponding mask register field is set to 0x03, the inbound unit would accept any source route values from 0x4-0x7. If multiple inbound Type8 classification units match the same port-write rules, the lowest numbered enabled inbound classification unit wins.

Offset: B0C0: 0010; B0C1: 0110; Access: Read/Write
B0C2: 0210; B0C3: 0310;
B0C4: 0410; B0C5: 0510;
B0C6: 0610; B0C7: 0710;
B1C0–7: 1*n*10;
B2C0–7: 2*n*10;
B3C0–7: 3*n*10;
B4C0–7: 4*n*10;
B5C0–7: 5*n*10;
B6C0–7: 6*n*10;
B7C0–7: 7*n*10



Reset All zeros

Figure 16-45. Inbound Block *m* Type8 Classification *n* Rule Value Register 0 (IB*m*T8C*n*RVR0)

Table 16-120. IB*m*T8C*n*RVR0 Field Descriptions

Bits	Name	Description
31–24	ESID	Extended source ID. Most significant byte of a 16-bit source ID when operating in large transport mode. Reserved when operating in small transport mode.
23–16	SID	Source ID. Contains the source ID field of the transaction (Device ID of the source).
15–8	EDID	Extended destination ID. Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
7–0	DID	Destination ID. Contains the destination ID field of the transaction (Device ID of the destination).

Offset: B0C0: 0014; B0C1: 0114;
B0C2: 0214; B0C3: 0314;
B0C4: 0414; B0C5: 0514;
B0C6: 0614; B0C7: 0714;
B1C0–7: 1n14;
B2C0–7: 2n14;
B3C0–7: 3n14;
B4C0–7: 4n14;
B5C0–7: 5n14;
B6C0–7: 6n14;
B7C0–7: 7n14

Access: Read/Write

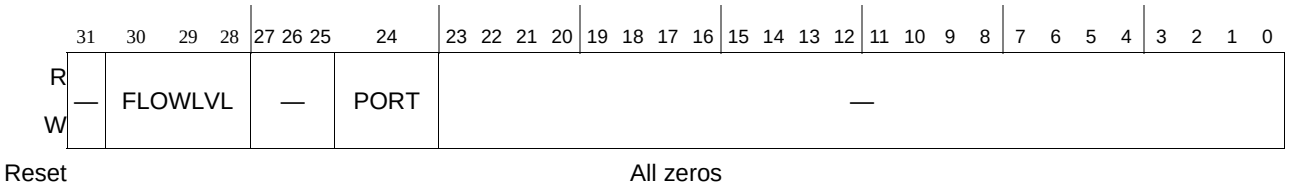


Figure 16-46. Inbound Block *m* Type8 Classification *n* Rule Value Register 1 (IB*m*T8C*n*RVR1)

Table 16-121. IB*m*T8C*n*RVR1 Field Descriptions

Bits	Name	Description
31	—	Reserved
30–28	FLOWLVL	Transaction flow level. 000 Lowest flow level 001 Next lowest flow level ... 110 Next highest flow level 111 Highest flow level
27–25	—	Reserved
24	PORT	Port number. 0 Port 1 1 Port 2
23–0	—	Reserved

16.4.2.5 Inbound Block *m* Type8 Classification *n* Rule Mask Register 0 (IB*m*T8CnRMR0) and Inbound Block *m* Type8 Classification *n* Rule Mask Register 1 (IB*m*T8CnRMR1)

The inbound Type8 classification rule mask registers are used to configure a port-write rule directing inbound port-writes to a programmable message queue. The mask registers are used in conjunction with the value registers to create the rule. Each bit set indicates a don’t care value for the inbound port-write header attribute.

Offset: B0C0: 0018; B0C1: 0118;
B0C2: 0218; B0C3: 0318;
B0C4: 0418; B0C5: 0518;
B0C6: 0618; B0C7: 0718;
B1C0–7: 1n18;
B2C0–7: 2n18;
B3C0–7: 3n18;
B4C0–7: 4n18;
B5C0–7: 5n18;
B6C0–7: 6n18;
B7C0–7: 7n18

Access: Read/Write

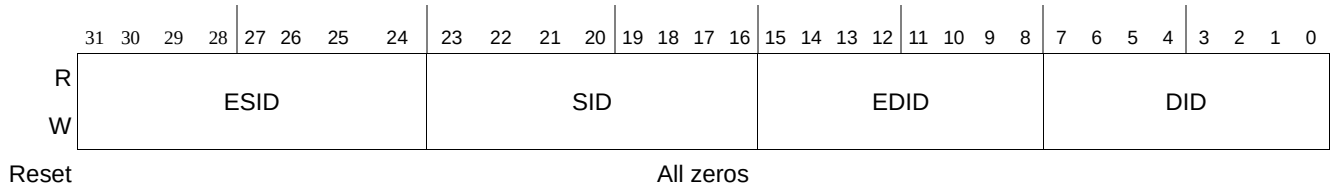


Figure 16-47. Inbound Block *m* Type8 Classification *n* Rule Mask Register 0 (IB*m*T8C*n*RMR0)

Table 16-122. IB*m*T8C*n*RMR0 Field Descriptions

Bits	Name	Description
31–24	ESID	Extended source ID bit mask- Most significant byte of a 16-bit source ID when operating in large transport mode. Reserved when operating in small transport mode.
23–16	SID	Source ID bit mask - Contains the source ID field of the transaction (Device ID of the source).
15–8	EDID	Extended destination ID bit mask - Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
7–0	DID	Destination ID bit mask - Contains the destination ID field of the transaction (Device ID of the destination).

Offset: B0C0: 001C; B0C1: 011C;
B0C2: 021C; B0C3: 031C;
B0C4: 041C; B0C5: 051C;
B0C6: 061C; B0C7: 071C;
B1C0–7: 1n1C;
B2C0–7: 2n1C;
B3C0–7: 3n1C;
B4C0–7: 4n1C;
B5C0–7: 5n1C;
B6C0–7: 6n1C;
B7C0–7: 7n1C

Access: Read/Write

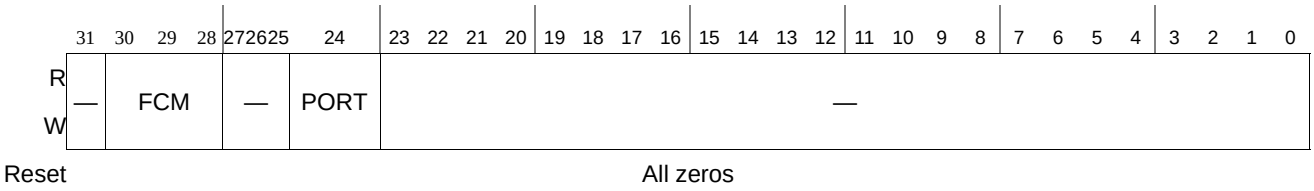


Figure 16-48. Inbound Block m Type8 Classification *n* Rule Mask Register 1 (IB*m*T8C*n*RMR1)

Table 16-123. IB*m*T8C*n*RMR1 Field Descriptions

Bits	Name	Description
31–30	—	Reserved
29–28	FCM	Flow conditional match. 00 Exact match on flow level 01 Match on less than or equal to flow level value 10 Match on greater than or equal to flow level value 11 Reserved
27–25	—	Reserved
24	PORT	Port number. 0 Port 1 1 Port 2
23–0	—	Reserved

16.4.2.6 Inbound Block *m* Type8 Classification *n* Data Buffer Pool Register (IB*m*T8C*n*DBPR)

This is the data buffer pool register, for inbound Type8 classification, used for requesting data buffers from the Buffer Manager. During the process of enqueueing inbound Type8, data buffers are prefetched from the assigned buffer pool for storage of data.

Offset: B0C0: 0030; B0C1: 0130;
B0C2: 0230; B0C3: 0330;
B0C4: 0430; B0C5: 0530;
B0C6: 0630; B0C7: 0730;
B1C0–7: 1n30;
B2C0–7: 2n30;
B3C0–7: 3n30;
B4C0–7: 4n30;
B5C0–7: 5n30;
B6C0–7: 6n30;
B7C0–7: 7n30

Access: Read/Write

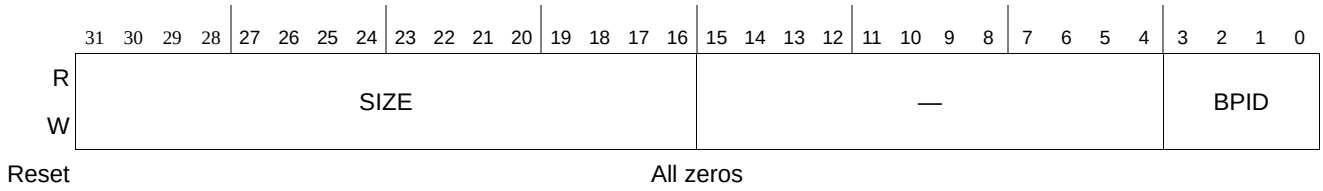


Figure 16-49. Inbound Block *m* Type8 Classification *n* Data Buffer Pool Register (IB*m*T8C*n*DBPR)

Table 16-124. IB*m*T8C*n*DBPR Field Descriptions

Bits	Name	Description
31–16	SIZE	Buffer size. Indicates the buffer size for the Buffer Pool used. 0x0000 64K bytes 0x0040 64 bytes 0x0041 65 bytes ... 0x0080 128 bytes 0x0081 129 bytes ... 0x0100 256 bytes ... 0xFFFF64K bytes - 1 Minimum buffer size is 64 bytes. Although larger data buffers are supported, port-write data payload is maximum 64 bytes.
15–4	—	Reserved.
3–0	BPID	Buffer Pool ID. Statically assigned buffer pool identifier for requesting free storage.

16.4.2.7 Inbound Block *m* Type8 Classification *n* Data Offset Register (IB*m*T8C*n*DOR)

This is the Type8 data offset register. The offset specified here indicates where the first byte of the data is located within the buffer. No descriptor is written for inbound Type8, thus there is no minimum offset requirement.

Offset: B0C0: 0034; B0C1: 0134;
B0C2: 0234; B0C3: 0334;
B0C4: 0434; B0C5: 0534;
B0C6: 0634; B0C7: 0734;
B1C0–7: 1n34;
B2C0–7: 2n34;
B3C0–7: 3n34;
B4C0–7: 4n34;
B5C0–7: 5n34;
B6C0–7: 6n34;
B7C0–7: 7n34

Access: Read/Write

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

Figure 16-50. Inbound Block *m* Type8 Classification *n* Data Offset Register (IB*m*T8C*n*DOR)

Table 16-125. IB*m*T8C*n*DOR Field Descriptions

Bits	Name	Description
31–9	—	Reserved
8–0	OFFSET	Byte offset from buffer starting address to where the first byte of data is located.

16.4.2.8 Inbound Block *m* Type9 Classification *n* Mode Registers (IB*m*T9C*n*MR)

The inbound Type9 classification mode register allows software to enable the classification unit and to control various reassembly operation characteristics.

Offset:B0C0: 0000; B0C1: 0100; Access: Read/Write
B0C2: 0200; B0C3: 0300;
B0C4: 0400; B0C5: 0500;
B0C6: 0600; B0C7: 0700;
B1C0–7: 1*n*00;
B2C0–7: 2*n*00;
B3C0–7:3*n*00;
B4C0–7:4*n*00;
B5C0–7:5*n*00;
B6C0–7:6*n*00;
B7C0–7:7*n*00

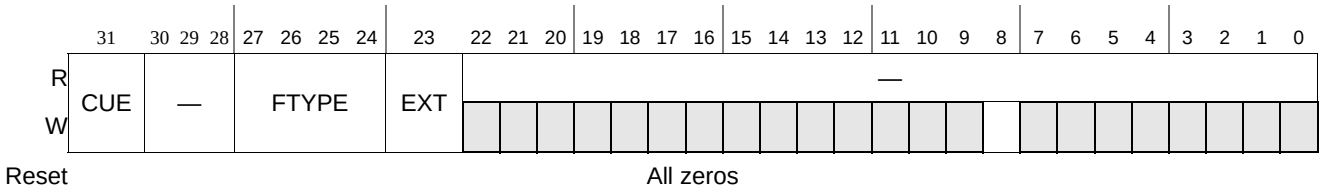


Figure 16-51. Inbound Block *m* Type9 Classification *n* Mode Registers (IB*m*T9C*n*MR)

Table 16-126. IB*m*T9C*n*MR Field Descriptions

Bits	Name	Description
31	CUE	Inbound Type9 classification unit enable. 0 Disabled. 1 Inbound Type9 classification unit has been initialized and can service incoming reassembly operations.
30–28	—	Reserved
27–24	FTYPE	Type select. Determines the type of operation for this message unit. 1001 Type9 All other values reserved.
23	EXT	Extended Type 0 Normal. 1 Flow control.
22–0	—	Reserved

16.4.2.9 Inbound Block *m* Type9 Classification *n* Status Register (IB*m*T9C*n*SR)

The status register indicates current status of the classification rule.

Offset: B0C0: 0x1E_0004; B0C1: 0x1E_0104; Access: Read-Only
B0C2: 0x1E_0204; B0C3: 0x1E_0304;
B0C4: 0x1E_0404; B0C5: 0x1E_0504;
B0C6: 0x1E_0604; B0C7: 0x1E_0704;
B1C0-7: 0x1E_1*n*04;
B2C0-7: 0x1E_2*n*04;
B3C0-7:0x1E_3*n*04;
B4C0-7:0x1E_4*n*04;
B5C0-7:0x1E_5*n*04;
B6C0-7:0x1E_6*n*04;
B7C0-7:0x1E_7*n*04

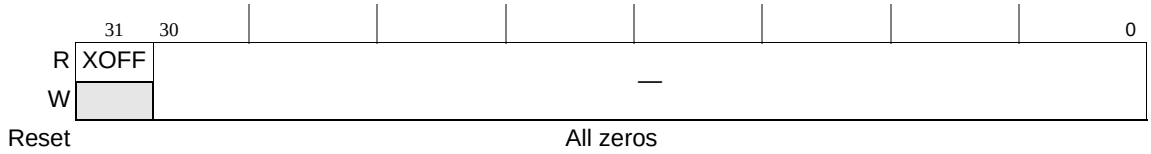


Figure 16-52. Inbound Block *m* Type9 Classification *n* Status Register (IB*m*T9C*n*SR)

Table 16-127. IB*m*T9C*n*SR Field Descriptions

Bits	Name	Description
31	XOFF	Critical flow control. 0 Classification rule not flow controlled. 1 Classification has received a critical flow control by an inbound queue, targeted by the rule, due to a queue near full condition. This bit is automatically cleared when the exit watermark is reached as defined by IB*m*MQ*n*CMR[EXIT_WM], or when the rule is disabled.
30–0	—	Reserved

16.4.2.10 Inbound Block *m* Type9 Classification *n* Message Queue Register

This register specifies the message queue ID.

Offset: B0C0: 0008; B0C1: 0108; Access: Read/Write
B0C2: 0208; B0C3: 0308;
B0C4: 0408; B0C5: 0508;
B0C6: 0608; B0C7: 0708;
B1C0–7: 1*n*08;
B2C0–7: 2*n*08;
B3C0–7:3*n*08;
B4C0–7:4*n*08;
B5C0–7:5*n*08;
B6C0–7:6*n*08;
B7C0–7:7*n*08

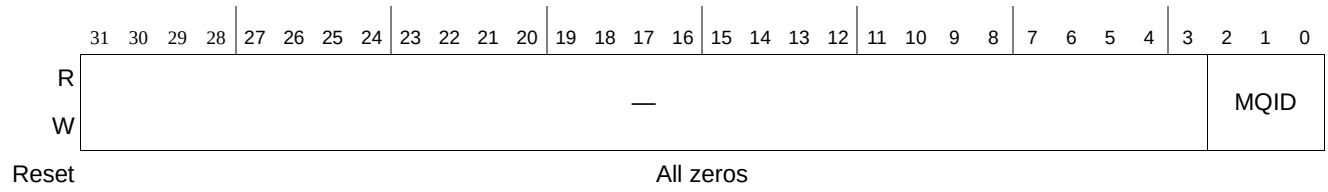


Figure 16-53. Inbound Block *m* Type9 Classification *n* Message Queue Register (IB*m*T9C*n*MQR)

Table 16-128. IB*m*T9C*n*MQR Field Descriptions

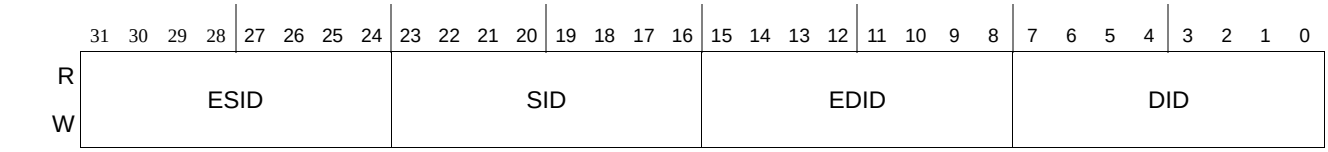
Bits	Name	Description
31–3	—	Reserved
2–0	MQID	Message queue ID. For proper operation, this field should only be modified when the inbound classification unit is not enabled.

16.4.2.11 Inbound Block *m* Type9 Classification *n* Rule Value Register 0 (IB*m*T9C*n*RVR0) and Inbound Block *m* Type9 Classification *n* Rule Value Register 1 (IB*m*T9C*n*RVR1)

The inbound Type9 classification value registers are used to configure a data streaming rule directing inbound PDUs to a programmable message queue. The value register is used in conjunction with the mask register to create the rule. For example if the class-of-service field is set to 0x20 and the corresponding mask register field is set to 0x07, the inbound Type9 classification unit would accept any class-of-service values from 0x20-0x25. If multiple classification units match the same packet attributes, the lowest numbered enabled inbound classification unit wins.

Offset: B0C0: 0010; B0C1: 0110;
B0C2: 0210; B0C3: 0310;
B0C4: 0410; B0C5: 0510;
B0C6: 0610; B0C7: 0710;
B1C0–7: 1*n*10;
B2C0–7: 2*n*10;
B3C0–7: 3*n*10;
B4C0–7: 4*n*10;
B5C0–7: 5*n*10;
B6C0–7: 6*n*10;
B7C0–7: 7*n*10

Access: Read/Write



Reset: All zeros

Figure 16-54. Inbound Block *m* Type9 Classification *n* Rule Value Register 0 (IB*m*T9C*n*RVR0)

Table 16-129. IB*m*T9C*n*RVR0 Field Descriptions

Bits	Name	Description
31–24	ESID	Extended source ID. Most significant byte of a 16-bit source ID when operating in large transport mode. Reserved when operating in small transport mode.
23–16	SID	Source ID. Contains the source ID field of the transaction (Device ID of the source).
15–8	EDID	Extended destination ID. Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
7–0	DID	Destination ID. Contains the destination ID field of the transaction (Device ID of the destination).

Offset B0C0: 0014; B0C1: 0114; Access: Read/Write
: B0C2: 0214; B0C3: 0314;
B0C4: 0414; B0C5: 0514;
B0C6: 0614; B0C7: 0714;
B1C0–7: 1n14;
B2C0–7: 2n14;
B3C0–7: 3n14;
B4C0–7: 4n14;
B5C0–7: 5n14;
B6C0–7: 6n14;
B7C0–7: 7n14

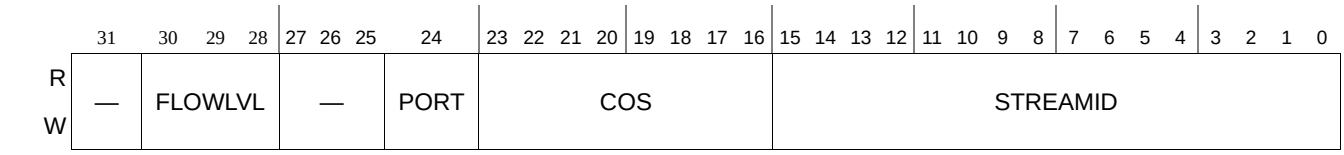


Figure 16-55. Inbound Block *m* Type9 Classification *n* Rule Value Register 1 (IB*m*T9C*n*RVR1)

Table 16-130. IB*m*T9C*n*RVR1 Field Descriptions

Bits	Name	Description
31	—	Reserved
30–28	FLOWLVL	Transaction flow level. 000 Lowest flow level 001 Next lowest flow level ... 110 Next highest flow level 111 Highest flow level
27–25	—	Reserved
24	PORT	Port number. 0 Port 1 1 Port 2
23–16	COS	Class-of-Service field value from the start or single packet header
15–0	STREAMID	StreamID field value from the start or single packet header

16.4.2.12 Inbound Block *m* Type9 Classification *n* Rule Mask Register 0 (IB*m*T9C*n*RMR) and Inbound Block *m* Type9 Classification *n* Rule Mask Register 1 (IB*m*T9C*n*RMR1)

The inbound Type9 classification mask registers are used to configure a packet rule directing inbound PDUs to a programmable message queue. The mask registers are used in conjunction with the value registers to create the rule. Each bit set indicates a don’t care value for the inbound packet header attribute.

Offset: B0C0: 0018; B0C1: 0118;
B0C2: 0218; B0C3: 0318;
B0C4: 0418; B0C5: 0518;
B0C6: 0618; B0C7: 0718;
B1C0–7: 1n18;
B2C0–7: 2n18;
B3C0–7: 3n18;
B4C0–7: 4n18;
B5C0–7: 5n18;
B6C0–7: 6n18;
B7C0–7: 7n18

Access: Read/Write

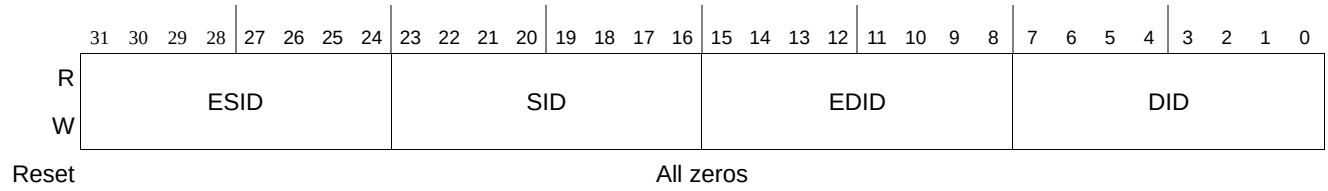


Figure 16-56. Inbound Block *m* Type9 Classification *n* Rule Mask Register 0 (IB*m*T9C*n*RMR0)

Table 16-131. IB*m*T9C*n*RMR0 Field Descriptions

Bits	Name	Description
31–24	ESID	Extended source ID mask - Most significant byte of a 16-bit source ID when operating in large transport mode. Reserved when operating in small transport mode. Ignored when Type9 flow control management is enabled (DSLLCCSR).
23–16	SID	Source ID mask - Contains the source ID field of the transaction (Device ID of the source). Ignored when Type9 flow control management is enabled (DSLLCCSR).
15–8	EDID	Extended destination ID mask - Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
7–0	DID	Destination ID mask - Contains the destination ID field of the transaction (Device ID of the destination).

Offset : B0C0: 001C; B0C1: 011C;
B0C2: 021C; B0C3: 031C;
B0C4: 041C; B0C5: 051C;
B0C6: 061C; B0C7: 071C;
B1C0–7: 1n1C;
B2C0–7: 2n1C;
B3C0–7: 3n1C;
B4C0–7: 4n1C;
B5C0–7: 5n1C;
B6C0–7: 6n1C;
B7C0–7: 7n1C

Access: Read/Write

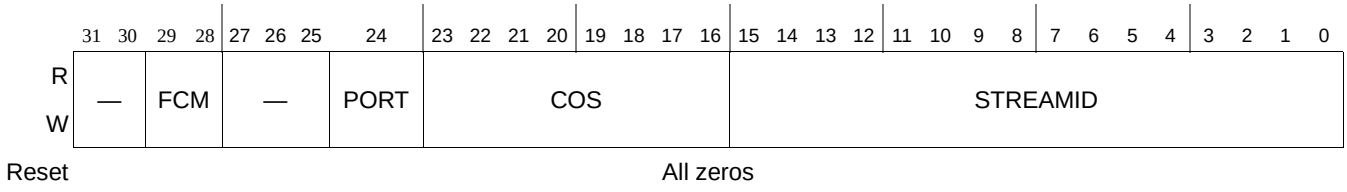


Figure 16-57. Inbound Block *m* Type9 Classification *n* Rule Mask Register 1 (IB*m*T9C*n*RMR1)

Table 16-132. IB*m*T9C*n*RMR1 Field Descriptions

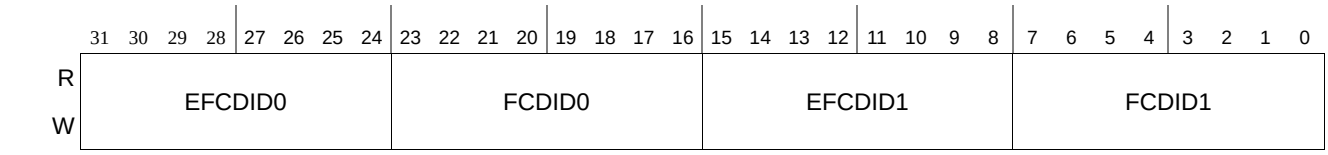
Bits	Name	Description
31–30	—	Reserved
29–28	FCM	Flow conditional match. 00 Exact match on flow level 01 Match on less than or equal to flow level value 10 Match on greater than or equal to flow level value 11 Reserved
27–25	—	Reserved
24	PORT	Port number. 0 Port 1 1 Port 2
23–16	COS	Mask for Class-of-Service field value from the start or single packet header. Mask is left justified to identify specific class bits as don't cares. MaskClass 0b000000000bnnnnnnnn256 classes n = valid class bits 0b000000010bnnnnnnnx128 classes x = don't cares 0b000000110bnnnnnnxx64 classes 0b000001110bnnnnnxxx32 classes 0b000011110bnnnnxxxx16 classes 0b000111110bnnnxxxxx8 classes 0b001111110bnnxxxxxx4 classes 0b011111110bnxxxxxxx2 classes 0b111111110bxxxxxxxx1 class
15–0	STREAMID	Mask for StreamID field value from the start or single packet header

16.4.2.13 Inbound Block *m* Type9 Classification *n* Flow Control Destination Register (IB*m*T9C*n*FCDR)

This is the flow control destination register, for inbound Type9 classification. When an inbound message queue receiving a Type9 PDUs enters or exist flow control, a RapidIO stream management flow control request is sent to the source or sources generating traffic to the message queue. The destination register ID should match the possible sources for the classification rule. If multiple sources are included (non-zero mask), the destination ID should reflect a multicast destination ID for the RapidIO switch.

Offset: B0C0: 0020; B0C1: 0120;
B0C2: 0220; B0C3: 0320;
B0C4: 0420; B0C5: 0520;
B0C6: 0620; B0C7: 0720;
B1C0–7: 1n20;
B2C0–7: 2n20;
B3C0–7: 3n20;
B4C0–7: 4n20;
B5C0–7: 5n20;
B6C0–7: 6n20;
B7C0–7: 7n20

Access: Read/Write



Reset: All zeros

Figure 16-58. Inbound Block *m* Type9 Classification *n* Flow Control Destination Register (IB*m*T9C*n*FCDR)

Table 16-133. IB*m*T9C*n*FCDR Field Descriptions

Bits	Name	Description
31–24	EFCDID1	Extended flow control destination ID for RapidIO port 1. Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode. Indicates the destination ID for stream management flow control to RapidIO sources on port 1 generating traffic to message queue specified in IB*m*T9C*n*MQR[MQID] which has entered or exited congestion. Destination ID may indicate a multicast target.
23–16	FCDID1	Flow control destination ID for RapidIO port 1.
15–8	EFCDID2	Extended flow control destination ID for RapidIO port 2.
7–0	FCDID2	Flow control destination ID for RapidIO port 2.

16.4.2.14 Inbound Block *m* Type9 Classification *n* Data Buffer Pool Register (IB*m*T9C*n*DBPR)

This is the data buffer pool register, for Type9 inbound PDUs, used for requesting buffers from the Buffer Manager. During the process of enqueueing inbound Type9 descriptors, buffers are prefetched from the assigned buffer pool for gathering of data.

Offset: B0C0: 0030; B0C1: 0130;
B0C2: 0230; B0C3: 0330;
B0C4: 0430; B0C5: 0530;
B0C6: 0630; B0C7: 0730;
B1C0–7: 1n30;
B2C0–7: 2n30;
B3C0–7: 3n30;
B4C0–7: 4n30;
B5C0–7: 5n30;
B6C0–7: 6n30;
B7C0–7: 7n30

Access: Read/Write

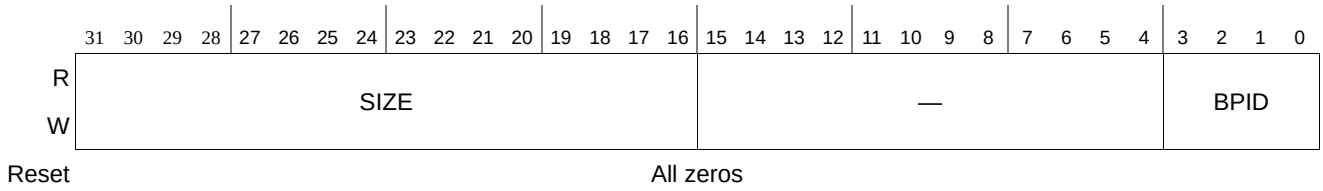


Figure 16-59. Inbound Block *m* Type9 Classification *n* Data Buffer Pool Register (IB*m*T9C*n*DBPR)

Table 16-134. IB*m*T9C*n*DBPR Field Descriptions

Bits	Name	Description
31–16	SIZE	Buffer size. Indicates the buffer size for the Buffer Pool used. 0x0000 64 K bytes 0x0040 64 bytes 0x0041 65 bytes ... 0x0080 128 bytes 0x0081 129 bytes ... 0x0100 256 bytes ... 0xFFFF 64K bytes – 1 Minimum buffer size is 64 bytes. **Note:** If 0x0000 is selected, the inbound buffer size is interpreted as 0 bytes and not 64 Kbytes. Therefore, do not select this value. If the buffer size is 64 Kbytes, the software can set IBmT9CnhDBPR[SIZE] to anything less than 64 Kbyte, but larger than the required message size to avoid a buffer size error.
15–4	—	Reserved.
3–0	BPID	Buffer Pool ID. Statically assigned buffer pool identifier for requesting free storage.

16.4.2.15 Inbound Block *m* Type9 Classification *n* Data Offset Register (IB*m*T9C*n*DOR)

This is the Type9 descriptor data offset register. Inbound transactions of Type9 writes the descriptor information into byte zero 0 of the first allocated data buffer. The offset specified here

indicates where the first byte of the data payload is located within the buffer. To allow for the descriptor, the minimum offset must be 32-bytes.

Offset: B0C0: 0034; B0C1: 0134;
B0C2: 0234; B0C3: 0334;
B0C4: 0434; B0C5: 0534;
B0C6: 0634; B0C7: 0734;
B1C0–7: 1n34;
B2C0–7: 2n34;
B3C0–7: 3n34;
B4C0–7: 4n34;
B5C0–7: 5n34;
B6C0–7: 6n34;
B7C0–7: 7n34

Access: Read/Write

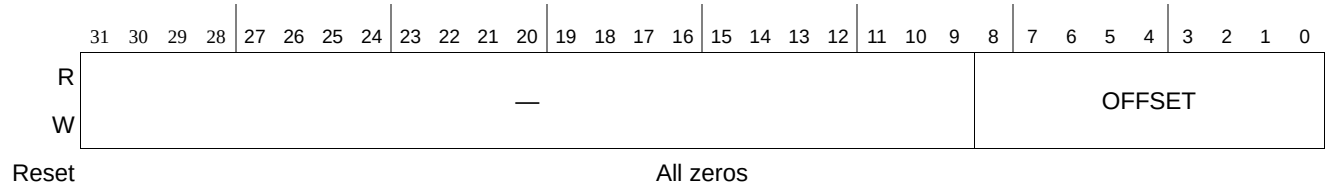


Figure 16-60. Inbound Block *m* Type9 Classification *n* Data Offset Register (IB*m*T9C*n*DOR)

Table 16-135. IB*m*T9C*n*DOR Field Descriptions

Bits	Name	Description
31–9	—	Reserved
8–0	OFFSET	Byte offset from buffer starting address to where the first byte of data payload is located.

16.4.2.16 Inbound Block *m* Type9 Classification *n* Scatter/Gather Buffer Pool Register (IB*m*T9C*n*SGBPR)

This is the scatter/gather buffer pool register, used for requesting data buffers from the Buffer Manager. During the process of enqueueing transaction onto the message queue, data buffers are prefetched from the assigned buffer pool for building scatter/gather tables.

Offset: B0C0: 0038; B0C1: 0138;
B0C2: 0238; B0C3: 0338;
B0C4: 0438; B0C5: 0538;
B0C6: 0638; B0C7: 0738;
B1C0–7: 1n38;
B2C0–7: 2n38;
B3C0–7: 3n38;
B4C0–7: 4n38;
B5C0–7: 5n38;
B6C0–7: 6n38;
B7C0–7: 7n38

Access: Read/Write

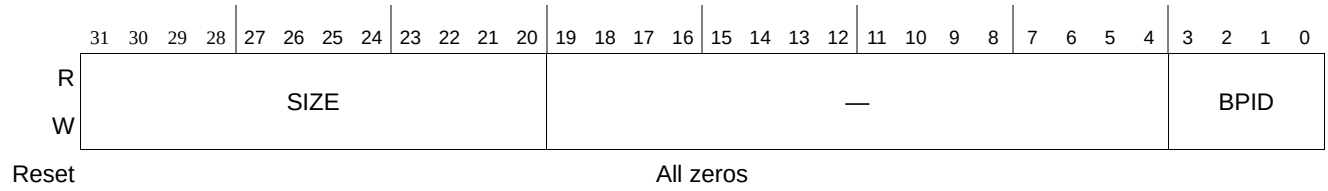


Figure 16-61. Inbound Block *m* Type9 Classification *n* Scatter/Gather Buffer Pool Register (IB*m*T9C*n*SGBPR)

Table 16-136. IB*m*T9C*n*SGBPR Field Descriptions

Bits	Name	Description
31–20	SIZE	Buffer size. Indicates the buffer size for the Buffer Pool used. 0x000 64K bytes 0x004 64 bytes 0x005 80 bytes ... 0x008 128 bytes 0x009 144 bytes ... 0x010 256 bytes 0x011 272 bytes ... 0xFFF 64K bytes - 16 Minimum buffer size is 64 bytes. The 4 least significant bytes in the size field are unused since the S/G record size is 16 bytes **Note:** If 0x000 is selected, the inbound buffer size is interpreted as 0 bytes and not 64 Kbytes. Therefore, do not select this value. Normally, scatter/gather data buffers are much smaller than 64Kbytes. If the actual data payload size is 64 Kbytes, software must also account for the descriptor size, which will not fit in a single 64 Kbyte buffer. The software can set IBmT9CnSGBPR[SIZE] to indicate anything less than 64Kbyte, but larger than the required scatter/gather size to avoid a buffer size error.
19–4	—	Reserved.
3–0	BPID	Buffer Pool ID. Statically assigned buffer pool identifier for requesting free storage.

16.4.2.17 Inbound Block *m* Type10 Classification *n* Mode Register (IB*m*T10C*n*MR)

The doorbell mode register allows software to enable the doorbell unit to control various doorbell operation characteristics. The IB*m*T10C*n*MR is shown in **Figure 16-62**.

Offset: B0C0: 0000; B0C1: 0100; Access: Read/Write
B0C2: 0200; B0C3: 0300;
B0C4: 0400; B0C5: 0500;
B0C6: 0600; B0C7: 0700;
B1C0–7: 1*n*00;
B2C0–7: 2*n*00;
B3C0–7:3*n*00;
B4C0–7:4*n*00;
B5C0–7:5*n*00;
B6C0–7:6*n*00;
B7C0–7:7*n*00

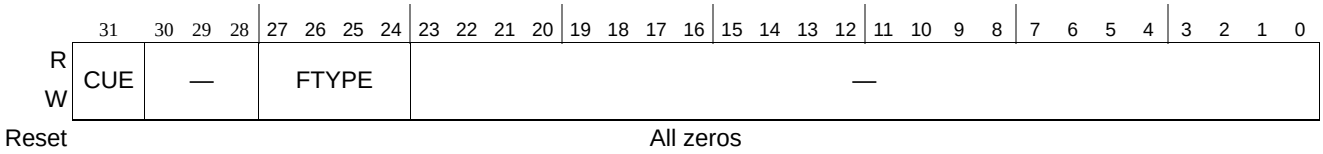


Figure 16-62. Inbound Block *m* Type10 Classification *n* Mode Register (IB*m*T10C*n*MR)

Table 16-137. IB*m*T10C*n*MR Field Descriptions

Bits	Name	Description
31	CUE	Inbound Type10 classification unit enable. 0 Disabled. 1 Inbound Type10 classification unit has been initialized and can service incoming reassembly operations.
30–28	—	Reserved
27–24	FTYPE	Type select. Determines the type of operation for this message unit. 1010 Type10 All other values reserved.
23–0	—	Reserved

16.4.2.18 Inbound Block *m* Type10 Classification *n* Status Register (IB*m*T10C*n*SR)

The status register indicates current status of the classification rule.

Offset: B0C0: 0x1E_0004; B0C1: 0x1E_0104; Access: Read-Only
B0C2: 0x1E_0204; B0C3: 0x1E_0304;
B0C4: 0x1E_0404; B0C5: 0x1E_0504;
B0C6: 0x1E_0604; B0C7: 0x1E_0704;
B1C0-7: 0x1E_1*n*04;
B2C0-7: 0x1E_2*n*04;
B3C0-7:0x1E_3*n*04;
B4C0-7:0x1E_4*n*04;
B5C0-7:0x1E_5*n*04;
B6C0-7:0x1E_6*n*04;
B7C0-7:0x1E_7*n*04

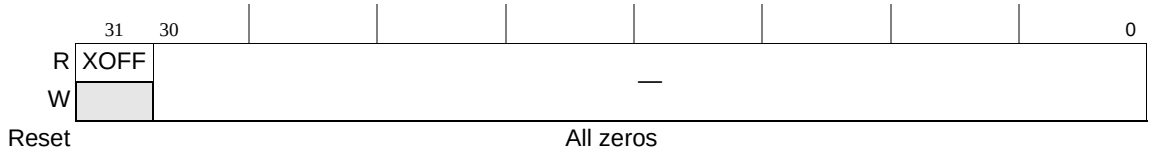


Figure 16-63. Inbound Block *m* Type10 Classification *n* Status Register (IB*m*T10C*n*SR)

Table 16-138. IB*m*T10C*n*SR Field Descriptions

Bits	Name	Description
31	XOFF	Critical flow control. 0 Classification rule not flow controlled. 1 Classification has received a critical flow control by an inbound queue, targeted by the rule, due to a queue near full condition. This bit is automatically cleared when the exit watermark is reached as defined by IB*m*MQ*n*CMR[EXIT_WM], or when the rule is disabled.
30–0	—	Reserved

16.4.2.19 Inbound Block *m* Type10 Classification *n* Message Queue Register

This register specifies the message queue ID.

Offset: B0C0: 0008; B0C1: 0108;
B0C2: 0208; B0C3: 0308;
B0C4: 0408; B0C5: 0508;
B0C6: 0608; B0C7: 0708;
B1C0–7: 1*n*08;
B2C0–7: 2*n*08;
B3C0–7: 3*n*08;
B4C0–7: 4*n*08;
B5C0–7: 5*n*08;
B6C0–7: 6*n*08;
B7C0–7: 7*n*08

Access: Read/Write

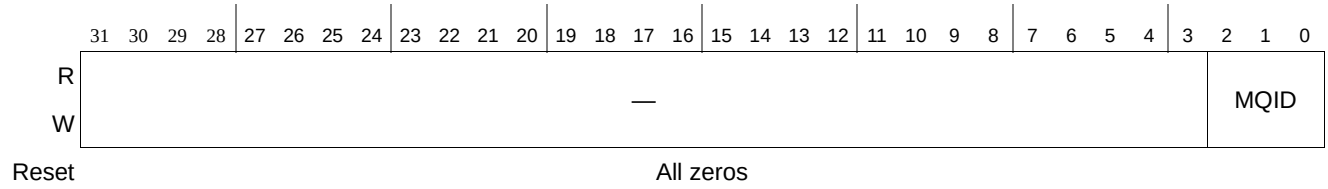


Figure 16-64. Inbound Block *m* Type10 Classification *n* Message Queue Register (IB*m*T10C*n*MQR)

Table 16-139. IB*m*T10C*n*MQR Field Descriptions

Bits	Name	Description
31–3	—	Reserved
2–0	MQID	Message queue ID. For proper operation, this field should only be modified when the inbound classification unit is not enabled.

16.4.2.20 Inbound Block *m* Type10 Classification *n* Rule Value Register 0 (IB*m*T10C*n*RVR0) and Inbound Block *m* Type10 Classification *n* Rule Value Register 1 (IB*m*T10C*n*RVR1)

The inbound Type10 classification rule value registers are used to configure a doorbell rule directing inbound doorbells to a programmable message queue. The value registers are used in conjunction with the mask registers to create the rule. For example if the source field is set to 0x07 and the corresponding mask register field is set to 0x03, the inbound unit would accept any

source route values from 0x4-0x7. If multiple inbound Type10 classification units match the same doorbell rules, the lowest numbered enabled inbound classification unit wins.

Offset: B0C0: 0010; B0C1: 0110;
B0C2: 0210; B0C3: 0310;
B0C4: 0410; B0C5: 0510;
B0C6: 0610; B0C7: 0710;
B1C0–7: 1*n*10;
B2C0–7: 2*n*10;
B3C0–7: 3*n*10;
B4C0–7: 4*n*10;
B5C0–7: 5*n*10;
B6C0–7: 6*n*10;
B7C0–7: 7*n*10

Access: Read/Write

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

Figure 16-65. Inbound Block *m* Type10 Classification *n* Rule Value Register 0 (IB*m*T10C*n*RVR0)

Table 16-140. IB*m*T10C*n*RVR0 Field Descriptions

Bits	Name	Description
31–24	ESID	Extended source ID - Most significant byte of a 16-bit source ID when operating in large transport mode. Reserved when operating in small transport mode.
23–16	SID	Source ID - Contains the source ID field of the transaction (Device ID of the source).
15–8	EDID	Extended destination ID - Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
7–0	DID	Destination ID - Contains the destination ID field of the transaction (Device ID of the destination).

Offset : B0C0: 0014; B0C1: 0114; B0C2: 0214; B0C3: 0314; B0C4: 0414; B0C5: 0514; B0C6: 0614; B0C7: 0714; B1C0–7: 1n14; B2C0–7: 2n14; B3C0–7: 3n14; B4C0–7: 4n14; B5C0–7: 5n14; B6C0–7: 6n14; B7C0–7: 7n14

Access: Read/Write

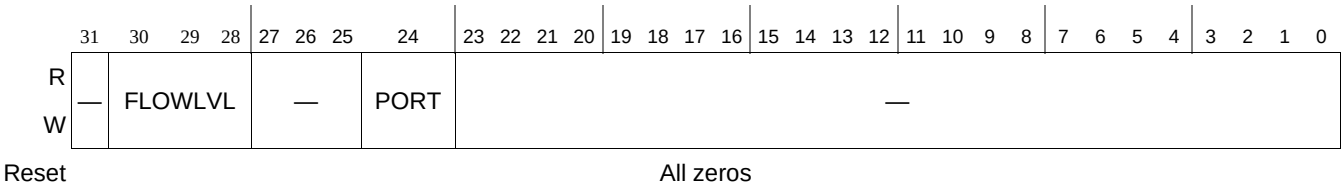


Figure 16-66. Inbound Block *m* Type10 Classification *n* Rule Value Register 1 (IB*m*T10C*n*RVR1)

Table 16-141. IB*m*T10C*n*RVR1 Field Descriptions

Bits	Name	Description
31	—	Reserved
30–28	FLOWLVL	Transaction flow level. 000 Lowest flow level 001 Next lowest flow level ... 110 Next highest flow level 111 Highest flow level
27–25	—	Reserved
24	PORT	Port number. 0 Port 1 1 Port 2
23–0	—	Reserved

16.4.2.21 Inbound Block *m* Type10 Classification *n* Rule Mask Register 0 (IB*m*T10CnRMR0) and Inbound Block *m* Type10 Classification *n* Rule Mask Register 1 (IB*m*T10CnRMR1)

The inbound Type10 classification rule mask registers are used to configure a doorbell rule directing inbound doorbells to a programmable message queue. The mask registers are used in conjunction with the value registers to create the rule. Each bit set indicates a don’t care value for the inbound doorbell header attribute.

Offset: B0C0: 0018; B0C1: 0118;
B0C2: 0218; B0C3: 0318;
B0C4: 0418; B0C5: 0518;
B0C6: 0618; B0C7: 0718;
B1C0–7: 1n18;
B2C0–7: 2n18;
B3C0–7: 3n18;
B4C0–7: 4n18;
B5C0–7: 5n18;
B6C0–7: 6n18;
B7C0–7: 7n18

Access: Read/Write

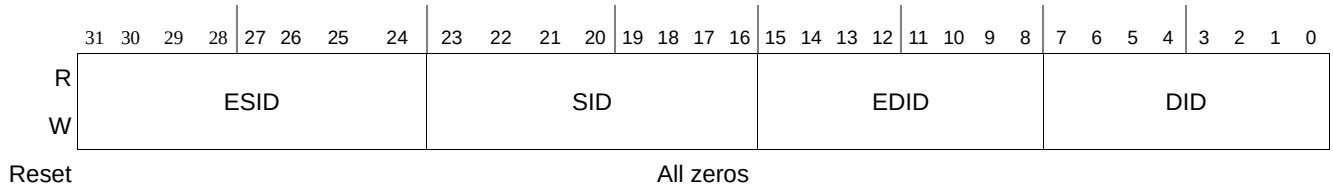
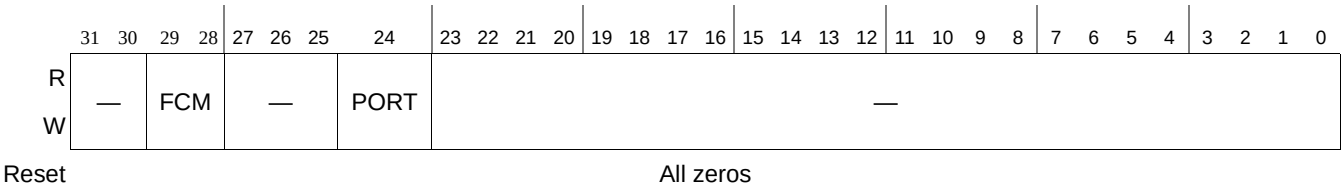


Figure 16-67. Inbound Block *m* Type10 Classification *n* Rule Mask Register 0 (IB*m*T10C*n*RMR0)

Table 16-142. IB*m*T10C*n*RMR0 Field Descriptions

Bits	Name	Description
31–24	ESID	Extended source ID bit mask- Most significant byte of a 16-bit source ID when operating in large transport mode. Reserved when operating in small transport mode.
23–16	SID	Source ID bit mask - Contains the source ID field of the transaction (Device ID of the source).
15–8	EDID	Extended destination ID bit mask - Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
7–0	DID	Destination ID bit mask - Contains the destination ID field of the transaction (Device ID of the destination).

Offset B0C0: 001C; B0C1: 011C; Access: Read/Write
: B0C2: 021C; B0C3: 031C;
B0C4: 041C; B0C5: 051C;
B0C6: 061C; B0C7: 071C;
B1C0–7: 1n1C;
B2C0–7: 2n1C;
B3C0–7: 3n1C;
B4C0–7: 4n1C;
B5C0–7: 5n1C;
B6C0–7: 6n1C;
B7C0–7: 7n1C



Reset All zeros

Figure 16-68. Inbound Block m Type10 Classification *n* Rule Mask Register 1 (IB*m*T10C*n*RMR1)

Table 16-143. IB*m*T10*n*RMR1 Field Descriptions

Bits	Name	Description
31–30	—	Reserved
29–28	FCM	Flow conditional match. 00 Exact match on flow level 01 Match on less than or equal to flow level value 10 Match on greater than or equal to flow level value 11 Reserved
27–25	—	Reserved
24	PORT	Port number. 0 Port 1 1 Port 2
23–0	—	Reserved

16.4.2.22 Inbound Block *m* Type10 Classification *n* Data Buffer Pool Register (IB*m*T10C*n*DBPR)

This is the data buffer pool register, for inbound Type10 classification, used for requesting data buffers from the Buffer Manager. During the process of enqueueing inbound Type10 descriptors, data buffers are prefetched from the assigned buffer pool for storage of data.

Offset: B0C0: 0030; B0C1: 0130;
B0C2: 0230; B0C3: 0330;
B0C4: 0430; B0C5: 0530;
B0C6: 0630; B0C7: 0730;
B1C0–7: 1n30;
B2C0–7: 2n30;
B3C0–7: 3n30;
B4C0–7: 4n30;
B5C0–7: 5n30;
B6C0–7: 6n30;
B7C0–7: 7n30

Access: Read/Write

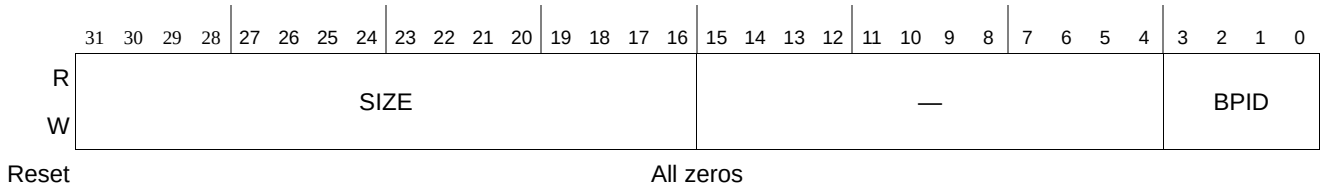


Figure 16-69. Inbound Block *m* Type10 Classification *n* Data Buffer Pool Register (IB*m*T10C*n*DBPR)

Table 16-144. IB*m*T10C*n*DBPR Field Descriptions

Bits	Name	Description
31–16	SIZE	Buffer size. Indicates the buffer size for the Buffer Pool used. 0x0000 64K bytes 0x0040 64 bytes 0x0041 65 bytes ... 0x0080 128 bytes 0x0081 129 bytes ... 0x0100 256 bytes ... 0xFFFF64K bytes - 1 Minimum buffer size is 64 bytes. Although larger data buffers are supported, doorbell data payload is maximum 2 bytes plus descriptor size.
15–4	—	Reserved.
3–0	BPID	Buffer Pool ID. Statically assigned buffer pool identifier for requesting free storage.

16.4.2.23 Inbound Block *m* Type10 Classification *n* Data Offset Register (IB*m*T10C*n*DOR)

This is the Type10 descriptor data offset register. Inbound transactions of Type10 writes the descriptor information into byte zero of the first allocated data buffer. The offset specified here

indicates where the first byte of the data is located within the buffer. To allow for the descriptor, the minimum offset should be 32-bytes.

Offset: B0C0: 0034; B0C1: 0134;
B0C2: 0234; B0C3: 0334;
B0C4: 0434; B0C5: 0534;
B0C6: 0634; B0C7: 0734;
B1C0–7: 1n34;
B2C0–7: 2n34;
B3C0–7: 3n34;
B4C0–7: 4n34;
B5C0–7: 5n34;
B6C0–7: 6n34;
B7C0–7: 7n34

Access: Read/Write

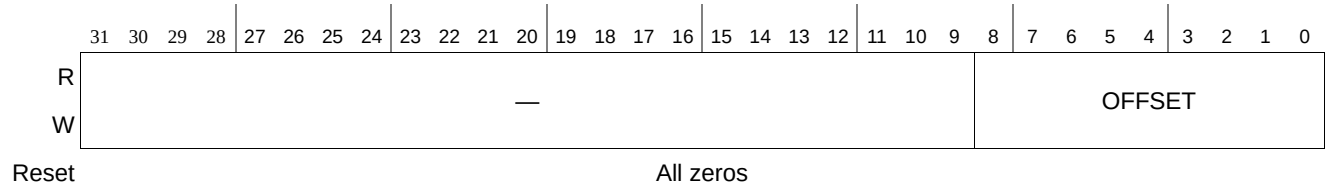


Figure 16-70. Inbound Block *m* Type10 Classification *n* Data Offset Register (IB*m*T10C*n*DOR)

Table 16-145. IB*m*T10C*n*DOR Field Descriptions

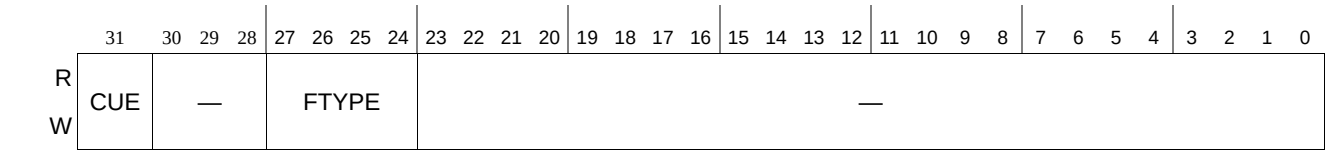
Bits	Name	Description
31–9	—	Reserved
8–0	OFFSET	Byte offset from buffer starting address to where the first byte of data is located.

16.4.2.24 Inbound Block *m* Type11 Classification *n* Mode Registers (IB*m*T11C*n*MR)

The inbound Type11 classification mode register allows software to enable the classification unit and to control various message operation characteristics.

Offset: B0C0: 0000; B0C1: 0100;
B0C2: 0200; B0C3: 0300;
B0C4: 0400; B0C5: 0500;
B0C6: 0600; B0C7: 0700;
B1C0–7: 1*n*00;
B2C0–7: 2*n*00;
B3C0–7: 3*n*00;
B4C0–7: 4*n*00;
B5C0–7: 5*n*00;
B6C0–7: 6*n*00;
B7C0–7: 7*n*00

Access: Read/Write



Reset: All zeros

Figure 16-71. Inbound Block *m* Type11 Classification *n* Mode Registers (IB*m*T11C1*n*MR)

Table 16-146. IB*m*T11C*n*MR Field Descriptions

Bits	Name	Description
31	CUE	Inbound Type11 classification unit enable. 0 Disabled. 1 Inbound Type11 classification unit has been initialized and can service incoming reassembly operations.
30–28	—	Reserved
27–24	FTYPE	Type select. Determines the type of operation for this message unit. 1011 Type11 All other values reserved.
23–0	—	Reserved

16.4.2.25 Inbound Block *m* Type11 Classification *n* Status Register (IB*m*T11C*n*SR)

The status register indicates current status of the classification rule.

Offset: B0C0: 0x1E_0004; B0C1: 0x1E_0104; Access: Read-Only
B0C2: 0x1E_0204; B0C3: 0x1E_0304;
B0C4: 0x1E_0404; B0C5: 0x1E_0504;
B0C6: 0x1E_0604; B0C7: 0x1E_0704;
B1C0-7: 0x1E_1*n*04;
B2C0-7: 0x1E_2*n*04;
B3C0-7:0x1E_3*n*04;
B4C0-7:0x1E_4*n*04;
B5C0-7:0x1E_5*n*04;
B6C0-7:0x1E_6*n*04;
B7C0-7:0x1E_7*n*04

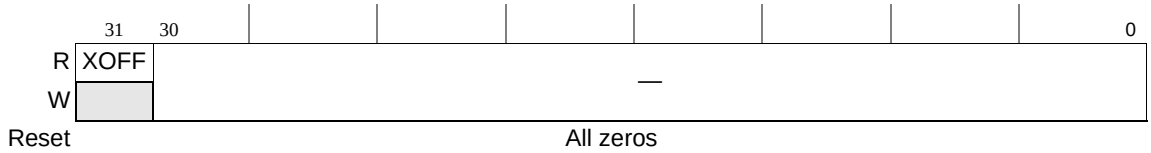


Figure 16-72. Inbound Block *m* Type11 Classification *n* Status Register (IB*m*T11C*n*SR)

Table 16-147. IB*m*T11C*n*SR Field Descriptions

Bits	Name	Description
31	XOFF	Critical flow control. 0 Classification rule not flow controlled. 1 Classification has received a critical flow control by an inbound queue, targeted by the rule, due to a queue near full condition. This bit is automatically cleared when the exit watermark is reached as defined by IB*m*MQ*n*CMR[EXIT_WM], or when the rule is disabled. Further transactions matching this rule will results in physical retry.
30–0	—	Reserved

16.4.2.26 Inbound Block *m* Type11 Classification *n* Message Queue Register (IB*m*T11C*n*MQR)

This register specifies the message queue ID destination for inbound messages.

Offset: B0C0: 0008; B0C1: 0108;
B0C2: 0208; B0C3: 0308;
B0C4: 0408; B0C5: 0508;
B0C6: 0608; B0C7: 0708;
B1C0–7: 1*n*08;
B2C0–7: 2*n*08;
B3C0–7:3*n*08;
B4C0–7:4*n*08;
B5C0–7:5*n*08;
B6C0–7:6*n*08;
B7C0–7:7*n*08

Access: Read/Write

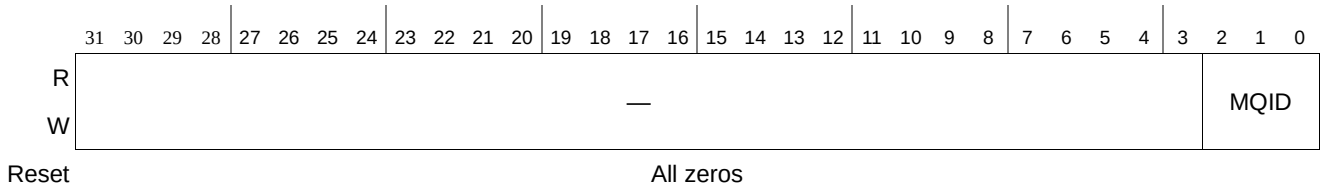


Figure 16-73. Inbound Block *m* Type11 Classification *n* Message Queue Register (IB*m*T11C*n*MQR)

Table 16-148. IB*m*T11C*n*MQR Field Descriptions

Bits	Name	Description
31–3	—	Reserved
2–0	MQID	Message queue ID destination for inbound messages to this unit. For proper operation, this field should only be modified when the inbound classification unit is not enabled.

16.4.2.27 Inbound Block *m* Type11 Classification *n* Rule Value Register 0 (IB*m*T11C*n*RVR0) and Inbound Block *m* Type11 Classification *n* Rule Value Register 1 (IB*m*T11C*n*RVR1)

The inbound Type11 classification rule value registers are used to configure a message rule directing inbound messages to a programmable message queue. The value registers are used in conjunction with the mask registers to create the rule. For example if the mbox field is set to 0x2 and the corresponding mask register field is set to 0x1, the inbound unit would accept any mbox values from 0x2-0x3. If multiple inbound classification units match the same message rule, the lowest numbered enabled inbound classification unit will determine the winner.

Offset: B0C0: 0010; B0C1: 0110;
B0C2: 0210; B0C3: 0310;
B0C4: 0410; B0C5: 0510;
B0C6: 0610; B0C7: 0710;
B1C0–7: 1*n*10;
B2C0–7: 2*n*10;
B3C0–7: 3*n*10;
B4C0–7: 4*n*10;
B5C0–7: 5*n*10;
B6C0–7: 6*n*10;
B7C0–7: 7*n*10

Access: Read/Write

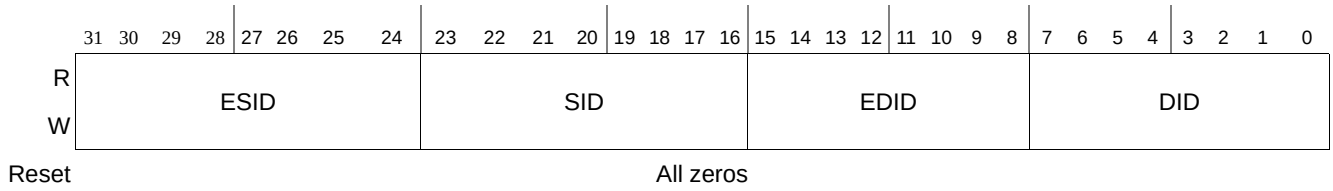


Figure 16-74. Inbound Block *m* Type11 Classification *n* Rule Value Register 0 (IB*m*T11C*n*RVR0)

Table 16-149. IB*m*T11C*n*RVR0 Field Descriptions

Bits	Name	Description
31–24	ESID	Extended source ID - Most significant byte of a 16-bit source ID when operating in large transport mode. Reserved when operating in small transport mode.
23–16	SID	Source ID - Contains the source ID field of the transaction (Device ID of the source).
15–8	EDID	Extended destination ID - Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
7–0	DID	Destination ID - Contains the destination ID field of the transaction (Device ID of the destination).

Offset: B0C0: 0014; B0C1: 0114;
B0C2: 0214; B0C3: 0314;
B0C4: 0414; B0C5: 0514;
B0C6: 0614; B0C7: 0714;
B1C0–7: 1n14;
B2C0–7: 2n14;
B3C0–7: 3n14;
B4C0–7: 4n14;
B5C0–7: 5n14;
B6C0–7: 6n14;
B7C0–7: 7n14

Access: Read/Write

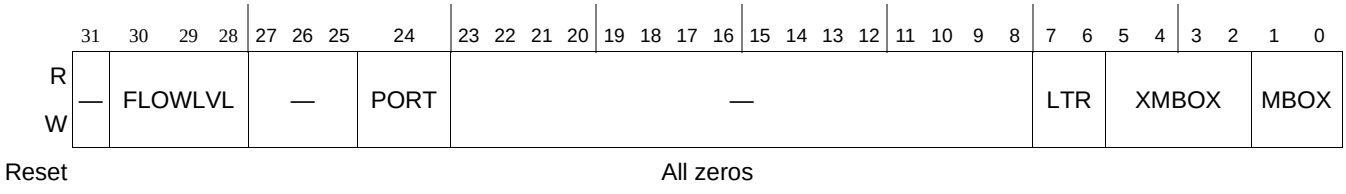


Figure 16-75. Inbound Block *m* Type11 Classification *n* Rule Value Register 1 (IB*m*T11C*n*RVR1)

Table 16-150. IB*m*T11C*n*RVR1 Field Descriptions

Bits	Name	Description
31	—	Reserved
30–28	FLOWLVL	Transaction flow level. 000 Lowest flow level 001 Next lowest flow level ... 100 Next highest flow level 101 Highest flow level 110 Reserved 111 Reserved
27–25	—	Reserved
24	PORT	Port number. 0 Port 1 1 Port 2
23–8	—	Reserved
7–6	LTR	Letter field from the packet header.
5–2	XMBOX	Extended mailbox field from the packet header. Valid only for single segment messages.
1–0	MBOX	Mailbox field from the packet header.

16.4.2.28 Inbound Block *m* Type11 Classification *n* Rule Mask Register 0 (IB*m*T11C*n*RMR0) and Inbound Block *m* Type11 Classification *n* Rule Mask Register 1 (IB*m*T11C*n*RMR1)

The inbound Type11 classification rule mask registers are used to configure a message rule directing inbound messages to a programmable message queue. The mask registers are used in conjunction with the value registers to create the rule.

Each bit set indicates a don’t care value for the inbound message header attribute.

Offset: B0C0: 0018; B0C1: 0118;
B0C2: 0218; B0C3: 0318;
B0C4: 0418; B0C5: 0518;
B0C6: 0618; B0C7: 0718;
B1C0–7: 1n18;
B2C0–7: 2n18;
B3C0–7: 3n18;
B4C0–7: 4n18;
B5C0–7: 5n18;
B6C0–7: 6n18;
B7C0–7: 7n18

Access: Read/Write

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

Figure 16-76. Inbound Block *m* Type11 Classification *n* Rule Mask Register 0 (IB*m*T11C*n*RMR0)

Table 16-151. IB*m*T11C*n*RMR0 Field Descriptions

Bits	Name	Description
31–24	ESID	Extended source ID mask - Most significant byte of a 16-bit source ID when operating in large transport mode. Reserved when operating in small transport mode.
23–16	SID	Source ID mask - Contains the source ID field of the transaction (Device ID of the source).
15–8	EDID	Extended destination ID mask - Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
7–0	DID	Destination ID mask - Contains the destination ID field of the transaction (Device ID of the destination).

Offset : B0C0: 001C; B0C1: 011C;
B0C2: 021C; B0C3: 031C;
B0C4: 041C; B0C5: 051C;
B0C6: 061C; B0C7: 071C;
B1C0–7: 1n1C;
B2C0–7: 2n1C;
B3C0–7: 3n1C;
B4C0–7: 4n1C;
B5C0–7: 5n1C;
B6C0–7: 6n1C;
B7C0–7: 7n1C

Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—		FCM		—			PORT	—																LTR		XMBOX				MBOX	
Reset	All zeros																															

Figure 16-77. Inbound Block *m* Type11 Classification *n* Rule Mask Register 1 (IB*m*T11C*n*RMR1)

Table 16-152. IB*m*T11C*n*RMR1 Field Descriptions

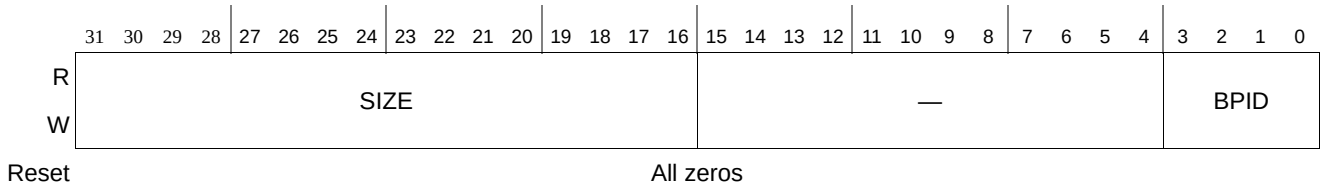
Bits	Name	Description
31–30	—	Reserved
29–28	FCM	Flow conditional match. 00 Exact match on flow level 01 Match on less than or equal to flow level value 10 Match on greater than or equal to flow level value 11 Reserved
27–25	—	Reserved
24	PORT	Port number. 0 Port 1 1 Port 2
23–8	—	Reserved
7–6	LTR	Bit mask for letter field from the packet header.
5–2	XMBOX	Bit mask for xmbox field from the packet header.
1–0	MBOX	Bit mask for mbox field from the packet header.

16.4.2.29 Inbound Block *m* Type11 Classification Unit *n* Data Buffer Pool Register (IB*m*T11C*n*DBPR)

This is the data buffer pool register, for inbound Type11 classification, used for requesting data buffers from the Buffer Manager. During the process of enqueueing inbound Type11 descriptors, buffers are prefetched from the assigned buffer pool for storage of data.

Offset: B0C0: 0030; B0C1: 0130;
B0C2: 0230; B0C3: 0330;
B0C4: 0430; B0C5: 0530;
B0C6: 0630; B0C7: 0730;
B1C0–7: 1n30;
B2C0–7: 2n30;
B3C0–7: 3n30;
B4C0–7: 4n30;
B5C0–7: 5n30;
B6C0–7: 6n30;
B7C0–7: 7n30

Access: Read/Write



Reset: All zeros

Figure 16-78. Inbound Block *m* Type11 Classification Unit *n* Data Buffer Pool Register (IB*m*T11C*n*DBPR)

Table 16-153. IB*m*T11C*n*DBPR Field Descriptions

Bits	Name	Description
31–16	SIZE	Buffer size. Indicates the buffer size for the Buffer Pool used. 0x0000 64K bytes 0x0040 64 bytes 0x0041 65 bytes ... 0x0080 128 bytes 0x0081 129 bytes ... 0x0100 256 bytes ... 0x0000 64K bytes – 1 Minimum buffer size is 64 bytes. Although larger buffer sizes are supported, message data payload is maximum 4K bytes plus descriptor size. The minimum buffer size must account for the last segment being equal to the segment size. For example, if the message has 4 segments and the segment size is 256 bytes, the buffer size must accommodate the maximum data payload of 1 KB, regardless of the last segment size. **Note:** If 0xFFFF is selected, the inbound buffer size is interpreted as 0 bytes and not 64 Kbytes. Therefore, do not select this value. If the buffer size is 64 Kbytes the software can set IBmT11CnhDBPR[SIZE] to anything less than 64 Kbytes, but larger than the required message size to avoid a buffer size error.
15–4	—	Reserved.
3–0	BPID	Buffer Pool ID. Statically assigned buffer pool identifier for requesting free storage.

16.4.2.30 Inbound Block *m* Type11 Classification Unit *n* Data Offset Register (IB*m*T11C*n*DOR)

This is the Type11 descriptor data offset register. Inbound transactions of Type11 writes the descriptor information into byte zero (0) of the first allocated data buffer. The offset specified here indicates where the first byte of the data payload is located within the buffer. To allow for the descriptor, the minimum offset must be 32-bytes.

Offset: B0C0: 0034; B0C1: 0134;
B0C2: 0234; B0C3: 0334;
B0C4: 0434; B0C5: 0534;
B0C6: 0634; B0C7: 0734;
B1C0–7: 1n34;
B2C0–7: 2n34;
B3C0–7: 3n34;
B4C0–7: 4n34;
B5C0–7: 5n34;
B6C0–7: 6n34;
B7C0–7: 7n34

Access: Read/Write

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

Figure 16-79. Inbound Block *m* Type11 Classification *n* Data Offset Register (IB*m*T11C*n*DOR)

Table 16-154. IB*m*T11C*n*DOR Field Descriptions

Bits	Name	Description
31–9	—	Reserved
8–0	OFFSET	Byte offset from buffer starting address to where the first byte of data payload is located.

16.4.2.31 Inbound Block *m* Message Queue *n* Mode Registers (IB*m*MQ*n*MR)

The inbound message queue mode register allows software to control various message queue operation characteristics.

Offset: B0Q0: 0040; B0Q1: 0140;
B0Q2: 0240; B0Q3: 0340;
B0Q4: 0440; B0Q5: 0540;
B0Q6: 0640; B0Q7: 0740;
B1Q0–7: 1*n*40;
B2Q0–7: 2*n*40;
B3Q0–7: 3*n*40;
B4Q0–7: 4*n*40;
B5Q0–7: 5*n*40;
B6Q0–7: 6*n*40;
B7Q0–7: 7*n*40

Access: Read/Write

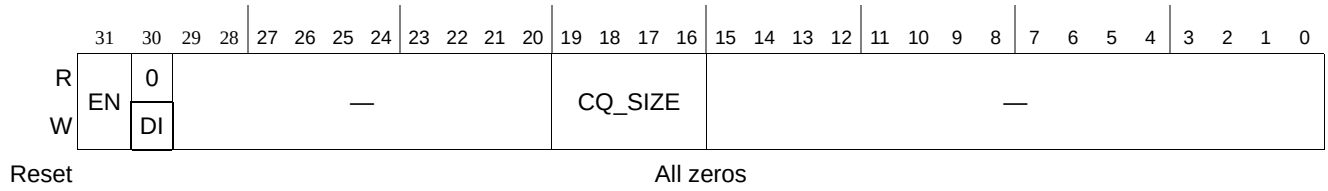


Figure 16-80. Inbound Block *m* Message Queue *n* Mode Registers (IB*m*MQ*n*MR)

Table 16-155. IB*m*MQ*n*MR Field Descriptions

Bits	Name	Description
31	EN	Message queue enable. 0 Disabled. 1 Enabled. Message queue has been initialized and can service incoming command descriptors. For proper operation, this bit should not be cleared while inbound units are steering traffic to this message queue.
30	DI	Dequeue pointer increment. Setting this bit will cause the dequeue pointer IB*m*MQ*n*DPAR to increment to the next entry. Always reads as 0.
29–20	—	Reserved
19–16	CQ_SIZE	Circular descriptor queue size. Determines the number of command descriptors that can be placed on the circular queue without overflow. 0000 64 0001 128 0010 256 0011 512 0100 1024 0101 2048 0110 4096 0111 8192 1000 16384 1001 Reserved ... 1111 Reserved For proper operation, this field should only be modified when the inbound message queue is not enabled.
15–0	—	Reserved

16.4.2.32 Inbound Block *m* Message Queue *n* Status Registers (IB*m*MQ*n*SR)

The inbound message queue status register reports various queue conditions during and after enqueue/dequeue operations.

Offset: B0Q0: 0044; B0Q1: 0144;
B0Q2: 0244; B0Q3: 0344;
B0Q4: 0444; B0Q5: 0544;
B0Q6: 0644; B0Q7: 0744;
B1Q0–7: 1*n*44;
B2Q0–7: 2*n*44;
B3Q0–7: 3*n*44;
B4Q0–7: 4*n*44;
B5Q0–7: 5*n*44;
B6Q0–7: 6*n*44;
B7Q0–7: 7*n*44

Access: Read-Only

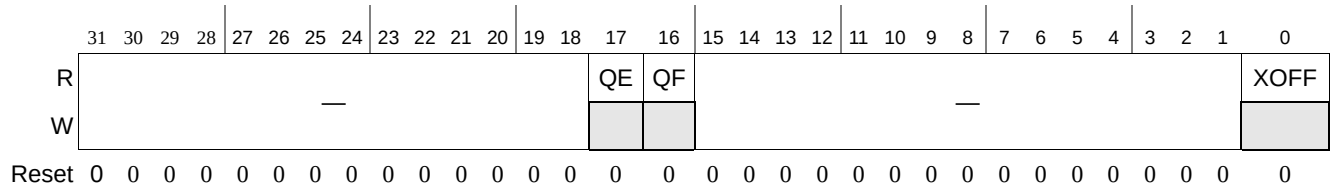


Figure 16-81. Inbound Block *m* Message Queue *n* Status Registers (IB*m*MQ*n*SR)

Table 16-156. IB*m*MQ*n*SR Field Descriptions

Bits	Name	Description
31–18	—	Reserved
17	QE	Queue empty. Read-only. 0 Queue is not empty. 1 Queue is empty.
16	QF	Queue full. Read-only. 0 Queue is not full. 1 Queue is full.
15–1	—	Reserved
0	XOFF	Message queue has entered congestion management (XOFF), and generated an outbound stream management flow control message to all possible sources of traffic. If this bit is set and the exit congestion threshold is reached, an XON outbound stream management flow control message is generated. This bit is automatically cleared if the message queue is disabled. This bit reflects received enqueue operations that may not have been committed to memory.

16.4.2.33 Inbound Block *m* Message Queue *n* Dequeue Pointer Address Registers (IB*m*MQ*n*DPAR)

The inbound message queue dequeue pointer address registers contain the address of the first command descriptor in memory to be processed by software. Software must initialize these registers to point to the first command descriptor in memory. After processing this command descriptor, software either write IB*m*MQ*n*DPAR to point to the next descriptor location in memory or sets IB*m*MQ*n*MR[DI] to cause the message unit to increment IB*m*MQ*n*DPAR to point to the next descriptor. The message queue is not empty if the inbound message queue enqueue pointers and the inbound message queue dequeue pointers are not equal. This is also reflected by the status bit IB*m*MQ*n*SR[QE] being clear. If the two pointers are equal, it could mean the queue is either full or empty and the status register IB*m*MQ*n*SR should be read to avoid

an underflow. Writing the dequeue pointer to a value outside of the queue size boundary will results in undefined behavior.

When software initializes the dequeue pointer, the queue to which it points must be aligned on a boundary equal to the number of queue entries x command descriptor size. For example, if there are 64 entries, the queue must be 1024-byte aligned.

Offset: B0Q0: 004C; B0Q1: 014C;
B0Q2: 024C; B0Q3: 034C;
B0Q4: 044C; B0Q5: 054C;
B0Q6: 064C; B0Q7: 074C;
B1Q0–7: 1*n*4C;
B2Q0–7: 2*n*4C;
B3Q0–7: 3*n*4C;
B4Q0–7: 4*n*4C;
B5Q0–7: 5*n*4C;
B6Q0–7: 6*n*4C;
B7Q0–7: 7*n*4C

Access: Read/Write

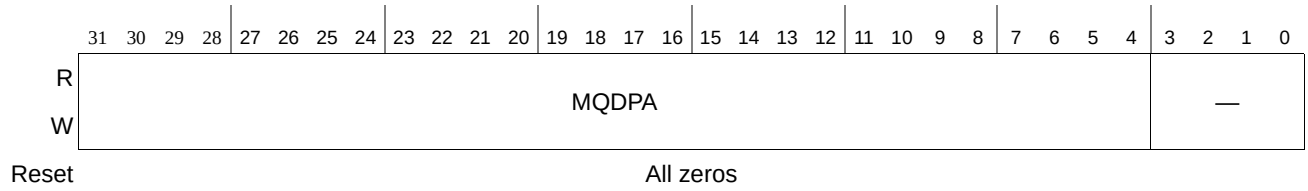


Figure 16-82. Inbound Block *m* Message Queue *n* Dequeue Pointer Address Registers (IB*m*MQ*n*DPAR)

Table 16-157. IB*m*MQ*n*DPAR Field Descriptions

Bits	Name	Description
31–4	MQDPA	Message queue dequeue pointer address. Contains the address of the first command descriptor in memory to process. The command descriptor must be aligned to a 16 byte boundary.
3–0	—	Reserved

16.4.2.34 Inbound Block *m* Message Queue *n* Enqueue Pointer Address Registers (IB*m*MQ*n*EPAR)

The inbound message queue enqueue pointer address registers contain the address of the next command descriptor to be added by the message unit. Software must initialize these registers to point to the first command descriptor in memory. After adding a new command descriptor, the message unit increments the inbound message descriptor queue enqueue pointer address in IB*m*MQ*n*EPAR to point to the next descriptor. The message queue is not empty if the inbound message queue enqueue pointers and the inbound message queue dequeue pointers are not equal. This is also reflected by the status bit IB*m*MQ*n*SR[QE] being clear. If the two pointers are equal, it could mean the queue is either full or empty and the status register IB*m*MQ*n*SR should be read. If the queue becomes full when the message unit writes a new command descriptor, the queue full status condition will occur, IB*m*MQ*n*SR[QF]=1. If the queue is full when the message unit tries to write a new command descriptor, the queue overflow condition will occur. If

IB*m*MQIER[QOIE]=1 and IB*m*MQIDR[QO]=1, a queue overflow interrupt will be generated. If the write of the command descriptor causes an internal transaction error, the transaction error condition will occur and the pointer will not be updated. If MQIER[TEIE]=1 and MQEDR[TE]=1, a transaction error interrupt will be generated. When software initializes the enqueue pointer, the queue to which it points must be aligned on a boundary equal to the number of queue entries x command descriptor size. For example, if there are 64 entries, the queue must be 1024-byte aligned.

Offset: B0Q0: 0054; B0Q1: 0154;
B0Q2: 0254; B0Q3: 0354;
B0Q4: 0454; B0Q5: 0554;
B0Q6: 0654; B0Q7: 0754;
B1Q0–7: 1*n*54;
B2Q0–7: 2*n*54;
B3Q0–7: 3*n*54;
B4Q0–7: 4*n*54;
B5Q0–7: 5*n*54;
B6Q0–7: 6*n*54;
B7Q0–7: 7*n*54

Access: Read/Write

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

Figure 16-83. Inbound Block *m* Message Queue *n* Enqueue Pointer Address Registers (IB*m*MQ*n*EPAR)

Table 16-158. IB*m*MQ*n*EPAR Field Descriptions

Bits	Name	Description
31–4	MQEPA	Message queue enqueue pointer address. Contains the address of the next command descriptor to be added by the message unit.
3–0	—	Reserved

16.4.2.35 Inbound Block *m* Message Queue *n* Congestion Management Register (IB*m*MQ*n*CMR)

The inbound message queue congestion management register allows the setting of watermark levels for generating basic stream management XON/XOFF flow control messages. Stream management flow control is enabled through DSLLCCSR (see SRIO block guide for definition) and is intended for Type9 only. The exit watermark is also used to clear the critical watermark condition, IB*m*T*t*C*n*SR[XOFF], set in the classification unit if an inbound queue reaches critical congestion.

The congestion enter watermark should be greater than the exit watermark for proper operation. The watermark levels should be set to accommodate system latencies for in-flight transactions to avoid overflow of the inbound queue.

Offset: B0Q0: 0058; B0Q1: 0158;
B0Q2: 0258; B0Q3: 0358;
B0Q4: 0458; B0Q5: 0558;
B0Q6: 0658; B0Q7: 0758;
B1Q0–7: 1*n*58;
B2Q0–7: 2*n*58;
B3Q0–7: 3*n*58;
B4Q0–7: 4*n*58;
B5Q0–7: 5*n*58;
B6Q0–7: 6*n*58;
B7Q0–7: 7*n*58

Access: Read/Write

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

Inbound Block *m* Message Queue *n* Congestion Management Register (IB*m*MQ*n*CMR)

Table 16-159. IB*m*MQ*n*CMR Field Descriptions

Bits	Name	Description
31–30	—	Reserved
29–16	ENTER_WM	Congestion enter watermark level. When the number of entries in the circular queue has passed over the enter watermark, a stream management XOFF flow control message is sent to all source devices directing traffic to this queue. For proper operation, this field should be greater than the exit watermark, less than the queue size and only be modified when the inbound message queue is not enabled. A value of zero disables the watermark.
15–14	—	Reserved
13–0	EXIT_WM	Congestion exit watermark level. When the number of entries in the circular queue has passed below the exit watermark, a stream management XON flow control message is sent to all source devices directing traffic to this queue. For proper operation, this field should be less than the exit watermark and only be modified when the inbound message queue is not enabled. A value of zero disables the watermark.

16.4.2.36 Inbound Block *m* Message Queue Interrupt Enable Registers (IB*m*MQIER)

The inbound message queue interrupt enable register determines if an event should cause an interrupt. An interrupt occurs if a bit in this register is set and the corresponding bit in the status or interrupt detect register is also set.

To clear an interrupt, write a 1 to the interrupt detect register. Interrupts caused by normal status conditions are cleared by responding to the condition for which the interrupt was caused.

Offset: B0: 0060;
B1: 1060;
B2: 2060;
B3: 3060;
B4: 4060;
B5: 5060;
B6: 6060;
B7: 7060

Access: Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MQ0TIE	MQ1TIE	MQ2TIE	MQ3TIE	MQ4TIE	MQ5TIE	MQ6TIE	MQ7TIE	MQ0PEIE	MQ1PEIE	MQ2PEIE	MQ3PEIE	MQ4PEIE	MQ5PEIE	MQ6PEIE	MQ7PEIE
Reset	All zeros															

	16	31
R W	—	
Reset	All zeros	

Inbound Block *m* Message Queue Interrupt Enable Register (IB*m*MQIER)

Table 16-160. IB*m*MQIER Field Descriptions

Bits	Name	Description
15	MQ0TIE	Message queue 0 threshold interrupt enable. 0 Disabled 1 Message queue 0 threshold interrupt enabled.
14	MQ1TIE	Same as MQ0TIE.
13	MQ2TIE	Same as MQ0TIE.
12	MQ3TIE	Same as MQ0TIE.
11	MQ4TIE	Same as MQ0TIE.
10	MQ5TIE	Same as MQ0TIE.
9	MQ6TIE	Same as MQ0TIE.
8	MQ7TIE	Same as MQ0TIE.
7	MQ0PEIE	Message queue 0 programming error interrupt enable. 0 Disabled 1 Message queue 0 programming error interrupt enabled.
6	MQ1PEIE	Same as MQ0PEIE.
5	MQ2PEIE	Same as MQ0PEIE.
4	MQ3PEIE	Same as MQ0PEIE.
3	MQ4PEIE	Same as MQ0PEIE.
2	MQ5PEIE	Same as MQ0PEIE.
1	MQ6PEIE	Same as MQ0PEIE.
0	MQ7PEIE	Same as MQ0PEIE.
16-31	—	Reserved

16.4.2.37 Inbound Block *m* Message Queue Interrupt Detect Registers (IB*m*MQIDR)

The inbound message queue interrupt detect register indicates if an error condition was detected. Write 1 to clear the detected error condition and the interrupt, if enabled.

Offset: B0-: 0064;
B1: 1064;
B2: 2064;
B3: 3064;
B4: 4064;
B5: 5064;
B6: 6064;
B7: 7064

Access: Special

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MQ0T	MQ1T	MQ2T	MQ3T	MQ4T	MQ5T	MQ6T	MQ7T	MQ0PE	MQ1PE	MQ2PE	MQ3PE	MQ4PE	MQ5PE	MQ6PE	MQ7PE
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	All zeros															

	16	31
R	—	
W		
Reset	All zeros	

Inbound Block *m* Message Queue Interrupt Detect Register (IB*m*MQIDR)

Table 16-161. IB*m*MQIDR Field Descriptions

Bits	Name	Description
15	MQ0T	Message queue 0 threshold. Write 1 to clear. 0 Message queue 0 threshold(s) not reached. 1 Message queue 0 holds at least the number of messages specified by IB*m*MQ*n*ICR[ICMT] or the threshold timer has expired as specified by IB*m*MQ*n*ICR[ICTT].
14	MQ1T	Same as MQ0T.
13	MQ2T	Same as MQ0T.
12	MQ3T	Same as MQ0T.
11	MQ4T	Same as MQ0T.
10	MQ5T	Same as MQ0T.
9	MQ6T	Same as MQ0T.
8	MQ7T	Same as MQ0T.
7	MQ0PE	Message queue 0 programming error. Message queue 0 was disabled during an enqueue operation. Write 1 to clear.
6	MQ1PE	Same as MQ0PE.
5	MQ2PE	Same as MQ0PE.
4	MQ3PE	Same as MQ0PE.
3	MQ4PE	Same as MQ0PE.

Table 16-161. IB*m*MQIDR Field Descriptions (Continued)

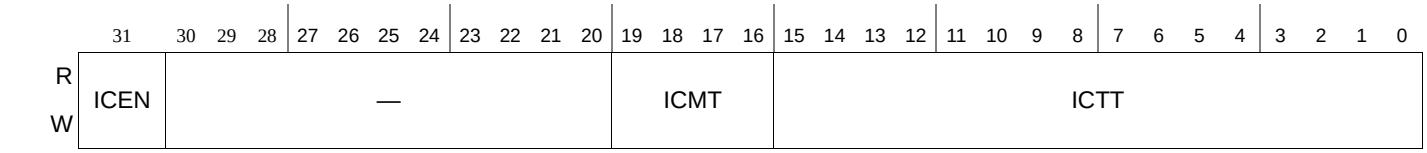
Bits	Name	Description
2	MQ5PE	Same as MQ0PE.
1	MQ6PE	Same as MQ0PE.
0	MQ7PE	Same as MQ0PE.
16-31	—	Reserved

16.4.2.38 Inbound Block *m* Message Queue *n* Interrupt Coalescing Registers (IB*m*MQ*n*ICR)

The inbound message queue interrupt coalescing register enables and configures the parameters for interrupt coalescing associated with received messages.

Offset: B0Q0: 0068; B0Q1: 0168;
B0Q2: 0268; B0Q3: 0368;
B0Q4: 0468; B0Q5: 0568;
B0Q6: 0668; B0Q7: 0768;
B1Q0–7: 1*n*68;
B2Q0–7: 2*n*68;
B3Q0–7: 3*n*68;
B4Q0–7: 4*n*68;
B5Q0–7: 5*n*68;
B6Q0–7: 6*n*68;
B7Q0–7: 7*n*68

Access: Read/Write



Inbound Block *m* Message Queue *n* Interrupt Coalescing Register (IB*m*MQ*n*ICR)

Table 16-162. IB*m*MQ*n*ICR Field Descriptions

Bits	Name	Description
31	ICEN	Interrupt coalescing enable. 0 Interrupt coalescing is disabled. 1 Interrupt coalescing is enabled. If the message queue threshold interrupt is enabled (IB*m*MQ*n*IER[MQTIE] is set), an interrupt is raised when the threshold number of messages is reached (defined by IB*m*MQ*n*ICR[ICMT]) or when the threshold timer expires (determined by IB*m*MQ*n*ICR[ICTT]).
30–20	—	Reserved

Table 16-162. IB*m*MQ*n*ICR Field Descriptions (Continued)

Bits	Name	Description
19–16	ICMT	Interrupt coalescing message threshold. While interrupt coalescing is enabled (IBmMQICR[ICEN] is set), this values determines the minimum number of transactions present in the inbound queue before raising an interrupt. Software should set the dequeue pointer after processing the event such that the number of transactions in the queue is less than the threshold to avoid further interrupts. 0000 0 (disabled) 0001 1 message 0010 2 messages 0011 4 messages 0100 8 messages 0101 16 messages 0110 32 messages 0111 64 messages 1000 128 messages 1001 256 messages 1010 512 messages 1011 1024 messages 1100 2048 messages 1101 Reserved ... 1111 Reserved For proper operation, this field should only be modified when the inbound message queue is not enabled.
15–0	ICTT	Interrupt coalescing timer threshold. While interrupt coalescing is enabled (IB*m*MQ*n*ICR[ICEN] is set), this values determines the maximum amount of time after receiving a message before raising an interrupt. If messages have been received but the message threshold has not been met, an interrupt is raised when the threshold timer reaches zero. The threshold timer is reset to this value upon receiving a new message or the queue becomes is empty as a result of software writing the dequeue pointer.

16.4.2.39 Outbound Block *m* Message Queue *n* Mode Registers (OB*m*MQ*n*MR)

The outbound message queue mode register allows software to control various message queue operation characteristics.

Offset: B0Q0: 00C0; B0Q1: 01C0;
B0Q2: 02C0; B0Q3: 03C0;
B0Q4: 04C0; B0Q5: 05C0;
B0Q6: 06C0; B0Q7: 07C0;
B1Q0–7: 1nC0;
B2Q0–7: 2nC0;
B3Q0–7: 3nC0;
B4Q0–7: 4nC0;
B5Q0–7: 5nC0;

Access: Read//Write

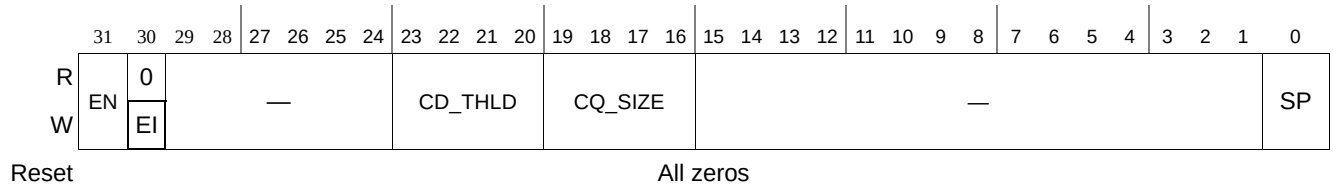


Figure 16-84. Outbound Block *m* Message Queue *n* Mode Registers (OB*m*MQ*n*MR)

Table 16-163. OB*m*MQ*n*MR Field Descriptions

Bits	Name	Description
31	EN	Message queue enable. 0 Disabled. 1 Enabled. Message queue must be initialized. When the queue is not empty, OB*m*MQ*n*SR[QE]=0, not flow controlled, OB*m*MQ*n*SR[XOFF]=0, and no errors are pending, OB*m*MQ*n*(M)EDR, transactions will be processed from the queue.
30	EI	Enqueue pointer increment. Setting this bit will cause the enqueue pointer OB*m*MQ*n*EPAR to increment to the next entry. Always reads as 0.
29–24	—	Reserved
23–20	CD_THLD	Interrupt threshold. Determines the threshold for the number of unprocessed command descriptors remaining before Message Queue Threshold interrupt is signaled. Undefined operation results if the message queue threshold is set greater than or equal to the message queue size (OB*m*MQ*n*MR[CQ_SIZE]). 0000 16 0001 32 0010 64 0011 128 0100 256 0101 512 0110 1024 0111 2048 1000 4096 1001 8192 1010 Reserved ... 1111 Reserved For proper operation, this field should only be modified when the outbound message queue is not enabled.
19–16	CQ_SIZE	Circular descriptor queue size. Determines the number of command descriptors that can be placed on the circular queue without overflow. 0000 64 0001 128 0010 256 0011 512 0100 1024 0101 2048 0110 4096 0111 8192 1000 16384 1001 reserved ... 1111 reserved For proper operation, this field should only be modified when the outbound message queue is not enabled
15–1	—	Reserved
0	SP	Sub-portal. 0 Sub-portal 0 1 Sub-portal 1 The message unit performs Round-Robin dequeue scheduling between sub-portals in an attempt to improve bandwidth performance. The intent is for a sub-portal to handle traffic destined for a specific RapidIO physical port, thus arbitrating between the sub-portals will improve saturation of the link(s). When assigning sub-portals, all queues of arbitration group N within a block must belong to the same sub-portal. For proper operation, this field should only be modified when the outbound message queue is not enabled.

16.4.2.40 Outbound Block *m* Message Queue *n* Status Registers (OB*m*MQ*n*SR)

The outbound message queue status register reports various queue conditions during and after enqueue/dequeue operations. Valid only when queue is enabled.

Offset: B1Q1:00C4; B1Q2:01C4;
B1Q3:02C4; B1Q4:03C4;
B1Q5:04C4; B1Q6:05C4;
B1Q7:06C4; B1Q8:07C4;
B1Q0–7:1mC4;
B2Q0–7:2mC4;
B3Q0–7:3mC4;
B4Q0–7:4mC4;
B5Q0–7:5mC4

Access: Read-Only

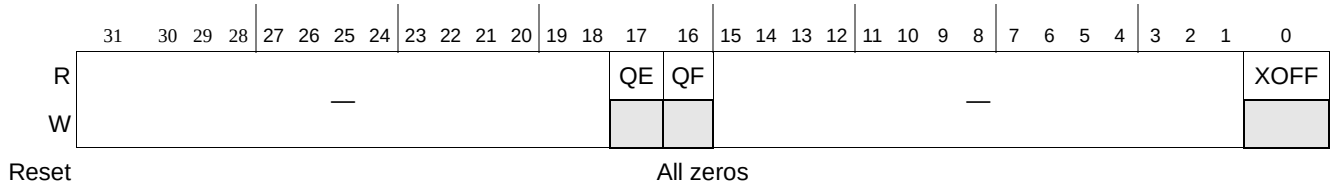


Figure 16-85. Outbound Block *m* Message Queue *n* Status Registers (OB*m*MQ*n*SR)

Table 16-164. OB*m*MQ*n*SR Field Descriptions

Bits	Name	Description
31–18	—	Reserved
17	QE	Queue empty. Read-only. 0 Queue is not empty. 1 Queue is empty. This bit is automatically cleared if the message queue is disabled.
16	QF	Queue full. Read-only. 0 Queue is not full. 1 Queue is full. This bit is automatically cleared if the message queue is disabled.
15–1	—	Reserved
0	XOFF	Message queue has entered congestion management (XOFF), temporarily halting further processing of transactions. There are two reasons for entering congestion: • Inbound XOFF flow control received from target device(s). • Outbound completion queue reached critical level. To exit congestion state, the outbound completion queue must be below critical level and all XOFF target devices must be XON. Up to 32 target devices are tracked for XOFF/XON state. This bit is automatically cleared if the message queue is disabled.

16.4.2.41 Outbound Block *m* Message Queue *n* Dequeue Pointer Address Registers (OB*m*MQ*n*DPAR)

The outbound message queue dequeue pointer address registers contain the address of the first command descriptor in memory to be processed. Software must initialize these registers to point to the first command descriptor in memory. After processing this descriptor, the message queue increments the outbound message queue dequeue pointer address in OB*m*MQ*n*DPAR to point to

the next descriptor. If the outbound message queue enqueue pointer and the outbound message queue dequeue pointer are not equal (indicating that the queue is not empty), the message queue reads the next descriptor from memory for processing. If the enqueue and dequeue pointers are equal after the dequeue pointer has been incremented by the message queue, the queue is empty and further processing halts until the enqueue pointer is incremented by the processor. Incrementing the pointer indicates that a new descriptor has been added to the queue and is ready for transmission. If the queue becomes empty, OB*m*MQ*n*SR[QE]=1 is set. If the read of the command descriptor causes an internal transaction error, the transaction error condition will occur and the pointer will not be updated. If MQIER[TEIE]=1 and MQEDR[TE]=1, a transaction error interrupt will be generated. When software initializes the dequeue pointer, the queue to which it points must be aligned on a boundary equal to the number of queue entries x command descriptor size. For example, if there are 64 entries, the queue must be 1024-byte aligned.

Offset: B1Q1:00CC; B1Q2:01CC; B1Q3:02CC; B1Q4:03CC; B1Q5:04CC; B1Q6:05CC; B1Q7:06CC; B1Q8:07CC; B1Q0–7:1nCC; B2Q0–7:2nCC; B3Q0–7:3nCC; B4Q0–7:4nCC; B5Q0–7:5nCC

Access: Read/Write

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

Figure 16-86. Outbound Block *m* Message Queue *n* Dequeue Pointer Address Registers (OB*m*MQ*n*DPAR)

Table 16-165. OB*m*MQ*n*DPAR Field Descriptions

Bits	Name	Description
31–4	MQDPA	Message queue dequeue pointer address. Contains the address of the first command descriptor in memory to process. The descriptor must be aligned to a 16 byte boundary. For proper operation, this field should only be modified when the outbound message queue is not enabled.
3–0	—	Reserved

16.4.2.42 Outbound Block *m* Message Queue *n* Enqueue Pointer Address Registers (OB*m*MQ*n*EPAR)

The outbound message queue enqueue pointer address registers contain the address for the next command descriptor in memory to be added to the queue. Software must initialize these registers to match the outbound message queue dequeue pointer address. When a message is ready to be sent, the processor writes a command descriptor to the next location in the queue (indicated by the address in OB*m*MQ*n*EPAR), and then either writes OB*m*MQ*n*EPAR to point to the next

descriptor location in memory or sets OB*m*MQ*n*MR[EI]. The queue is full when the enqueue pointer and the dequeue pointer are equal. Writing the enqueue pointer to a value outside of the queue size boundary will results in undefined behavior. When software initializes the enqueue pointer, the queue to which it points must be aligned on a boundary equal to the number of queue entries x command descriptor size. For example, if there are 64 entries, the queue must be 1024-byte aligned.

Offset: B0Q0:00D4; B0Q1:01D4;
B0Q2:02D4; B0Q3:03D4;
B0Q4:04D4; B0Q5:05D4;
B0Q6:06D4; B0Q7:07D4;
B1Q0–7:1nD4;
B2Q0–7:2nD4;
B3Q0–7:3nD4;
B4Q0–7:4nD4;
B5Q0–7:5nD4

Access: Read/Write

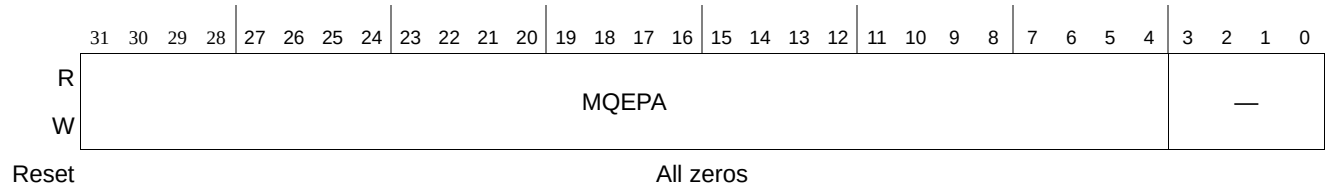


Figure 16-87. Outbound Block *m* Message Queue *n* Enqueue Pointer Registers (OB*m*MQ*n*EPAR)

Figure 16-88.

Table 16-166. OB*m*MQ*n*EPAR Field Descriptions

Bits	Name	Description
31–4	MQEPA	Message queue enqueue pointer address. Contains the address of the last command descriptor in memory to process. The descriptor must be aligned to a 16 byte boundary and a message queue boundary.
3–0	—	Reserved

16.4.2.43 Outbound Block *m* Message Queue Interrupt Enable Registers (OB*m*MQIER)

The outbound message queue interrupt enable register determines if an event should cause an interrupt. An interrupt occurs if a bit in this register is set and the corresponding bit in the interrupt detect register is also set. To clear an interrupt, write a 1 to the interrupt detect register.

Interrupts caused by normal status conditions are cleared by responding to the condition for which the interrupt was caused.

Offset: B0: 00E0; Access: Read/Write
B1: 10E0;
B2: 20E0;
B3: 30E0;
B4: 40E0;
B5: 50E0;
B6: 60E0;
B7: 70E0;

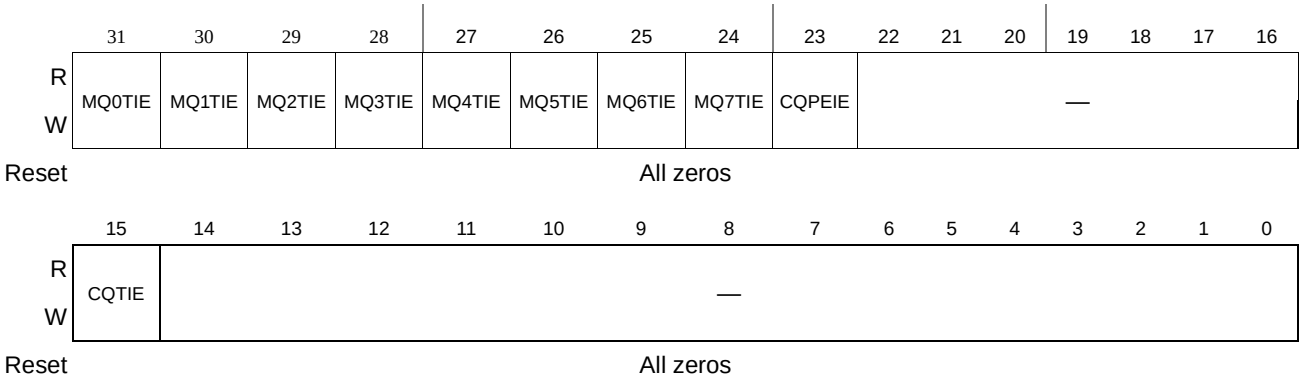


Figure 16-89. Outbound Block *m* Message Queue Interrupt Enable Register (OB*m*MQIER)

Table 16-167. OB*m*MQIER Field Descriptions

Bits	Name	Description
31	MQ0TIE	Message queue 0 threshold interrupt enable. 0 Disabled 1 Message queue 0 threshold interrupt enabled.
30	MQ1TIE	Same as MQ0TIE.
29	MQ2TIE	Same as MQ0TIE.
28	MQ3TIE	Same as MQ0TIE.
27	MQ4TIE	Same as MQ0TIE.
26	MQ5TIE	Same as MQ0TIE.
25	MQ6TIE	Same as MQ0TIE.
24	MQ7TIE	Same as MQ0TIE.
23	CQPEIE	Completion queue programming error interrupt enable. 0 Disabled 1 Completion queue programming error interrupt enabled.
22–16	—	Reserved
15	CQTIE	Completion queue threshold interrupt enable. 0 Disabled 1 Completion queue threshold interrupt enabled.
14–0	—	Reserved

16.4.2.44 Outbound Block *m* Message Queue Interrupt Detect Registers (OB*m*MQIDR)

The outbound message queue interrupt detect register indicates if an event was detected. Write 1 to clear the detected event and the interrupt, if enabled.

Offset: B0: 00E4;
B1: 10E4;
B2: 20E4;
B3: 30E4;
B4: 40E4;
B5: 50E4;
B6: 60E4;
B7: 70E4;

Access: Special

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MQ0T	MQ1T	MQ2T	MQ3T	MQ4T	MQ5T	MQ6T	MQ7T	CQPE	—						
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c							
Reset	All zeros															

	15	14	133	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CQT	—														
W	w1c															
Reset	All zeros															

Figure 16-90. Outbound Block *m* Message Queue Interrupt Detect Register (OB*m*MQIDR)

Table 16-168. OB*m*MQIDR Field Descriptions

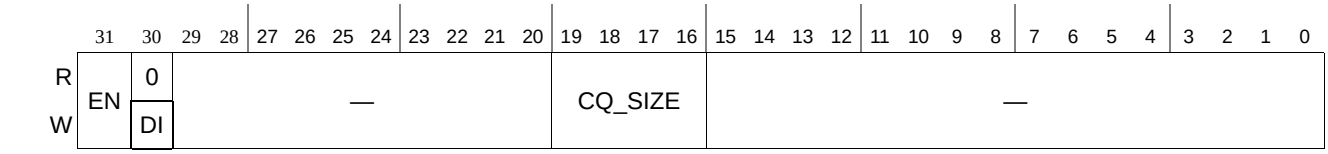
Bits	Name	Description
31	MQ0T	Message queue 0 threshold. Write 1 to clear. 0 Message queue 0 threshold has not been crossed. 1 Message queue 0 threshold has been crossed and less than OB*m*MQ0MR[CD_THLD] entries remains in the queue.
30	MQ1T	Same as MQ0T.
29	MQ2T	Same as MQ0T.
28	MQ3T	Same as MQ0T.
27	MQ4T	Same as MQ0T.
26	MQ5T	Same as MQ0T.
25	MQ6T	Same as MQ0T.
24	MQ7T	Same as MQ0T.
23	CQPE	Completion queue programming error. Completion queue was disabled during an enqueue operation. Write 1 to clear.
22–16	—	Reserved
15	CQT	Completion queue threshold. Write 1 to clear. 0 Completion queue threshold not reached. 1 Completion queue holds at least the number of messages specified by OBmCQICR[ICMT] or the threshold timer has expired as specified by OBmCQICR[ICTT].
14–0	—	Reserved

16.4.2.45 Outbound Block *m* Completion Queue Mode Registers (OB*m*CQMR)

The outbound completion queue mode register allows software to control various completion queue operation characteristics.

Offset: B0: 0800
B1: 1800;
B2: 2800;
B3: 3800;
B4: 4800;
B5: 5800;
B6: 6800;
B7: 7800;

Access: Read/Write



Reset: All zeros

Figure 16-91. Outbound Block *m* Completion Queue Mode Registers (OB*m*CQMR)

Table 16-169. OB*m*CQMR Field Descriptions

Bits	Name	Description
31	EN	Message queue enable. 0 Disabled. 1 Enabled. Completion queue has been initialized and can service outbound completion command descriptors. For proper operation, this bit should not be cleared while outbound units are steering traffic to this completion queue.
30	DI	Dequeue pointer increment. Setting this bit will cause the dequeue pointer OB*m*CQDPAR to increment to the next entry. Always reads as 0.
29–20	—	Reserved
19–16	CQ_SIZE	Circular descriptor queue size. Determines the number of command descriptors that can be placed on the circular queue without overflow. 0000 64 0001 128 0010 256 0011 512 0100 1024 0101 2048 0110 4096 0111 8192 1000 16384 1001 Reserved ... 1111 Reserved For proper operation, this field should only be modified when the outbound completion queue is not enabled.
15–0	—	Reserved

16.4.2.46 Outbound Block *m* Completion Queue Status Registers (OB*m*CQSR)

The outbound completion queue status register reports various queue conditions during and after enqueue/dequeue operations. Valid only when queue is enabled.

Offset: B0: 0804
B1: 1804;
B2: 2804;
B3: 3804;
B4: 4804;
B5: 5804;
B6: 6804;
B7: 7804

Access: Read-Only

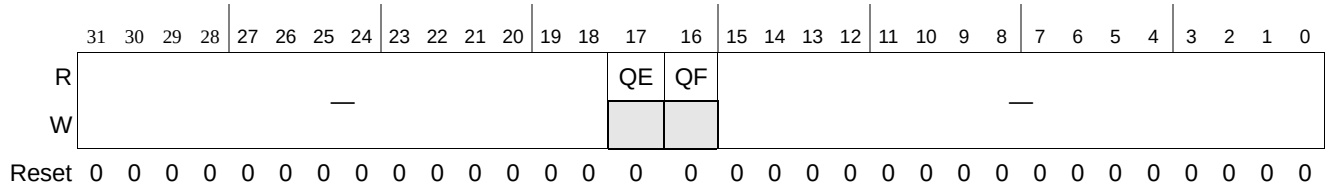


Figure 16-92. Outbound Block *m* Completion Queue Status Registers (OB*m*CQSR)

Table 16-170. OB*m*CQSR Field Descriptions

Bits	Name	Description
31–18	—	Reserved
17	QE	Queue empty. Read-only. 0 Queue is not empty. 1 Queue is empty.
16	QF	Queue full. Read-only. 0 Queue is not full. 1 Queue is full.
15–-0	—	Reserved

16.4.2.47 Outbound Block *m* Completion Queue Dequeue Pointer Address Registers (OB*m*MQ*n*DPAR)

The outbound completion queue dequeue pointer address registers contain the address of the first command descriptor in memory to be processed by software. Software must initialize these registers to point to the first command descriptor in memory. After processing this command descriptor, software either write OB*m*CQDPAR to point to the next descriptor location in memory or sets OB*m*CQMR[DI] to cause the message unit to increment OB*m*CQDPAR to point to the next descriptor. The completion queue is not empty if the outbound completion queue enqueue pointers and the outbound completion queue dequeue pointers are not equal. This is also reflected by the status bit OB*m*CQSR[QE] being clear. If the two pointers are equal, it could mean the queue is either full or empty and the status register OB*m*CQSR should be read to avoid an underflow. Writing the dequeue pointer to a value outside of the queue size boundary will results in undefined behavior. When software initializes the dequeue pointer, the queue to which

it points must be aligned on a boundary equal to the number of queue entries x command descriptor size. For example, if there are 64 entries, the queue must be 1024-byte aligned.

Offset: B0: 080C; B1: 180C; B2: 280C; B3: 380C; B4: 480C; B5: 580C; B6: 680C; B7: 780C — Access: Read/Write

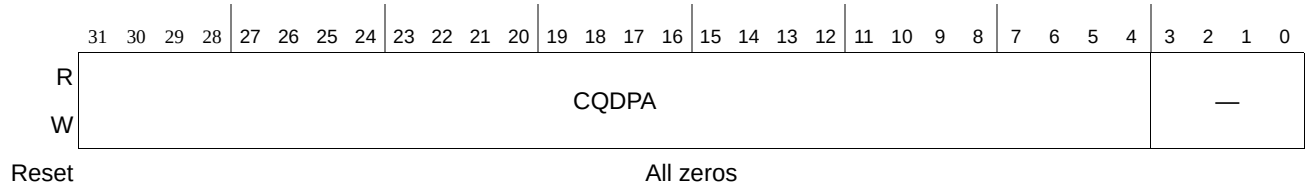


Figure 16-93. Outbound Block *m* Completion Queue Dequeue Pointer Address Registers (OB*m*CQDPAR)

Table 16-171. OB*m*CQDPAR Field Descriptions

Bits	Name	Description
31–4	CQDPA	Completion queue dequeue pointer address. Contains the address of the first command descriptor in memory to process. The command descriptor must be aligned to a 16 byte boundary.
3–0	—	Reserved

16.4.2.48 Outbound Block *m* Completion Queue Enqueue Pointer Address Registers (OB*m*CQEPAR)

The outbound completion queue enqueue pointer address registers contain the address of the next command descriptor to be added by the message unit. Software must initialize these registers to point to the first command descriptor in memory. After adding a new command descriptor, the message unit increments the outbound completion descriptor queue enqueue pointer address in OB*m*CQEPAR to point to the next descriptor. The completion queue is not empty if the outbound completion queue enqueue pointers and the outbound completion queue dequeue pointers are not equal. This is also reflected by the status bit OB*m*CQSR[QE] being clear. If the two pointers are equal, it could mean the queue is either full or empty and the status register OB*m*CQSR should be read. If the queue becomes full when the message unit writes a new command descriptor, the queue full status condition will occur, OB*m*CQSR[QF]=1. An internal critical level watermark will stop the completion queue from reaching overflow condition. If this watermark has been reached, outbound dequeue is automatically halted for all queues associated with the completion queue. Software should enable the completion queue threshold detect in advance to avoid reaching this condition. If the write of the command descriptor causes an internal transaction error, the transaction error condition will occur. If MQIER[TEIE]=1 and MQEDR[TE]=1, a transaction error interrupt will be generated. When software initializes the enqueue pointer, the queue to which it points must be aligned on a boundary equal to the number of queue entries x

command descriptor size. For example, if there are 64 entries, the queue must be 1024-byte aligned.

Offset: B0: 0814;
B1: 1814;
B2: 2814;
B3: 3814;
B4: 4814;
B5: 5814;
B6: 6814;
B7: 7814;

Access: Read/Write

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

Figure 16-94. Outbound Block *m* Completion Queue Enqueue Pointer Address Registers (OB*m*CQEPAR)

Table 16-172. OB*m*CQEPAR Field Descriptions

Bits	Name	Description
31–4	CQEPA	Completion queue enqueue pointer address. Contains the address of the next command descriptor to be added by the message unit.
3–0	—	Reserved

16.4.2.49 Outbound Block *m* Completion Queue Interrupt Coalescing Registers (OB*m*CQICR)

The outbound completion queue interrupt coalescing register enables and configures the parameters for interrupt coalescing associated with received messages.

Offset: B0: 0828;
B1: 1828;
B2: 2828;
B3: 3828;
B4: 4828;
B5: 5828;
B6: 6828;
B7: 7828

Access: Read/Write

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

Figure 16-95. Outbound Block *m* Completion Queue Interrupt Coalescing Register (OB*m*CQICR)

Table 16-173. OB*m*CQICR Field Descriptions

Bits	Name	Description
31	ICEN	Interrupt coalescing enable. 0 Interrupt coalescing is disabled. 1 Interrupt coalescing is enabled. If the completion queue threshold interrupt is enabled (OB*m*MQIER[CQTIE] is set), an interrupt is raised when the threshold number of completions is reached (defined by OB*m*CQICR[ICMT]) or when the threshold timer expires (determined by OB*m*CQICR[ICTT]).
30–20	—	Reserved
19–16	ICCT	Interrupt coalescing completion threshold. While interrupt coalescing is enabled (OB*m*CQICR[ICEN] is set), this values determines the minimum number of transactions present in the completion queue before raising an interrupt. Software should set the dequeue pointer after processing the event such that the number of transactions in the queue is less than the threshold to avoid further interrupts. 0000 0 (disabled) 0001 1 completion 0010 2 completions 0011 4 completions 0100 8 completions 0101 16 completions 0110 32 completions 0111 64 completions 1000 128 completions 1001 256 completions 1010 512 completions 1011 1024 completions 1100 2048 completions 1101 Reserved ... 1111 Reserved For proper operation, this field should only be modified when the outbound completion queue is not enabled.
15–0	ICTT	Interrupt coalescing timer threshold. While interrupt coalescing is enabled (OB*m*CQICR[ICEN] is set), this values determines the maximum amount of time after receiving a completion before raising an interrupt. If completions have been received but the completion threshold has not been met, an interrupt is raised when the threshold timer reaches zero. The threshold timer is reset to this value upon receiving a completion.

16.4.2.50 Software Portal Interrupt Status Register (SWP*n*_ISR)

BMLite contains a line per processor interrupt line for signalling functional conditions to software. This ISR register identifies the source of the interrupt within BMLite. Any asserted bit in this register can assert its interrupt line if enabled to do so in the SWP*n*_IER register. Once a bit in this register is asserted, it will remain asserted until cleared by writing a 1 into its bit location (write-1-to-clear).

Offset 0x0900 (SWP<n>_ISR) Access: w1c
offset 0x1000
range n=0–7

	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	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	19	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	BDSCN							
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-96. Interrupt Status Register (SWP*n*_ISR)

Table 16-174. Register SWP<*n*>_ISR (n=0–7) Bits Description

Field	Description
31–8	Reserved, read as 0.
7–0 BDSCN	Buffer Pool Depletion State change notification interrupt. Affects only ipi_swp_int interrupt. Mapped to SWPn_ASSIGN ring buffers. If the ring buffer assignments are changed in SWPn_ASSIGN the corresponding bits in the ISR register should be serviced and cleared to avoid stale pending interrupts before putting the new ring buffer assignment into service. Ring A: bit 7–Ring H: bit 0

16.4.2.51 Software Portal Interrupt Enable Register (SWP*n*_IER)

The SWP*n*_IER register provides the ability to individually enable each interrupt source to assert the error interrupt signal. If a bit location is asserted in both SWP*n*_ISR and SWP*n*_IER, the interrupt will be asserted. Clearing of an interrupt is achieved by first clearing the interrupt source if applicable, then clearing the associated bit in SWP*n*_ISR.

Offset 0x0904 (SWP<n>_IER) Access: R/W
offset 0x1000
range n=0–7

	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	0	0	0	0	0
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	BDSCN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-97. Interrupt Enable Register (SWP*n*_IER)

Table 16-175. Register SWP<*n*>_IER (n=0–7) Bits Description

Field	Description
31–8	Reserved, read as 0.
7–0 BDSCN	Buffer Pool Depletion State change notification interrupt enable. All fields in this register carry the same assignment as in the ISR register except INH.

16.4.2.52 Software Portal Interrupt Status Disable Register (SWP*n*_ISDR)

The SWP*n*_ISDR register controls reporting of interrupts in the SWP*n*_ISR register. A bit in SWP*n*_ISR will never be asserted if the corresponding bit in the SWP*n*_ISDR register is set (that is, interrupt status is disabled). A set ISR bit will be cleared if the corresponding bit in the ISDR register is set.

Offset 0x0908 (SWP<n>_ISDR)
offset 0x1000
range n=0–7

Access: R/W

	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	0	0	0	0	0
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	BDSCN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-98. Interrupt Status Disable Register (SWP*n*_ISDR)

Table 16-176. Register SWP*n*_ISDR (n=0–7) Bits Description

Field	Description
31–8	Reserved, read as 0.
7–0 BDSCN	Buffer Pool Depletion State change notification interrupt disable. All fields in this register carry the same assignment as in the ISR register except INH.

16.4.2.53 Software Portal Interrupt Inhibit Register (SWP*n*_IIR)

This register allows the interrupt signal to be inhibited without modifying the enable or the status disable registers.

Offset 0x090C (SWP<n>_IIR) Access: R/W
offset 0x1000
range n=0–7

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

Figure 16-99. Software Portal Interrupt Inhibit Register (SWP*n*_IIR)

Table 16-177. Register SWP<*n*>_IIR (n=0–7) Bits Description

Field	Description
31–1	Reserved, read as 0.
0 I	Interrupt Inhibit. 0: Interrupt not inhibited. The interrupt signal may assert. 1:Interrupt inhibited. The interrupt signal will not assert.

16.4.2.54 Software Portal Interrupt Force Register (SWP*n*_IFR)

This is a debug/verification assist register. Writing a 1 to any valid bit in this register will set the corresponding bit in SWP*n*_ISR, unless the corresponding bit in SWP*n*_ISDR is also set. This is a write only register, reading this register will return 0.

Offset 0x0910 (SWP<n>_IFR) Access: W
offset 0x1000
range n=0–7

	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	0	0	0	0	0
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									BDSCN							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-100. Interrupt Force Register (SWP*n*_IFR)

Table 16-178. Register SWP<*n*>_IFR (n=0–7) Bits Description

Field	Description
31–8	Reserved, read as 0.
7–0 BDSCN	Buffer Pool Depletion State change notification interrupt force. All fields in this register carry the same assignment as in the ISR register except INH.

16.4.2.55 Software Portal Configuration/Assign, Enable, and Base Address Registers

Software portal *n*'s ring buffers are configured via these registers. Acquire and release ring buffers are configured in pairs. SWP*n*_CONFIG and SWP*n*_ASSIGN allows configuration of the memory footprint of the acquire and release ring buffers by setting the number of ring buffers and their size. The location of the ring buffers is determined by SWP*n*_BAR1 and SWP*n*_BAR2 where SWP*n*_BAR1 determines the location of ring buffer pair A and SWP*n*_BAR2 determines the location of a ring buffer pair set by SWP*n*_CONFIG. SWPn_EN provides per-ring pair enable bits that allow software to selectively enable/disable individual ring pairs within the software portal.

A maximum sized configuration would result in 8 ring buffer pairs with room for 128 buffer pointers per ring buffer. This would consume (8 ring buffer pairs * 2 per-pair (acq&rel) * 128 entries * 4B/ptr + 64B(hardware PI/CI) = 8256Bytes). If a smaller footprint is required, the number of enabled ring buffer pairs or the size of the ring buffers can be reduced. The block of memory for acquire and release buffers can be split between BAR1 and BAR2 to allow some ring buffers to occupy a different memory region.

The amount of contiguous memory in bytes required for a software portal can be calculated as

```
BAR1_RING_MAX = (8-BAR2_RINGS) or less if the corresponding rings are not enabled
BAR2_RING_MAX = (BAR2_RINGS) or less if the corresponding rings are not enabled
BAR1_SPACE = (BAR1_RING_MAX * 2) * (2^RB_SIZE) * 4B
BAR2_SPACE = (BAR2_RING_MAX * 2) * (2^RB_SIZE) * 4B
PICI_SPACE = 64B if SWPn_CONFIG[PD] is asserted, 0B otherwise
```

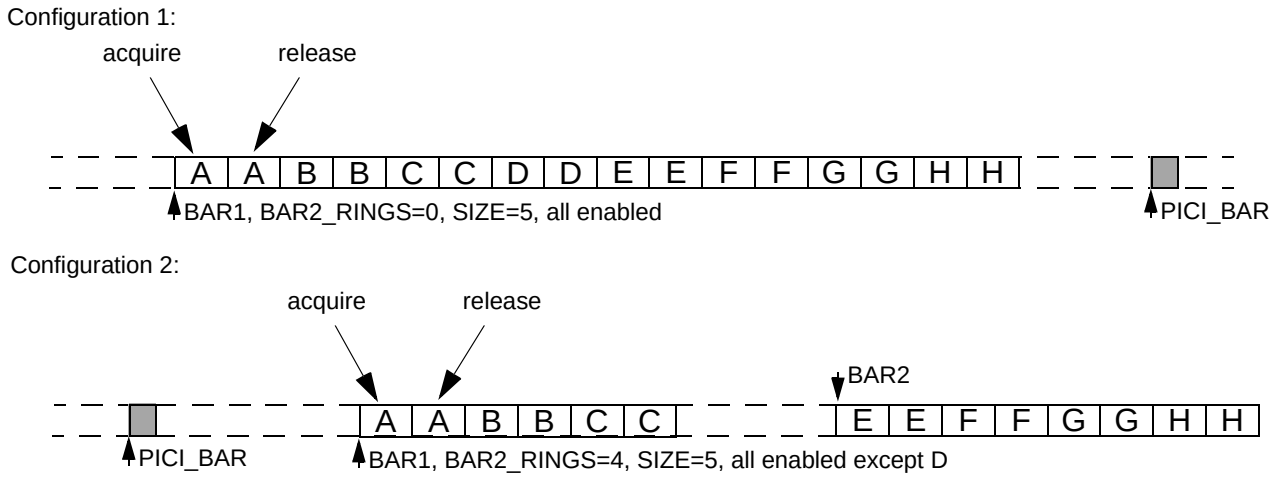


Figure 16-101. Example Ring Buffer Memory Footprints

The PI/CI region (**Figure 16-101**) is a 64 Byte region located at SWP*n*_PICI_BAR. Disabled ring buffers (SWP*n*_EN) do not contain valid values. These values are written intermittently by the hardware when the ring buffers are enabled. The hardware never reads these locations.

0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
ACQ_PI_A	ACQ_CI_A	REL_PI_A	REL_CI_A	ACQ_PI_B	ACQ_CI_B	RELPI_B	REL_CI_B
16 17	18 19	20 21	22 23	24 25	26 27	28 29	30 31
ACQ_PI_C	ACQ_CI_C	REL_PI_C	REL_CI_C	ACQ_PI_D	ACQ_CI_D	RELPI_D	REL_CI_D
32 33	34 35	36 37	38 39	40 41	42 43	44 45	46 47
ACQ_PI_E	ACQ_CI_E	REL_PI_E	REL_CI_E	ACQ_PI_F	ACQ_CI_F	RELPI_F	REL_CI_F
48 49	50 51	52 53	54 55	56 57	58 59	60 61	62 63
ACQ_PI_G	ACQ_CI_G	REL_PI_G	REL_CI_G	ACQ_PI_H	ACQ_CI_H	REL_PI_H	REL_CI_H

Figure 16-102. PI/CI Region Format (Software Portal Ring Buffers)

Note: the PI/CI region is 64B, though the individual PI/CIs could be fit into a 32B region since each PI/CI is 8 bits.The ring buffers A through H can be assigned to any pool (pools 0 through 15) and the same pool can be assigned to multiple ring buffers. If reconfiguration is required (updating SWP*n*_{CONFIG,ASSIGN,BAR1,BAR2,_PICI_BAR}) software shall not modify these registers unless it has taken care to empty the ring buffers and ensure that hardware will not be accessing them during the reconfiguration.

- For **release** ring buffers; do not produce further buffer pointers into the ring buffer, and trigger hardware consumption of the current ring buffer entries by announcing the current producer indices to hardware (write SWP*n*_REL_PI_RING*k*, initiating consumption from those ring buffers), then waiting for the ring buffer to empty (PI=CI).
- For **acquire** ring buffers; consume all buffer pointers in the ring buffer, and do not update consumer indices (write SWP*n*_ACQ_CI_RING*k*) implying that the ring buffers are full. (if SIZE=0x7 and CI=0, then PI=128, and PI-CI=128 such that the ring buffer appears full to hardware).

Note that deasserting the enable bits (SWP*n*_EN) does *not* guarantee that the hardware will not access the ring buffer memory locations since memory accesses may still be in-flight at the time of the register write. No new memory accesses will be initiated while the enable bit is deasserted.

Note that the ring buffers should not be enabled (SWP*n*_EN) until after the base address registers (SWP*n*_BAR1, SWP*n*_BAR2, SWP*n*_PICI_BAR) and configuration registers (SWP*n*_CONFIG, SWPn_ASSIGN) have been written.

.

Offset 0x0920 (SWP<n>_CONFIG) Access: R/W
offset 0x1000
range n=0–7

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PD	0	0	0	0	0	BAR2_RINGS			0	0	0	0	0	RB_SIZE		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-103. Software Portal Configuration Register (SWP*n*_CONFIG)

Table 16-179. SWP*n*_CONFIG Bits Description

Field	Description
31–17	Reserved, read as 0.
16 PD	PI/CI region update Disable 0: enabled 1: disabled, no writes to the PI/CI region configured in PICI_BAR will occur
15–11	Reserved, read as 0.
10–8 BAR2_RINGS	number of acquire/release Ring buffer's assigned to base address 2 (SWPn_BAR2). 0: No Ring buffers start at SWPn_BAR2 1: Ring Buffers H start at SWPn_BAR2 2: Ring Buffers G, H start at SWPn_BAR2 3: Ring Buffers F, G, H start at SWPn_BAR2 4: Ring Buffers E, F, G, H start at SWPn_BAR2 5: Ring Buffers D, E, F, G, H start at SWPn_BAR2 6: Ring Buffers C, D, E, F, G, H start at SWPn_BAR2 7: Ring Buffers B, C, D, E, F, G, H start at SWPn_BAR2
7–3	Reserved, read as 0.
2–0 RB_SIZE	acquire/release ring buffer Size Ring buffer entries for all acquire & release ring buffers. entries = 2^RB_SIZE (min: 2^4=16, max: 2^7=128), where ring buffer entries are 4Bytes (32bit buffer pointers). RB_SIZE=0,1,2,3 are invalid.

Offset 0x0924 (SWP<n>_ASSIGN) Access: R/W
offset 0x1000
range n=0–7

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	A				B				C				D				E				F				G				H			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-104. Software Portal Ring Buffer Assignment Register (SWP*n*_ASSIGN)

Table 16-180. SWP*n*_ASSIGN Bits Description

Field	Description
31–28, 27–24, 23–20, 19–16, 15–12, 11–8, 7–4, 3–0 A,B,C,D,E, F,G,H	acquire & release ring buffer X pool assignment, where X is one of A,B,C,D,E,F,G,H 0:pool 0–15: pool 15

Offset 0x0928 (SWP<n>_EN) Access: R/W
offset 0x1000
range n=0–7

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	A	0	0	0	B	0	0	0	C	0	0	0	D	0	0	0	E	0	0	0	F	0	0	0	G	0	0	0	H	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-105. Software Portal Ring Buffer Enable Register (SWP*n*_EN)

Table 16-181. SWP*n*_EN Bits Description

Field	Description
31, 27, 23, 19, 15, 11, 7, 3 {A,B,C,D,E ,F,G,H}	Enable the ring buffer X, where X is one of A,B,C,D,E,F,G,H 0: disabled, no memory accesses will be initiated to this ring buffer, irrespective of the SWP*n_{ACQ,REL}_{CI,PI}* 1: enabled, ring buffer is active Note that even if the pool is disabled memory accesses may still occur for a time. See text on reconfiguration (above) for details.
30–28, 26–24, 22–20, 18–16, 14–12, 10–8, 6–4, 2–0	Reserved, read as 0.

Offset 0x0930 (SWP<n>_BAR1) Access: R/W
offset 0x1000
range n=0–7
0x0934 (SWP<n>_BAR2)
offset 0x1000
range n=0–7
0x0938 (SWP<n>_PICI_BAR)
offset 0x1000
range n=0–7

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

Figure 16-106. Software Portal Response 0 Register (SWP*n*_{BAR1,BAR2,PICI_BAR})

Table 16-182. SWP*n*_{BAR1,BAR2,PICI_BAR} Bits Description

Field	Description
31–6 ADDR	Acquire/release Base Address Register 32bit base address, 64 Byte aligned
5–0	Reserved, read as 0.

16.4.2.56 Software Portal Acquire Consumer Index (SWP*n*_ACQ_CI_RING*k*)

This register allows a software client to inform/update hardware on the current state of the consumer index (CI) of ring buffer *k* within portal *n*. The consumer index is a value managed by the software client (the consumer) that explicitly indicates the next item in the ring buffer that software shall consume and implicitly indicates to the producer items that have been consumed.

Updates/changes to this value will typically permit hardware (the producer) to acquire more buffers from a corresponding pool and load them into the corresponding ring buffer.

Offset 0x0940 (SWP<n>_ACQ_CI_RINGA) Access: R/W
offset 0x1000
range n=0–7
0x0948 (SWP<n>_ACQ_CI_RINGB)
offset 0x1000
range n=0–7
0x0950 (SWP<n>_ACQ_CI_RINGC)
offset 0x1000
range n=0–7
0x0958 (SWP<n>_ACQ_CI_RINGD)
offset 0x1000
range n=0–7
0x0960 (SWP<n>_ACQ_CI_RINGE)
offset 0x1000
range n=0–7
0x0968 (SWP<n>_ACQ_CI_RINGF)
offset 0x1000
range n=0–7
0x0970 (SWP<n>_ACQ_CI_RINGG)
offset 0x1000
range n=0–7
0x0978 (SWP<n>_ACQ_CI_RINGH)
offset 0x1000
range n=0–7

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

Figure 16-107. Software Portal Acquire Consumer Index Register (SWP*n*_ACQ_CI_RING*k*)

Table 16-183. SWP*n*_ACQ_CI_RING*k* Bits Description

Field	Description
31–8	Reserved, read as 0.
7–0 IDX	Acquire ring buffer Consumer Index.

16.4.2.57 Software Portal Release Producer Index (SWP*n*_REL_PI_RING*k*)

This register allows a software client to inform/update hardware on the current state of the producer index (PI) of ring buffer *k* within portal *n*. The producer index is a value managed by the software client (the producer) that explicitly indicates the next location in the ring buffer that software shall produce into and implicitly indicates to the consumer items that have been produced. Updates/changes to this value will typically permit hardware (the consumer) to release more buffers from the ring buffer into a corresponding pool.

Offset 0x0944 (SWP<n>_REL_PI_RINGA) Access: R/W
offset 0x1000
range n=0–7
0x094C (SWP<n>_REL_PI_RINGB)
offset 0x1000
range n=0–7
0x0954 (SWP<n>_REL_PI_RINGC)
offset 0x1000
range n=0–7
0x095C (SWP<n>_REL_PI_RINGD)
offset 0x1000
range n=0–7
0x0964 (SWP<n>_REL_PI_RINGE)
offset 0x1000
range n=0–7
0x096C (SWP<n>_REL_PI_RINGF)
offset 0x1000
range n=0–7
0x0974 (SWP<n>_REL_PI_RINGG)
offset 0x1000
range n=0–7
0x097C (SWP<n>_REL_PI_RINGH)
offset 0x1000
range n=0–7

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

Figure 16-108. Software Release Producer Index Register (SWP*n*_REL_PI_RING*k*)

Table 16-184. SWP*n*_REL_PI_RING*k* Bits Description

Field	Description
31–8	Reserved, read as 0.
7–0 IDX	Release ring buffer Producer Index.

16.4.2.58 Software Portal Acquire Producer Index (SWP*n*_ACQ_PI_RING*k*)

This register allows a software client to learn the current state of the producer index (PI) of ring buffer *k* within portal *n*. The producer index is a value managed by BMLite (the producer) that explicitly indicates the next location in the ring buffer that BMLite shall produce into and implicitly indicates to the consumer items that have been produced. Changes to this value indicate that there are new buffers available in the ring buffer for software (the consumer) to acquire.

Offset 0x9C0 (SWP<n>_ACQ_PI_RINGA) Access: R
offset 0x1000
range n=0–7
0x9C8 (SWP<n>_ACQ_PI_RINGB)
offset 0x1000
range n=0–7
0x9D0 (SWP<n>_ACQ_PI_RINGC)
offset 0x1000
range n=0–7
0x9D8 (SWP<n>_ACQ_PI_RINGD)
offset 0x1000
range n=0–7
0x9E0 (SWP<n>_ACQ_PI_RINGE)
offset 0x1000
range n=0–7
0x9E8 (SWP<n>_ACQ_PI_RINGF)
offset 0x1000
range n=0–7
0x9F0 (SWP<n>_ACQ_PI_RINGG)
offset 0x100
range n=0–7
0x9F8 (SWP<n>_ACQ_PI_RINGH)
offset 0x1000
range n=0–7

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

Figure 16-109. Software Portal Acquire Consumer Index Register (SWP*n*_ACQ_PI_RING*k*)

Table 16-185. SWP*n*_ACQ_PI_RING*k* Bits Description

Field	Description
31–8	Reserved, read as 0.
7–0 IDX	Acquire ring buffer Producer Index.

16.4.2.59 Software Portal Release Consumer Index (SWP*n*_REL_CI_RING*k*)

This register allows a software client to learn the current state of the consumer index (CI) of ring buffer *k* within portal *n*. The consumer index is a value managed by the BMLite (the consumer) that explicitly indicates the next item in the ring buffer that BMLite shall consume and implicitly indicates to the producer items that have been consumed. Changes to this value will typically permit software (the producer) to release more buffers into the ring buffer.

Offset 0x0C4 (SWP<n>_REL_CI_RINGA) Access: R
offset 0x1000
range n=0–7
0x0CC (SWP<n>_REL_CI_RINGB)
offset 0x1000
range n=0–7
0x0D4 (SWP<n>_REL_CI_RINGC)
offset 0x1000
range n=0–7
0x0DC (SWP<n>_REL_CI_RINGD)
offset 0x1000
range n=0–7
0x0E4 (SWP<n>_REL_CI_RINGE)
offset 0x1000
range n=0–7
0x0EC (SWP<n>_REL_CI_RINGF)
offset 0x1000
range n=0–7
0x0F4 (SWP<n>_REL_CI_RINGG)
offset 0x1000
range n=0–7
0x0FC (SWP<n>_REL_CI_RINGH)
offset 0x1000
range n=0–7

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

Figure 16-110. Software Release Producer Index Register (SWP*n*_REL_CI_RING*k*)

Table 16-186. SWP*n*_REL_CI_RING*k* Bits Description

Field	Description
31–8	Reserved, read as 0.
7–0 IDX	Release ring buffer Producer Index.

16.4.2.60 Message Queue Mode Register (MQMR)

The mode register allows for global control of system access parameters.

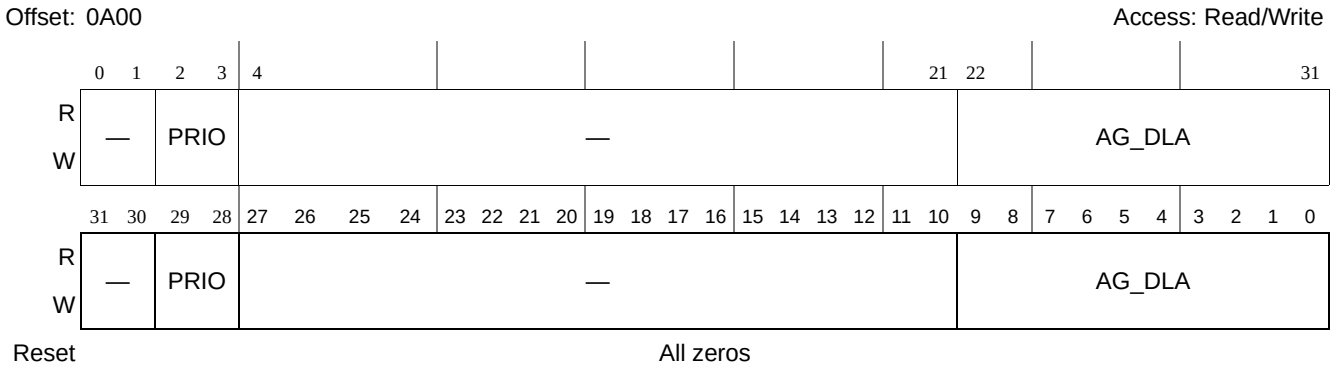


Figure 16-111. Message Queue Mode Register (MQMR)

Table 16-187. MQMR Field Descriptions

Bits	Name	Description
31–30	—	Reserved
29–28	PRIO	Priority used for memory access.
27–10	—	Reserved
9–0	AG_DLA	Arbitration group deadlock avoidance. This field controls the deadlock avoidance threshold for arbitration group selection. Deadlock avoidance is applied if a lower priority request has lost arbitration *N* number of times. A request that has reached the deadlock watermark will Round-Robin with other request in the same state and has priority over normal requests. A zero value will disable the deadlock avoidance mechanism and perform strict priority only.

16.4.2.61 Outbound Message Queue Dequeue Scheduler Configuration Register 1 (OMQDSCR1)

The dequeue scheduler configuration register 1 is used to configure the arbitration group of message queues based on the dequeue scheduling arbitration scheme. The maximum number of arbitration groups used in the system is defined by the highest assigned arbitration group number.

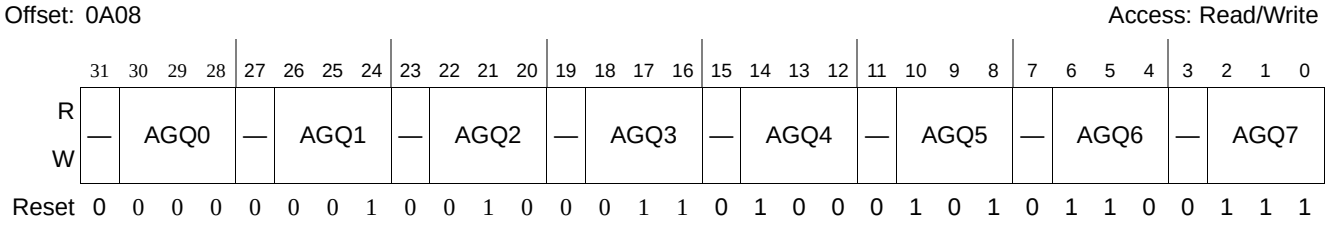


Figure 16-112. Outbound Message Queue Dequeue Scheduler Configuration Register 1 (OMQDSCR1)

Table 16-188. OMQDSCR1 Field Descriptions

Bits	Name	Description
31	—	Reserved
30–28	AGQ0	Arbitration group number for message queue 0. This field must be zero
27	—	Reserved
26–24	AGQ1	Arbitration group number for message queue 1. Same or one higher than AGQ0.
23	—	Reserved
22–20	AGQ2	Arbitration group number for message queue 2. Same or one higher than AGQ1.
19	—	Reserved
18–16	AGQ3	Arbitration group number for message queue 3. Same or one higher than AGQ2.
15	—	Reserved
14–12	AGQ4	Arbitration group number for message queue 4. Same or one higher than AGQ3.
11	—	Reserved
10–8	AGQ5	Arbitration group number for message queue 5. Same or one higher than AGQ4.
7	—	Reserved
6–4	AGQ6	Arbitration group number for message queue 6. Same or one higher than AGQ5.
3	—	Reserved
2–0	AGQ7	Arbitration group number for message queue 7. Same or one higher than AGQ6. This field determines the highest numbered arbitration group in the system.

16.4.2.62 Outbound Message Queue Dequeue Scheduler Configuration Register 2 (OMQDSCR2)

The dequeue scheduler configuration register 2 is used to configure the weight selection of message queues based on the dequeue scheduling arbitration scheme.

Offset: 0A0C Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—	AWQ0			—	AWQ1			—	AWQ2			—	AWQ3			—	AWQ4			—	AWQ5			—	AWQ6			—	AWQ7		
Reset	All zeros																															

Figure 16-113. Outbound Message Queue Dequeue Scheduler Configuration Register 2 (OMQDSCR2)

Table 16-189. OMQDSCR2 Field Descriptions

Bits	Name	Description
31	—	Reserved
30–28	AWQ0	Arbitration weight for message queue 0. Assigned weight = AWQ0, range 0 to 7.
27	—	Reserved
26–24	AWQ1	Arbitration weight for message queue 1.

Table 16-189. OMQDSCR2 Field Descriptions (Continued)

Bits	Name	Description
23	—	Reserved
22–20	AWQ2	Arbitration weight for message queue 2.
19	—	Reserved
18–16	AWQ3	Arbitration weight for message queue 3.
15	—	Reserved
14–12	AWQ4	Arbitration weight for message queue 4.
11	—	Reserved
10–8	AWQ5	Arbitration weight for message queue 5.
7	—	Reserved
6–4	AWQ6	Arbitration weight for message queue 6.
3	—	Reserved
2–0	AWQ7	Arbitration weight for message queue 7.

16.4.2.63 Message Queue Interrupt Enable Register (MQIER)

The message queue interrupt enable register determines if a detected error condition should cause an interrupt. An interrupt occurs if a bit in this register is set and the corresponding bit in the error detect register is also set. To clear the interrupt, write a 1 to the error detect register.

Offset: 0A10 Access: Read/Write

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

Figure 16-114. Message Queue Interrupt Enable Register (MQIER)

Table 16-190. MQIER Field Descriptions

Bits	Name	Description
31	MEIE	Multiple error interrupt enable. 0 Disabled. 1 Multiple error interrupt enabled.
30–1	—	Reserved
0	TEIE	Transaction error interrupt enable. 0 Disabled. 1 Transaction error interrupt enabled.

16.4.2.64 Message Queue Error Detect Registers (MQEDR)

The message queue error detect register indicates if an error condition was detected. Detection of a condition can be disabled through the error capture disable register. Some errors will cause the

error capture registers to be updated with additional information. Write 1 to clear the detected condition and the interrupt, if enabled.

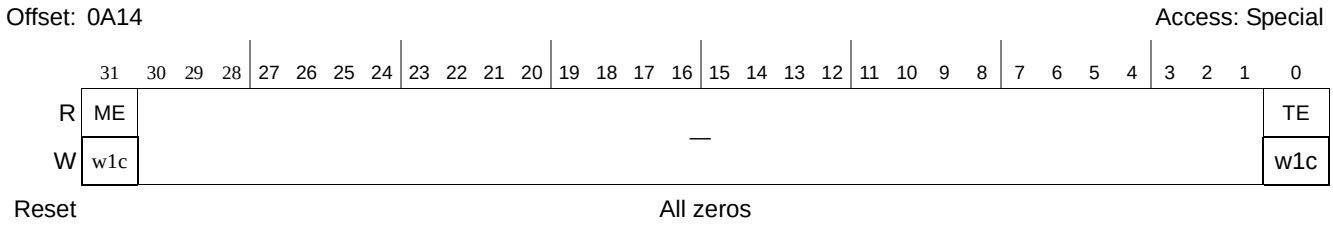


Figure 16-115. Message Queue Error Detect Register (MQEDR)

Table 16-191. MQEDR Field Descriptions

Bits	Name	Description
31	ME	**Multiple Error** Multiple errors of the same kind reported. Write 1 to clear. The following conditions will set this bit: • Transaction error encountered while previously set.
30–1	—	Reserved
0	TE	**Transaction Error** A read/write access to memory has caused an access violation. The Message Queue Error Capture Address Registers (MQECARs) has captured the memory address. Write 1 to clear.

16.4.2.65 Message Queue Error Capture Address Register (MQECAR)

This is the message queue error capture address register. When an enabled error condition occurs, additional information may be captured to indicate the source of the error. Software must clear the error condition to capture the next address. Transaction error (MQEDR[TE]) condition captures the memory address:.

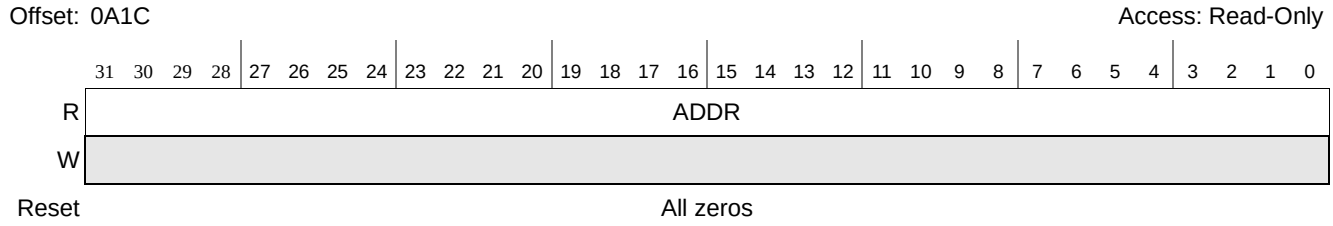


Figure 16-116. Message Queue Error Capture Address Register

Table 16-192. MQECAR Field Descriptions

Bits	Name	Description
31–0	ADDR	Address.

16.4.2.66 S/W Portal Depletion Entry Threshold Register (POOL*k*_SWDET)

The S/W portals can be notified through a BSCN functional interrupt when the number of free buffers available in pool *k* falls below the threshold set by this register. An entry threshold of zero or an entry threshold greater than the pool's exit threshold, will disable depletion reporting on pool *k* to all software portals.

Offset 0x0c00 (POOL<k>_SWDET) Access: R/W
offset 0x20
range k=0–15

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

Figure 16-117. S/W Portal Depletion Entry Threshold Register (POOL*k*_SWDET)

Table 16-193. POOL*k*_SWDET Bits Description

Field	Description
31–12	Reserved, read as 0
11–8 EXP	S/W Depletion Entry Threshold Exponent.
7–0 COEF	S/W Depletion Entry Threshold Coefficient. The threshold is COEF * 2^{EXP} buffer pointers. (max: 255* 2^{15} = 8355840 or ~8M, min: 0)

16.4.2.67 S/W Portal Depletion Exit Threshold Register (POOL*k*_SWDXT)

The S/W portals can be notified through a BSCN functional interrupt when the number of free buffers available in pool *k* rises above the threshold set by this register.

Offset 0x0c04 (POOL<k>_SWDXT) Access: R/W
offset 0x20
range k=0–15

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

Figure 16-118. S/W Portal Depletion Exit Threshold Register (POOL*k*_SWDXT)

Table 16-194. POOL*k*_SWDXT Bits Description

Field	Description
31–12	Reserved, read as 0
11–8 EXP	S/W Depletion Exit Threshold Exponent.
7–0 COEF	S/W Depletion Exit Threshold Coefficient. The threshold is COEF * 2^{EXP} buffer pointers. (max: 255* 2^{15} = 8355840 or ~8M, min: 0)

16.4.2.68 S/W Portal Depletion Count Register (POOL*k*_SDCNT)

The hardware will keep track of the number of times any pool *k* has entered the depletion state (See **Section 16.4.2.66**, *S/W Portal Depletion Entry Threshold Register (POOLk_SWDET)*). The register will not roll-over (255 max) until cleared by a read from the user.

Offset 0x0c08 (POOL<k>_SDCNT) Access: RR
offset 0x20
range k=0–15

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

Figure 16-119. BMLite S/W Depletion Count Register (POOL*k*_SDCNT)

Table 16-195. POOL*k*_SDCNT Bits Description

Field	Description
31–8	Reserved, read as 0.
7–0 N	8-bit linear count indicating number of times pool entered depletion, terminal count of 255 until cleared by a read (read-reset).

16.4.2.69 Pool Content Register (POOL*k*_CONTENT)

The hardware will keep track of the number of free buffers available in pool *k* and provides a snapshot using of the count through this register.

Note: This register is only a snapshot of the buffers in a pool that are either in the stockpile or in a FBPR stored in external memory. It does not include buffers pending release in the software or hardware portals.

Offset 0x0c0C (POOL<k>_CONTENT) Access: R
offset 0x20
range k=0–15

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

Figure 16-120. Pool Content Register (POOL*k*_CONTENT)

Table 16-196. POOL*k*_CONTENT Bits Description

Field	Description
31–24	Reserved, read as 0.
23–0 N	24-bit pool content value of number of free buffers.

16.4.2.70 H/W Portal Depletion Entry Threshold Register (POOL*k*_HWDET)

The H/W portals are notified through DCP portal interface, via the bpds[*k*] bit when the number of free buffers available in pool *k* falls below the threshold set by this register. An entry threshold of zero or an entry threshold greater than the pool's exit threshold, will disable depletion status for pool *k* to the DCP portal.

Offset 0x0c10 (POOL<k>_HWDET) Access: R/W
offset 0x20
range k=0–15

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

Figure 16-121. H/W Portal Depletion Entry Threshold Register (POOL*k*_HWDET)

Table 16-197. POOL*k*_HWDET Bits Description

Field	Description
31–12	Reserved, read as 0
11–8 EXP	H/W Depletion Entry Threshold Exponent.
7–0 COEF	H/W Depletion Entry Threshold Coefficient. The threshold is COEF * 2^{EXP} buffer pointers. (max: 255* 2^{15} = 8355840 or ~8M, min: 0)

16.4.2.71 H/W Portal Depletion Exit Threshold Register (POOL*k*_HWDXT)

The H/W portals are notified through DCP portal interface, via the bpds[*k*] bit when the number of free buffers available in pool *k* rises above the threshold set by this register.

Offset 0x0c14 (POOL<k>_HWDXT) Access: R/W
offset 0x20
range k=0–15

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

Figure 16-122. H/W Portal Depletion Exit Threshold Register (POOL*k*_HWDXT)

Table 16-198. POOL*k*_HWDXT Bits Description

Field	Description
31–12	Reserved, read as 0
11–8 EXP	H/W Depletion Exit Threshold Exponent.
7–0 COEF	H/W Depletion Exit Threshold Coefficient. The threshold is COEF * 2^{EXP} buffer pointers. (max: 255* 2^{15} = 8355840 or ~8M, min: 0)

16.4.2.72 H/W Portal Depletion Count Register (POOL*k*_HDCNT)

The hardware will keep track of the number of times any pool *k* has entered the depletion state (See **Section 16.4.2.70**, *H/W Portal Depletion Entry Threshold Register (POOLk_HWDET)*). The register will not roll-over (255 max) until cleared by a read from the user.

Offset 0x0c18 (POOL<k>_HDCNT) Access: R
offset 0x20
range k=0–15

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

Figure 16-123. H/W Depletion Count Register (POOL*k*_HDCNT)

Table 16-199. POOL*k*_HDCNT Bits Description

Field	Description
31–8	Reserved, read as 0.
7–0 N	8-bit linear count indicating number of times pool entered depletion, terminal count of 255 until cleared by a read (read-reset).

16.4.2.73 Free List Head Pointer Register (POOLk_HDPTR)

This is a debug/verification assist register.

This register indicates the 32bit address of the head of the singly linked list of FBPRs for this pool. Zero indicates an empty list.

Offset 0x0c1c (POOL<k>_HDPTR) Access: R
offset 0x20
range k=0–15

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

Figure 16-1. BMLite Pool Content Register (POOL*k*_HDPTR)

Table 16-1. POOL*k*_HDPTR Bits Description

Field	Description
0-25 ADDR	Address in external memory of last stored FBPR
26-31	Reserved, read as 0.

16.4.2.74 AXI Configuration Registers (AXI_CFG_[1–2])

AXI access to software portal locations is configured through these registers.

Offset 0x0e8C (AXI_CFG_1) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRS	SRP	SRC				SRA		SWS	SWP	SWC				SWA		0	0	0	0	0	0	0	0	PWS	PWP	PWC				PWA	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-124. AXI Configuration Register 2 (AXI_CFG_1)

Table 16-200. Register AXI_CFG_1 Bits Description

Field	Description
31 SRS	SWP ring AXI read access non-secure See FRS.
30 SRP	SWP ring AXI read access protection See FRP.
29–26 SRC	SWP ring AXI read transaction caching configuration See FRC.
25–24 SRA	SWP ring AXI read access priority See FRA.

Table 16-200. Register AXI_CFG_1 Bits Description

Field	Description
23 SWS	SWP ring AXI write access non-secure See FRS.
22 SWP	SWP ring AXI write access protection See FRP.
21–18 SWC	SWP ring AXI write transaction caching configuration See FRC.
17–16 SWA	SWP ring AXI write access priority See FRA
15–8	Reserved, read as 0.
7 PWS	SWP PICI region AXI write access non-secure See FRS.
6 PWP	SWP PICI region AXI write access protection See FRP.
5–2 PWC	SWP PICI region AXI write transaction caching configuration See FRC.
1–0 PWA	SWP PICI region AXI write access priority See FRA

Offset 0x0B90 (AXI_CFG_2) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FRS	FRP	FRC				FRA		FWS	FWP	FWC				FWA		BRS	BRP	BRC				BRA		BWS	BWP	BWC				BWA	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-125. AXI Configuration Register 2 (AXI_CFG_2)

Table 16-201. Register AXI_CFG_2 Bits Description

Field	Description
31 FRS	FBPR free list AXI read access non-secure0: secure access 1: non-secure access
30 FRP	FBPR free list AXI read access protection 0: normal access 1: privileged access

Table 16-201. Register AXI_CFG_2 Bits Description (Continued)

Field	Description
29–26 FRC	FBPR free list AXI read transaction caching configuration 0: noncacheable and nonbufferable 1: bufferable only 2: cacheable, but do not allocate 3: cacheable and bufferable, but do not allocate 6: cacheable write-through, allocate on reads only 7: cacheable write-back, allocate on reads only 10: cacheable write-through, allocate on writes only 11: cacheable write-back, allocate on writes only 14: cacheable write-through, allocate on both reads and writes 15: cacheable write-back, allocate on both reads and writes Note: all other values are reserved For bufferable configurations (1,3,7,11,15) bufferable accesses will be used if possible, non-buffered accesses otherwise. For non-bufferable configurations (0,2,6,10,14) only non-buffered accesses will be used.
25–24 FRA	FBPR free list AXI read access priority Sets the two bit Magenta priority for this access. All values are valid with 3: highest priority to 0: lowest priority
23 FWS	FBPR free list AXI write access non-secure See FRS.
22 FWP	FBPR free list AXI write access protection See FRP.
21–18 FWC	FBPR free list AXI write transaction caching configuration See FRC.
17–16 FWA	FBPR free list AXI write access priority See FRA.
15 BRS	Buffer pool AXI read access non-secure See FRS.
14 BRP	Buffer pool AXI read access protection See FRP.
13–10 BRC	Buffer pool AXI read transaction caching configuration See FRC.
9–8 BRA	Buffer pool AXI read access priority See FRA.
7 BWS	Buffer pool AXI write access non-secure See FRS.
6 BWP	Buffer pool AXI write access protection See FRP.
5–2 BWC	Buffer pool AXI write transaction caching configuration See FRC.
1-0 BWA	Buffer pool AXI write access priority See FRA.

16.4.2.75 Buffer Pointer Range Release Registers (BPRR_{CFG,START,END})

This register specifies a range of addresses to be released in a Buffer Pointer Range Release. The BMLite will release a Buffer Pointer BPRR_START into the pool specified in BPRR_CFG.POOLID, then increment by BPRR_CFG.STRIDE and repeat until > BPRR_END.

If BPRR_START and BPRR_END are aligned with respect to the BPRR_CFG.STRIDE, the range release will be inclusive such that both BPRR_START and BPRR_END will be released to the specified buffer pool.

To initiate a range release, the registers shall be updated in the following order:

1. write BPRR_START = first Buffer Pointer to release
2. write BPRR_END = last Buffer Pointer to release, and
3. write BPRR_CFG.POOLID and BPRR_CFG.STRIDE, initiating range release on write.

When BPRR_CFG is written in the last step, the hardware will assert BPRR_CFG.BUSY and start releasing buffers into the indicated buffer pool. The writes to these registers (BPRR_START, BPRR_END, BPRR_CFG) will be ignored while BPRR_CFG.BUSY is asserted. BPRR_CFG.BUSY can be read to see when the command has completed. It will deassert upon completion.

If a range release must stop due to FBPR exhaustion, the BPRR_CFG.BUSY will deassert and BPRR_CFG.ERR will assert. In this event BPRR_START will contain the next Buffer Pointer that would have been released. Once the FBPR exhaustion issue has been resolved by software, the interrupted range release may be resumed from where it left off by re-writing the original command but using the updated BPRR_START address.

For **FBPR** Range Releases, registers shall be set to the following: BPRR_CFG.STRIDE=6 (2^6=64B), BPRR_CFG.POOLID=16 (FBPR pool), and BPRR_START shall be 64B aligned such that the lower 6 bits are zero and BPRR_START shall never be set to zero. (FBPR pointer zero is the null pointer indicating end-of-linked-list.) Contravening these requirements will result in an ERR being reported.

If a range release is requested for an invalid POOLID, an ERR will be reported.

Offset 0x0e94 (BPRR_CFG) Access: R/W

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

Figure 16-126. Buffer Pointer Range Release Configuration Register (BPRR_CFG)

Table 16-202. Register BPRR_CFG Bits Description

Field	Description
31 BUSY	Range Release Busy. 0: idle, ready to accept new commands 1: active, writes to BPRR_CFG, BPRR_START, BPRR_END will be ignored
30 ERR	Range Release Error valid when BUSY=0, invalid otherwise 0: no error 1: an error occurred and the range release was halted, read BPRR_START to find the next address that would have been released
29–21	Reserved, read as 0.
20–16 POOLID	Range Release Pool ID Identifies into which pool Buffer Pointers of a Range Release will go. 0--15: pool 0 to 15respectively 16: FBPR pool all other values are invalid
15–5	Reserved, read as 0.
4–0 STRIDE	Range Release Stride(2^N) Identifies by how much the next buffer pointer should increment for each incremental release in the range release operation. min: 1=2^0, max: 2GB-1=2^31 Note: for FBPR releases stride shall be 64B increments, such that STRIDE=6 (2^6=64B).

Offset 0x0e98 (BPRR_START) Access: R/W

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

Figure 16-127. Buffer Pointer Range Release Start Register (BPRR_START)

Table 16-203. Register BPRR_START Bits Description

Field	Description
31–0 N	Buffer Pointer Range Release Start Index

Offset 0x0B9C (BPRR_END) Access: R/W

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

Figure 16-128. Buffer Pointer Range Release End Register (BPRR_END)

Table 16-204. Register BPRR_END Bits Description

Field	Description
31–0 N	Buffer Pointer Range Release End Index

16.4.2.76 Free Buffer Proxy Record Free Pool Count (FBPR_FPC)

The FBPR_FPC register is used to read a snapshot of the Free Buffer Proxy Record (FBPR) free pool size at a given time.

Offset 0x0eA0 (FBPR_FPC) Access: R

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

Figure 16-129. Free Buffer Proxy Record Free Pool Count Register (FBPR_FPC)

Table 16-205. Register FBPR_FPC Bits Description

Field	Description
31–24	Reserved, read as 0.
23–0 N	Free Pool Count. Total Free Buffer Proxy Record (FBPR) Free Pool Count in external memory

16.4.2.77 Free Buffer Proxy Record List Head Pointer Register (FBPR_HDPTR)

This is a debug/verification assist register.

BMLite maintains an FBPR free list in exactly the same manner as the buffer pool free lists.

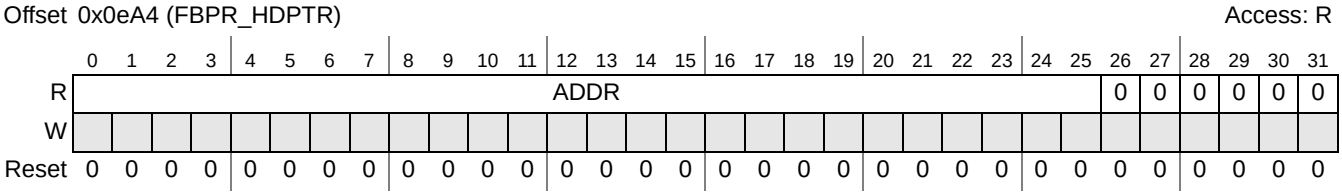


Figure 16-2. FBPR Head Pointer Register (FBPR_HDPTR)

Table 16-2. FBPR_HDPTR Bits Description

Field	Description
0-31 ADDR	external memory address of last stored FBPR

16.4.2.78 FBPR Free Pool Depletion Interrupt Threshold (FBPR_FP_THRES)

If the number of unused Free Buffer Proxy Records (FBPRs) fall below a threshold value configured using this register, then BMLite can assert an error interrupt (FPD) described in **Section 16.4.2.80**, *Error Interrupt Status Register (ERR_ISR)* if the interrupt is enabled to do so.

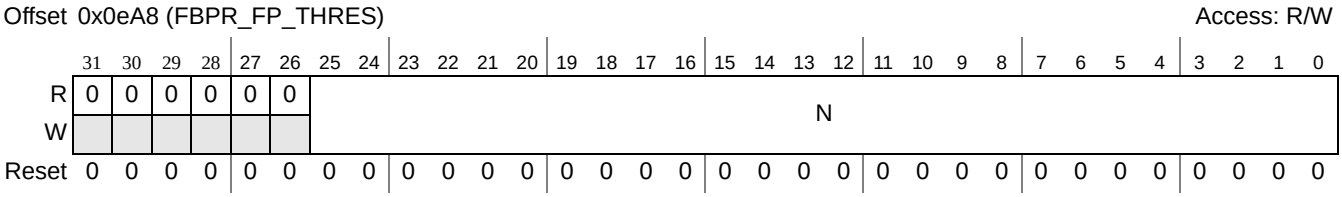


Figure 16-130. FBPR Free Pool Depletion Interrupt Threshold Register (FBPR_FP_THRES)

Table 16-206. Register FBPR_FP_THRES Bits Description

Field	Description
31–26	Reserved, read as 0.
25–0 N	FBPR Free Pool Depletion Interrupt Threshold An interrupt can be asserted (if enabled in ERR_IER) when FBPR_FPC falls below this threshold value.

16.4.2.79 Dynamic Power Management Configuration (DPM_CFG)

BMLite can be configured to delay the publication of when it is idle. It can also have dynamic power management enabled such that when the block becomes idle, clocks to portions of the block will be disabled, saving power.

Offset 0x0eAC (DPM_CFG) Access: R/W

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

Figure 16-131. Dynamic Power Management Configuration (DPM_CFG)

Table 16-207. Register DPM_CFG Bits Description

Field	Description
31 EN	Dynamic Power Management Enable. When EN is asserted, perform dynamic power management, turning off the internal clocks to a portion of the block while ipg_idle is asserted. EN does not affect the operation of ipg_idle. While performing dynamic power management, the block will still appear fully functional, ready to leave the idle state and process new work. 0: DPM disabled 1: DPM enabled
30–28	Reserved, read as 0.
27 RSVSM	Reserved, read as 0.
26 RSVSD	Reserved, read as 0.
25–4	Reserved, read as 0.
3–0 DELAY	Idle Delay Configure the delay as idle (ipg_idle). Will wait 2^DELAY cycles after the block finishes processing before asserting ipg_idle. (min DELAY=0: 2^0=1 cycle, max: DELAY=15: 2^15=32k cycles = 65.6us @ 500MHz, longer delays should be implemented in software by deasserting EN, setting DELAY=0 and performing SoC level power management)

16.4.2.80 Error Interrupt Status Register (ERR_ISR)

BMLite contains one dedicated interrupt line for signalling error conditions to software. This ISR register identifies the source of the interrupt within BMLite. Any asserted bit in this register can assert its interrupt line if enabled to do so in the ERR_IER register. Once a bit in this register is asserted, it will remain latched asserted until cleared by writing a 1 into its bit location.

Offset 0x0eB0 (ERR_ISR) Access: w1c

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BASCN															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BDP	BDB	0	0	0	0	0	0	0	EMCI	EMAI	0	FPD	MBEI	SBEI	0
W	w1c	w1c								w1c	w1c		w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-132. Error Interrupt Status Register (ERR_ISR)

Table 16-208. Register ERR_ISR Bits Description

Field	Description
31–16 BASCN	Buffer Pool Availability State change notification interrupt Pool 0: bit 15–Pool 15: bit 0
15 BDP	Bad DCP (Direct Connect Portal) Pool ID The DCP was directed to release a buffer into a pool that does not exist. (DCP pool ID signal is 6 bits but there are only 16 valid pools (0-15), if an invalid pool ID is specified this interrupt will occur.)
14 BDB	Bad DCP (Direct Connect Portal) Buffer Pointer The DCP was directed to release a buffer pointer that is too large. (DCP buffer pointer signals are 48 bits but BMLite only supports 32 bits, if the upper bits [47:32] are asserted this interrupt will occur.)
13–7	Reserved, read as 0.
6 EMCI	External memory corruption detected.
5 EMAI	External memory access exception.
4	Reserved, read as 0.
3 FPD	FBPR Low Watermark Interrupt. This interrupt is asserted if the FBPR free pool occupancy drops below a programmable threshold, see **Section 16.4.2.78**, *FBPR Free Pool Depletion Interrupt Threshold (FBPR_FP_THRES)*".
2 MBEI	Multi Bit ECC Error Interrupt. This interrupt is asserted if a multi bit ECC error is detected in one or more of the BMLite internal memories.
1 SBEI	Single Bit ECC Error Interrupt. This interrupt is asserted if the number of single bit ECC errors detected in one or more of the Man internal memories exceeds a programmable threshold. The threshold is described in **Section 16.4.2.85**, *Single Bit ECC Error Threshold Register (SBET)*.
0	Reserved, read as 0.

16.4.2.81 Error Interrupt Enable Register (ERR_IER)

The ERR_IER register provides the ability to individually enable each interrupt source to assert the error interrupt signal. If a bit location is asserted in both ERR_ISR and ERR_IER, the interrupt will be asserted. Clearing of an interrupt is achieved by first clearing the interrupt source if applicable, then clearing the associated bit in ERR_ISR.

Offset 0x0eB4 (ERR_IER) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BASCN															
Reset	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	BDP	BDB	0	0	0	0	0	0	0	EMCI	EMAI	0	FPD	MBEI	SBEI	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-133. Error Interrupt Enable Register (ERR_IER)

Table 16-209. Register ERR_IER Bits Description

Field	Description
31–0	Interrupt Enable bit for each bit in the Interrupt Status Register (ISR). All fields in this register carry the same assignment as in the ISR register, see **Section 16.4.2.80**, *Error Interrupt Status Register (ERR_ISR)*.

16.4.2.82 Interrupt Status Disable Register (ERR_ISDR)

The ERR_ISDR register controls reporting of interrupts in the ERR_ISR register. A bit in ERR_ISR will never be asserted if the corresponding bit in the ERR_ISDR register is set (that is, interrupt status is disabled). A set ISR bit will be cleared if the corresponding bit in the ISDR register is set.

Offset 0x0eB8 (ERR_ISDR) Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BASCN															
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	BDP	BDB	0	0	0	0	0	0	0	EMCI	EMAI	0	FPD	MBEI	SBEI	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-134. Interrupt Status Disable Register (ERR_ISDR)

Table 16-210. Register ERR_ISDR Bits Description

Field	Description
31–0	Interrupt Status Disable bit for each bit in the Interrupt Status Register (ISR). All fields in this register carry the same assignment as in the ISR register, see **Section 16.4.2.80**, *Error Interrupt Status Register (ERR_ISR)*.

16.4.2.83 Error Interrupt Inhibit Register (ERR_IIR)

This register allows the interrupt signal to be inhibited without modifying the enable or the status disable registers.

Offset 0x0eBC (ERR_IIR) Access: R/W

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

Figure 16-135. Error Interrupt Inhibit Register (ERR_IIR)

Table 16-211. Register ERR_IIR Bits Description

Field	Description
31–1	Reserved, read as 0.
0 I	Interrupt Inhibit. 0: Interrupt not inhibited. The interrupt signal may assert. 1:Interrupt inhibited. The interrupt signal will not assert.

16.4.2.84 Error Interrupt Force Register (ERR_IFR)

This is a debug/verification assist register. Writing a 1 to any valid bit in this register will set the corresponding bit in ERR_ISR, unless the corresponding bit in ERR_ISDR is also set. This is a write only register, reading this register will return 0.

Offset 0x0eC0 (ERR_IFR) Access: W

	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	0	0	0	0	0
W	BASCN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	9	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	BDP	BDB								EMCI	EMAI		FPD	MBEI	SBEI	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-136. Error Interrupt Force Register (ERR_IFR)

Table 16-212. Register ERR_IFR Bits Description

Field	Description
31–0	Interrupt Force bit for each bit in the Interrupt Status Register (ISR). All fields in this register carry the same assignment as in the ISR register, see **Section 16.4.2.80**, *Error Interrupt Status Register (ERR_ISR)*.

16.4.2.85 Single Bit ECC Error Threshold Register (SBET)

Local SRAM will be supported by ECC parity insert & error detection/correction logic for both single- and multi-bit errors. For single-bit errors, an 8-bit threshold must be configured for an error interrupt using this register.

Offset 0x0eD0 (SBET) Access: R/W

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

Figure 16-137. Single Bit Error Threshold Register (SBET)

Table 16-213. Register SBET Bits Description

Field	Description
31 ECD	Error Correction and Detection Disable. This field allows ECC error correction and detection to be individually disabled for each of BMLite's internal memories. The definition of each bit is as follows: 0 = ECC correction and ECC error reporting are enabled in the memory associated with this bit. 1 = ECC correction and ECC error reporting are disabled in the memory associated with this bit. The memory associated with this bit is the Stockpile memory.
30–8	Reserved, read as 0.
7–0 SBET	Single Bit Error Threshold. Threshold value for the number of single bit ECC errors in the BMLite internal memories that are detected before reporting an error. When the count in one of the SBEC registers is greater than SBET, an error is reported in ERR_ISR (unless it is disabled in ERR_ISDR).

16.4.2.86 Single Bit ECC Error Count Registers (SBEC)

This register provides a count of the number of single bit ECC errors on the internal memory. Eight bits are provided for the linear count, when it reaches a maximum of 255 it will hold until cleared on a read access.

Offset 0x0eD4 (SBEC) Access: RR

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

Figure 16-138. Single Bit Error Count Registers (SBEC)

Table 16-214. SBEC Bits Description

Field	Description
31–8	Reserved, read as 0.
7–0 N	Single Bit Error Count. Provides a count of the number of single bit ECC errors that have occurred when reading from the internal stockpile memory. This register is cleared on read. When the count reaches its max value (0xFF), it will stick at that value without rollover until this register is read.

16.4.2.87 External Memory Access Interrupt Capture Register (EMAI_ECR)

If an error occurs on a BMLite initiated read or write to external memory, an AXI exception may be returned. Since data read from external memory can be a data structure (FBPR) belonging to one of the buffer pool lists or the list of free FBPRs, then an exception that occurs on a read will cause BMLite to no longer use the affected list and possibly start a new one if there exists resources to do so. Like-wise a write would eventually result in a corresponding read exception and so is treated in exactly the same manner. BMLite can create more than one new list on the same pool in case of multiple errors, provided sufficient memory resources exist. All BMLite initiated read and write transactions are through an AXI interface to external memory. If an AXI error response is detected an interrupt will be signalled by setting EMAI in ERR_ISR. BMLite will detect the exception and use the EMAI_ECR register to report the event, the buffer pool or FBPR free list affected by the first exception, and whether or not more errors have occurred since the first one (that is, more than one list may be affected).All fields in this register, with the exception of the multiple exceptions (M) field in this register, are latched on the first exception. Upon further exceptions the M field will also latch. The content of this register will remain latched until the EMCI ERR_ISR is cleared.

Offset 0x0eE0 (EMAI_ECR) Access: R

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	M	RW	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

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

Figure 16-139. Exception Fetch/Flush Capture Register (EMAI_ECR)

Table 16-215. Register EMAI_ECR Bits Description

Field	Description
31 M	Multiple exceptions. More than one exception has occurred since the EMAI bit of ERR_ISR was asserted.
30 RW	Read/Write This bit indicates whether a read or a write caused the first exception. 0: Read 1: Write
29–9	Reserved, read as 0.
8–7 RESP	AXI Response The AXI response for the first exception memory transaction. 2: SLVERR -- slave error (the slave has returned an exception for this access) 3: DECERR -- decode error (no slave at this address)
6	Reserved, read as 0.
5–0 ID	ID. Identifies the buffer pool FBPR list in external memory or the affected Software Portal that has been affected by the first exception. ID = 0 to 15; List for Pool 0 to Pool 15 ID = 16; List of free FBPRs ID = 32 to 39; Software Portal 0 to Software Portal 7 All other values reserved.

16.4.2.88 External Memory Access Interrupt Address Register (EMAI_EADR)

If an exception occurs on a BMLite initiated read or write to external memory as described in EMAI_ECR, the external memory address used in the transaction will be captured in this register. The content of this register will remain latched until EMAI ERR_ISR is cleared.

Offset 0x0eE4 (EMAI_EADR) Access: R

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

Figure 16-140. Exception Fetch/Flush Address Register (EMAI_EADR)

Table 16-216. Register EMAI_EADR Bits Description

Field	Description
31–4 ADDR	Address. This field identifies the external memory address used for a read or write transaction that resulted in a data error described in **Section 16.4.2.87**, *External Memory Access Interrupt Capture Register (EMAI_ECR)*
3–0	Reserved, read as 0.

16.4.2.89 External Memory Corruption Interrupt Capture Register (EMCI_ECR)

To detect memory scribbling by other devices or FBPRs that are released more than once to the BMLite, the pool ID and a rotating 3-bit field per-free-list or buffer pool are recorded within the FBPR itself. When the FBPR is read, the recorded and expected values are compared. If this check fails then the FBPR has been corrupted and the buffer pool or free list is treated as in the AXI memory access exception case (including signalling an interrupt). This case will be indicated the EMCI ERR_ISR bit. All fields in this register and the Address register (EMCI_EADR), with the exception of the multiple exceptions (M) field in this register, are latched on the first exception. Upon further exceptions the M field will also latch. The content of this register will remain latched until the EMCI ERR_ISR is cleared.

Offset 0x0eF0 (EMCI_ECR) Access: R

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	M	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

	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	LID				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-141. External Memory Corruption Interrupt Capture Register (EMCI_ECR)

Table 16-217. Register EMCI_ECR Bits Description

Field	Description
31 M	Multiple exceptions. One or more exceptions have occurred since the EMCI ERR_ISR bit was asserted.

Table 16-217. Register EMCI_ECR Bits Description

Field	Description
30–5	Reserved, read as 0.
4–0 LID	List ID. This field identifies the BMLite buffer pool or free-list of data structures (FBPRs) in external memory that has been affected by the first error as described in the RESP & RW fields. LID = 0 to 15; List for Pool 0 to Pool 15 LID = 16; List of free FBPRs

16.4.2.90 External Memory Corruption Interrupt Address Register (EMCI_EADR)

If an exception occurs on a BMLite initiated read to external memory as described in EMCI_ECR, the external memory address used in the transaction will be captured in this register. The content of this register will remain latched until the EMCI ERR_ISR is cleared.

Offset 0x0eF4 (EMCI_EADR) Access: R

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

Figure 16-142. External Memory Corruption Interrupt Address Register (EMCI_EADR)

Table 16-218. Register EMCI_EADR Bits Description

Field	Description
31–6 ADDR	Address. This field identifies the external memory address used for a read or write transaction that resulted in a data error described in **Section 16.4.2.89**, *External Memory Corruption Interrupt Capture Register (EMCI_ECR)*
5–0	Reserved, read as 0.

16.4.2.91 IP Block Revision 1 Register (IP_REV_1)

The IP_REV_1 register is a read-only register which provides the unique IP block ID for the BMLite block, as well as major and minor revision numbers.

Offset 0x0eF8 (IP_REV_1) Access: R

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IP_ID																IP_MJ								IP_MN							
W																																
Reset	0x01D0																0x01								0x00							

Figure 16-143. IP Block Revision 1Register (IP_REV_1)

Table 16-219. Register IP_REV_1 Bits Description

Field	Description
31–16 IP_ID	IP Block ID. For the BMLite, this value is 0x01D0.
15–8 IP_MJ	Major revision. This is currently set to 0x01.
7–0 IP_MN	Minor revision. This is currently set to 0x00.

16.4.2.92 IP Block Revision 2 Register (IP_REV_2)

The IP_REV_2 register is a read-only register which provides BMLite block integration and configuration options.

Offset 0x0eFC (IP_REV_2) Access: R

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	IP_INT								IP_ERR								IP_CFG							
W																																
Reset	0	0	0	0	0	0	0	0	0x01								0x00								0x08							

Figure 16-144. IP Block Revision 2 Register (IP_REV_2)

Table 16-220. Register IP_REV_2 Bits Description

Field	Description
31–24	Reserved, read as 0.
23–16 IP_INT	Integration options. This is currently set to 0x01. This field indicates the Software Portal access method is via ring buffers and register access for 8 SWP.
15–8 IP_ERR	Errata revision level. This is currently set to 0x00.
7–0 IP_CFG	Configuration options. This is currently set to 0x08. This field indicates the number of buffer pools available in this implementation.

16.4.2.93 Message Unit Mode Register (MUMR)

The mode register allows for global control of hardware.

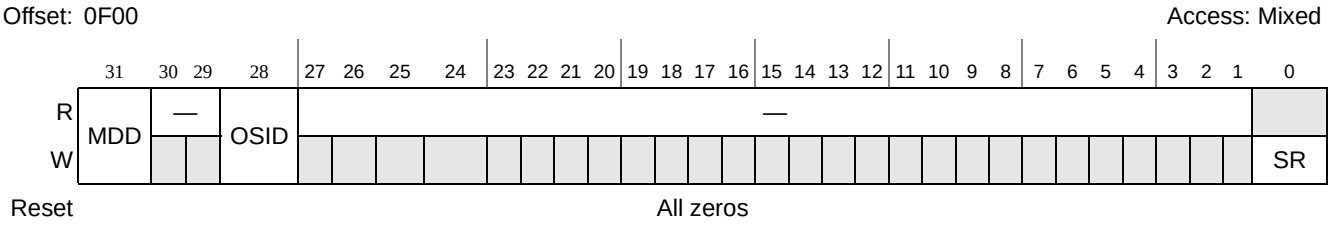


Figure 16-145. Message Unit Mode Register (MUMR)

Table 16-221. MUMR Field Descriptions

Bits	Name	Description
31	MDD	Inbound message descriptor write disable. 0 Inbound message descriptor written to memory for inbound transaction. 1 No message descriptor written. A performance improvement may be achieved by not writing the message descriptor. Note that several of the fields in the descriptor can be extracted from the classification unit(s) rules.
30–29	—	Reserved
28	OSID	Outbound segmentation interleaving disable. Segmentation interleaving allows the message unit to interleave segments from two or more transactions with the same destination device ID (regardless of type). This may increase performance for transmission but may also consume additional reassembly resources at the destination. 0 Segmentation interleaving enabled. 1 Segmentation interleaving disabled.
27–1	—	Reserved
0	SR	Soft reset. Write only. Setting this bit causes reset of all registers and state for the message unit, all state will be lost. Care should be taken when setting this bit as external interfaces must also be in an idle state for proper operation after reset. Writing zero to this bit has no effect.

16.4.2.94 Message Unit Status Register (MUSR)

The status register indicates current status of hardware resources.

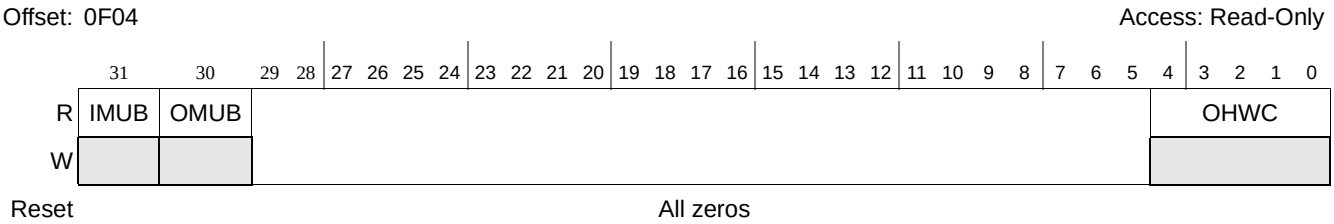


Figure 16-146. Message Unit Status Register (MUSR)

Table 16-222. MUSR Field Descriptions

Bits	Name	Description
31	IMUB	Inbound message unit busy. Read-only. 0 Inbound message unit is idle. 1 Inbound message unit operation in progress.
30	OMUB	Outbound message unit busy. Read-only. 0 Outbound message unit is idle. 1 Outbound message unit operation in progress.
29–5	—	Reserved
4–0	OHWC	Open hardware contexts. The current number of hardware context used for reassembly. A zero value indicates that all hardware contexts are available for reassembly. Based on the MURCAR[0–2] values, if the open threshold has been reached, new transactions are retried (Type10/11) or dropped (Type9).

16.4.2.95 Message Unit Interrupt Enable Registers (MUIER)

The message unit interrupt enable register determines if a detected status or error condition should cause an interrupt. An interrupt occurs if a bit in this register is set and the corresponding bit in the error detect register is also set. To clear the interrupt, write a 1 to the error detect register.

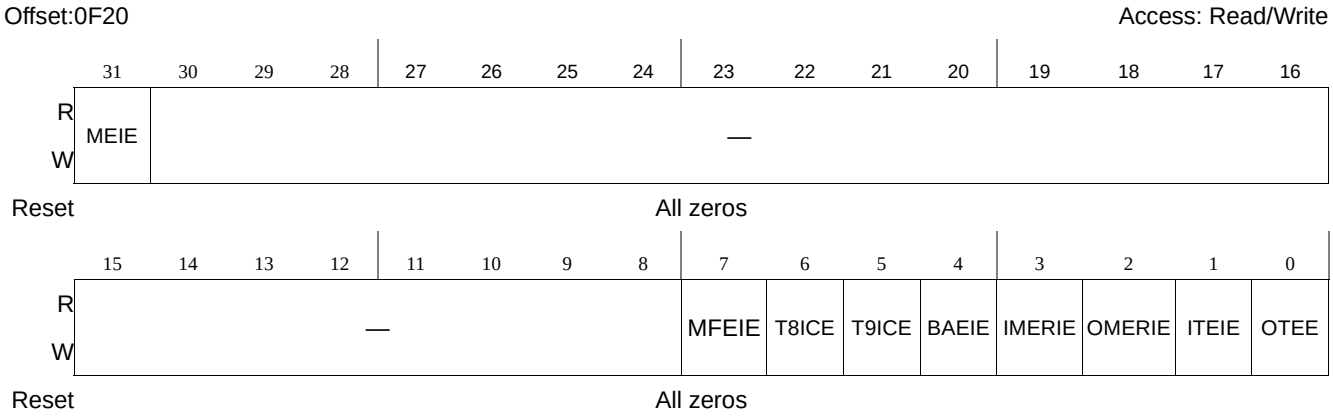


Figure 16-147. Message Unit Interrupt Enable Register (MUIER)

Table 16-223. MUIER Field Descriptions

Bits	Name	Description
31	MEIE	Multiple error interrupt enable. 0 Disabled. 1 Multipe error interrupt enable.
30–8	—	Reserved
7	MFEIE	Message format error interrupt enable. 0 Disabled. 1 Message format error interrupt enabled.
6	T8ICE	Type8 interrupt coalescing drop threshold enable. 0 Disabled. 1 Interrupt coalescing drop threshold enabled.

Table 16-223. MUIER Field Descriptions (Continued)

Bits	Name	Description
5	T9ICE	Type8 interrupt coalescing drop threshold enable. 0 Disabled. 1 Interrupt coalescing drop threshold enabled.
4	BAEIE	Buffer allocation error interrupt enable. 0 Disabled. 1 Buffer allocation error interrupt enabled.
3	IMERIE	Inbound message queue enqueue rejection interrupt enable. 0 Disabled. 1 Inbound message queue enqueue rejection interrupt enabled.
2	OMERIE	Outbound message queue enqueue rejection interrupt enable. 0 Disabled. 1 Outbound message queue enqueue rejection interrupt enabled.
1	ITEIE	Inbound transaction error interrupt enable. 0 Disabled. 1 Inbound transaction error interrupt enabled.
0	OTEIE	Outbound transaction error interrupt enable. 0 Disabled. 1 Outbound transaction error interrupt enabled.

16.4.2.96 Message Unit Error Detect Registers (MUEDR).

Offset:0F24 Access: Special

	31	30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7	6	5	4	3	2	1	0
R	ME	—						MFE	T8IC	T9IC	BAE	IMER	OMER	ITE	OTE
W	w1c							w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	All zeros														

Figure 16-148. Message Unit Error Detect Register (MUEDR)

The message unit error detect register indicates if an error condition was detected. Detection of a condition can be disabled through the error capture disable register. Some errors will cause the error capture registers to be updated with additional information. Write 1 to clear the detected condition and the interrupt, if enabled.

Note: As noted in **Section 25.2.9.1**, *Debug Errors*, on page 25-25, all message errors are ORed with Serial RapidIO errors for reporting to the cores. If an error is reported to the core and no message errors are flagged in MUEDR, check the Serial RapidIO status registers to determine the source of the error/interrupt.

Table 16-224. MUEDR Field Descriptions

Bits	Name	Description
31	ME	Multiple error. Multiple errors of the same kind reported. Write 1 to clear.
30–8	—	Reserved
7	MFE	Message format error. One of the following conditions will cause the error: • A transaction of priority 3 was received for Type10 doorbell or Type11 message. Transaction has been dropped. • Spurious (or duplicate) response received. Response has been dropped.

Table 16-224. MUEDR Field Descriptions (Continued)

Bits	Name	Description
6	T8IC	Type8 interrupt coalescing drop threshold exceeded. The drop counter value in MUT8DCR has exceeded the interrupt coalescing interrupt threshold MUICR[T8ICDT]. Write 1 to clear.
5	T9IC	Type9 interrupt coalescing drop threshold reached. The drop counter value in MUT9DCR has exceeded the interrupt coalescing interrupt threshold MUICR[T9ICDT]. Write 1 to clear.
4	BAE	Buffer allocation error. A data buffer could not be acquired for an inbound transaction. This has caused an inbound transaction to complete with an internal error. Write 1 to clear.
3	IMER	Inbound message queue enqueue rejection. In an attempt to enqueue, the inbound message queue rejected the command descriptor. Rejection occurs due to a disabled/full queue or a memory transaction error. This error has caused an inbound transaction to be discarded. The Message Unit Error Capture CD Registers (MUECCDR0–3) will contain the command descriptor of the attempted enqueue operation. Write 1 to clear.
2	OMER	Outbound message queue enqueue rejection. In an attempt to enqueue, the completion queue rejected the command descriptor. Rejection occurs due to a disabled queue or a memory transaction error. This error has caused an outbound completion event to be discarded. Write 1 to clear.
1	ITE	Inbound transaction error. A write access to memory has caused a violation. The Message Unit Error Capture Address Register will contain the violating memory address. Write 1 to clear.
0	OTE	Outbound transaction error. A read access to memory has caused a violation. The Message Unit Error Capture CD Registers (MUECCDR[0–3]) will contain the command descriptor which caused the memory transaction error. Write 1 to clear.

16.4.2.97 Message Unit Interrupt Coalescing Registers (MUICR)

The message unit interrupt coalescing register configures the threshold parameters for interrupt coalescing associated with inbound Type8/9 transactions that may be dropped to congestion or failed classification.

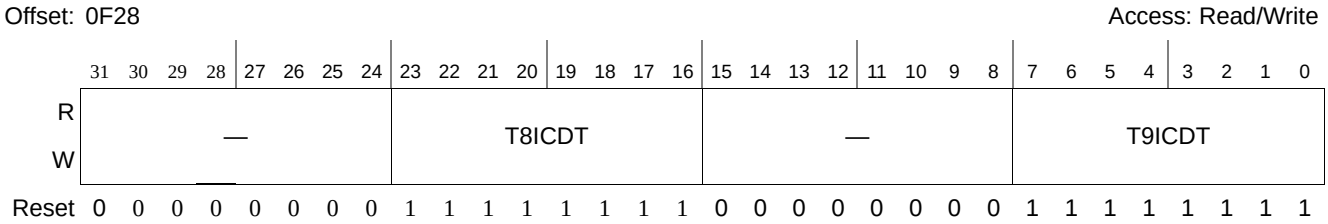


Figure 16-149. Message Unit Interrupt Coalescing Registers (MUICR)

Table 16-225. MUICR Field Descriptions

Bits	Name	Description
31–24	—	Reserved
23–16	T8ICDT	Type8 interrupt coalescing drop threshold. This values determines how many RapidIO Type8 maintenance port-writes are allowed to be dropped before setting the error detect bit MUEDR[T8IC]. MUEDR[ME] may be set if the threshold is crossed a second time before clearing the error detect bit. The total drop count is available in register MUT8DCR[DROP_CNT].

Table 16-225. MUICR Field Descriptions (Continued)

Bits	Name	Description
15–8	—	Reserved
7–0	T9ICDT	Type9 interrupt coalescing drop threshold. This values determines how many RapidIO Type9 segments are allowed to be dropped before setting the error detect bit MUEDR[T9IC]. MUEDR[ME] may be set if the threshold is crossed a second time before clearing the error detect bit. The total drop count is available in register MUT9DCR[DROP_CNT].

16.4.2.98 Message Unit T8 Drop Counter Registers (MUT8DCR)

The message unit Type8 drop counter register tracks number of inbound maintenance port-writes dropped due to pending port-writes or failed classification.

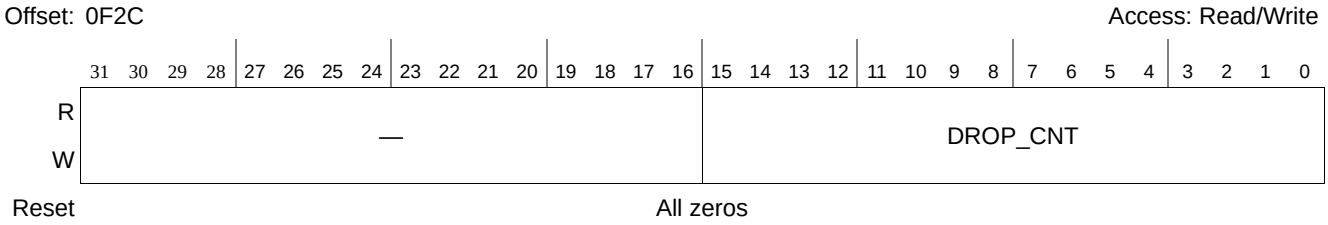


Figure 16-150. Message Unit T8 Drop Counter Registers (MUT8DCR)

Table 16-226. MUT8DCR Field Descriptions

Bits	Name	Description
31–16	—	Reserved
15–0	DROP_CNT	Drop count for RapidIO Type8 maintenance port-writes due to hardware resource restrictions (pending port-writes) or failed classification. Software should clear this register field by reading if an interrupt has occured due to the counter exceeding the interrupt coalescing threshold value MUICR[T8ICDT] to avoid additional interrupts.

16.4.2.99 Message Unit T9 Drop Counter Registers (MUT9DCR)

The message unit Type9 drop counter register tracks number of inbound PDU segments dropped due to exhausted hardware resources or failed classification.

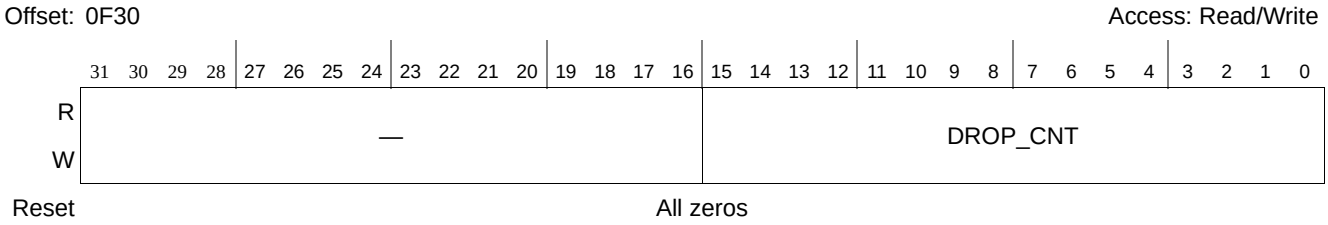


Figure 16-151. Message Unit T9 Drop Counter Registers (MUT9DCR)

Table 16-227. MUT9DCR Field Descriptions

Bits	Name	Description
31–16	—	Reserved
15–0	DROP_CNT	Drop count for RapidIO Type9 PDU segments due to exhausted hardware resources or failed classification. Software should clear this register field by reading if an interrupt has occurred due to the counter exceeding the interrupt coalescing threshold value MUICR[T9ICDT] to avoid additional interrupts.

16.4.2.100 Message Unit Error Capture MQ Register (MUECMQR)

This is the message unit error capture message queue register. When an enabled error condition occurs, additional information may be captured to indicate the source of the error. Software must clear the error condition(s) to capture the next error. The following error condition(s) will capture the message queue identifier:

- Inbound enqueue rejection (MUEDR[IMER])

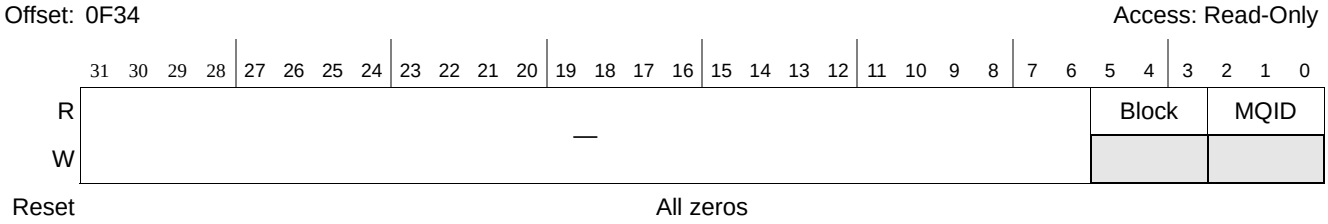


Figure 16-152. Message Unit Error Capture MQ Register (MUECMQR)

Table 16-228. MUECMQR Field Descriptions

Bits	Name	Description
31–6	—	Reserved
5–3	UNIT	Block associated with the error.
2–0	MQID	Message queue ID associated with the error.

16.4.2.101 Message Unit Error Capture CD Register 0 (MUECCDR0)

This is the message unit error capture command descriptor register 0. When an enabled error condition occurs, additional information may be captured to indicate the source of the error. Software must clear the error condition(s) to capture the next command descriptor. The following error condition(s) will capture the command descriptor:

- Outbound transaction error (MUEDR[OTE])..

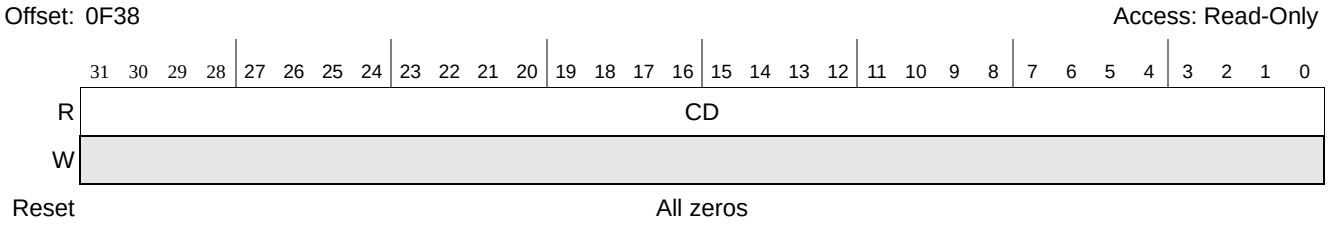


Figure 16-153. Message Unit Error Capture CD Register 0

Table 16-229. MUECCDR0 Field Descriptions

Bits	Name	Description
31–0	CD	Command descriptor associated with the error, bits 127–96.

16.4.2.102 Message Unit Error Capture CD Register 1 (MUECCDR1)

This is the message unit error capture command descriptor register 1. See **Section 16.4.2.101**, *Message Unit Error Capture CD Register 0 (MUECCDR0)* for description.

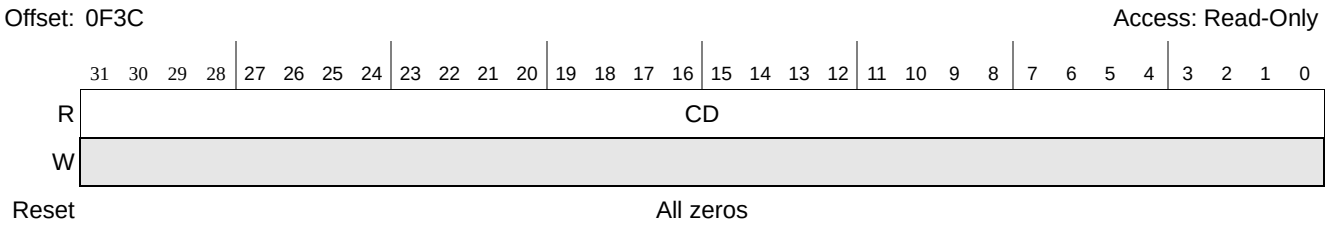


Figure 16-154. Message Unit Error Capture CD Register 1

Table 16-230. MUECCDR1 Field Descriptions

Bits	Name	Description
31–0	CD	Command descriptor associated with the error, bits 95–64.

16.4.2.103 Message Unit Error Capture CD Register 2 (MUECCDR2)

This is the message unit error capture command descriptor register 2. See **Section 16.4.2.101**, *Message Unit Error Capture CD Register 0 (MUECCDR0)* for description.

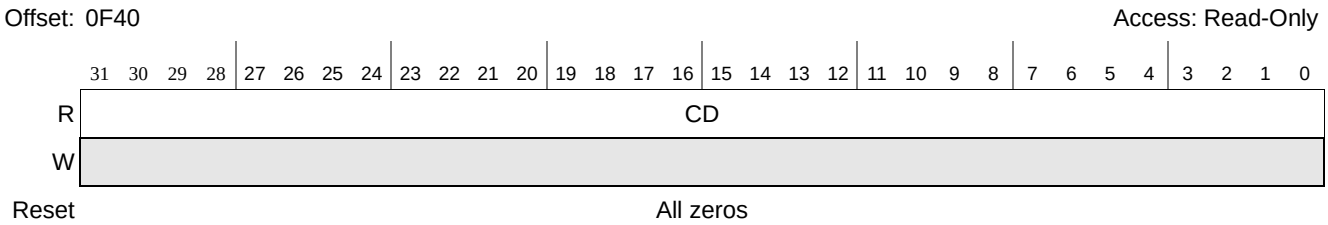


Figure 16-155. Message Unit Error Capture CD Register 2

Table 16-231. MUECCDR2 Field Descriptions

Bits	Name	Description
31–0	CD	Command descriptor associated with the error, bits 63–32.

16.4.2.104 Message Unit Error Capture CD Register 3 (MUECCDR3)

This is the message unit error capture command descriptor register 3. See **Section 16.4.2.101**, *Message Unit Error Capture CD Register 0 (MUECCDR0)* for description.

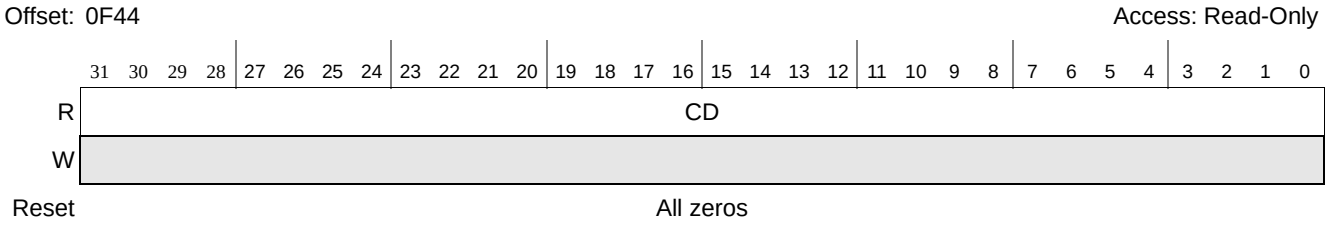


Figure 16-156. Message Unit Error Capture CD Register 3

Table 16-232. MUECCDR3 Field Descriptions

Bits	Name	Description
31–0	CD	Command descriptor associated with the error, bits 31–0.

16.4.2.105 Message Unit Error Capture Address Register (MUECAR)

This is the message unit error capture address register. When an enabled error condition occurs, additional information may be captured to indicate the source of the error. Software must clear the error condition to capture the next address. An inbound transaction error (MUEDR[ITE]) condition(s) captures the memory address:

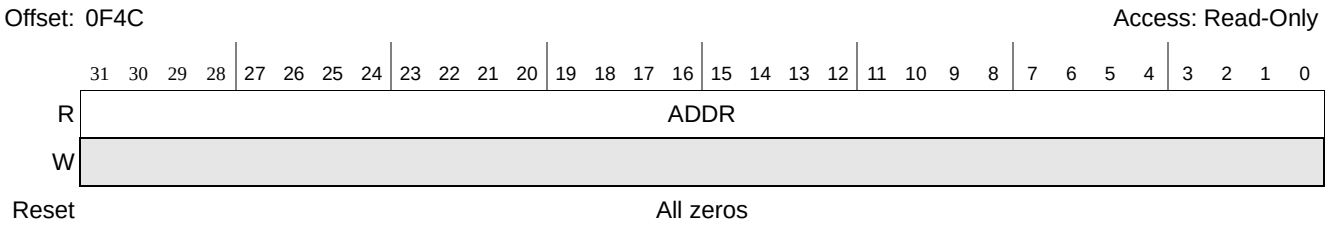


Figure 16-157. Message Unit Error Capture Address Register

When an enabled error condition occurs, additional information may be captured to indicate the source of the error. Software must clear the error condition to capture the next address.

The following error condition(s) captures the memory address:

- Inbound transaction error (MUEDR[ITE]).

Table 16-233. MUECAR Field Descriptions

Bits	Name	Description
31–0	ADDR	Memory address associated with the error. This is the lower address bits 31–0. For Type8, the address indicates the allocated buffer pointer address and may not be precise.

16.4.2.106 Message Unit Arbitration Weight Register (MUAWR)

Offset: 0F50 Access: Read//Write

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

Figure 16-158. Message Unit Arbitration Weight Register (MUAWR)

The message unit arbitration weight register controls the weight of several outbound arbitration points, including segmentation unit starvation threshold and outbound RapidIO request/response/ flow control arbitration.

The weighted round-robin arbitration scheme used works as follows:

1. All selection counters start at 0.
2. A requestor is eligible for selection if its selection counter does not equal its programmed weight.
3. A work conserving round-robin selection is made among all eligible requestors. The selection counter for the selected requestor is incremented if the threshold has not been reached.
4. The counter of the selected port is reset when the counter has reached the threshold. In this instance, the round-robin arbiter is also updated.

Table 16-234. MUAWR Field Descriptions

Bits	Name	Description
31–28	OREQW	Outbound request arbitration weight. This field controls the arbitration weight in a weighted Round-Robin scheme for access to the outbound RapidIO port(s).
27–24	ORSPW	Outbound response arbitration weight. This field controls the arbitration weight in a weighted Round-Robin scheme for access to the outbound RapidIO port(s).
23–20	OFCW	Outbound flow control arbitration weight. This field controls the arbitration weight in a weighted Round-Robin scheme for access to the outbound RapidIO port(s).

Table 16-234. MUAWR Field Descriptions (Continued)

Bits	Name	Description
19–10	—	Reserved.
9–0	OSST	Outbound segmentation strict-priority threshold. This field controls the deadlock avoidance threshold for granting access to a segmentation unit for a lower priority arbitration group. Deadlock avoidance is applied if a lower priority request has lost arbitration N number of times. A request that has reached the deadlock watermark will round-robin with other requests in the same state and has priority over normal requests. Default is maximum value. A zero value will disable the deadlock avoidance mechanism and perform strict priority only.

16.4.2.107 Message Unit Outbound Interleaving Mask Register (MUOIMR)

The message unit outbound interleaving mask register provides a more coarse control of transaction interlock for the purpose of interleaving transactions based on outbound ordering rules. The setting of a mask bit excludes it for comparison when determining outbound interleaving. For example, an endpoint may be assigned a range of destination IDs, but may require strict ordering for the entire range. This could for example be used to provide additional mailbox/letter combinations by utilizing multiple destination IDs. The setting of bits in this mask register must be contiguous.

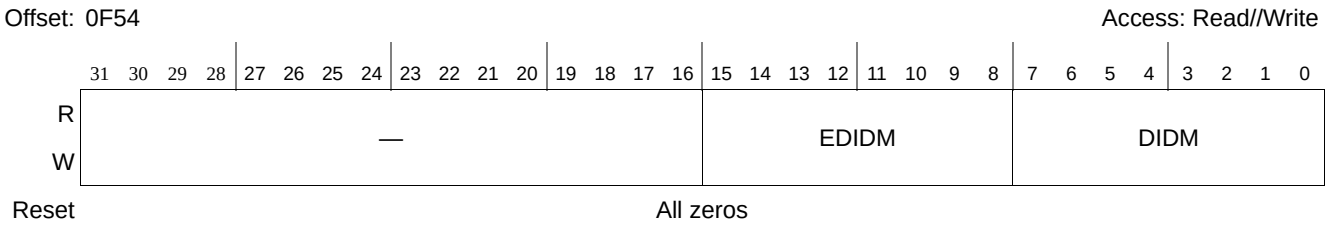


Figure 16-159. Message Unit Outbound Interleaving Mask Register (MUOIMR)

Table 16-235. MUOIMR Field Descriptions

Bits	Name	Description
31–16	—	Reserved.
15–8	EDIDM	Extended destination ID bit mask . Most significant byte of a 16-bit destination ID when operating in large transport mode. Reserved when operating in small transport mode.
7–0	DIDM	Destination ID bit mask. Contains the destination ID field of the transaction (Device ID of the destination).

16.4.2.108 Message Unit Segmentation Execution Privilege Register 0 (MUSEPR0) and Message Unit Segmentation Execution Privilege Register 1 (MUSEPR1)

The message unit segmentation execution privilege registers specify the arbitration group execution privileges for each segmentation unit present. Writing the arbitration group assignment register, OMQDSCR1, will re-initialize the execution privileges for all segmentation units to a

setting based on the maximum number of arbitration groups supported; see **Table 16-236**. Writing this register will override the default setting.

Table 16-236. SU Execution Privileges Based on Maximum AG Setting.

AGs Supported	SU0	SU1	SU2	SU3	SU4	SU5
1	0	0	0	0	0	0
2	0	0	0	0,1	0,1	0,1
3	0	0	0,1	0,1,2	0,1,2	0,1,2
4	0	0	0,1	0,1,2	0,1,2,3	0,1,2,3
5	0	0,1	0,1,2	0,1,2,3	0,1,2,3,4	0,1,2,3,4
6	0	0,1	0,1,2	0,1,2,3	0,1,2,3,4	0,1,2,3,4,5
7	0,1	0,1,2	0,1,2,3	0,1,2,3,4	0,1,2,3,4,5	0,1,2,3,4,5,6
8	0,1	0,1,2	0,1,2,3	0,1,2,3,4	0,1,2,3,4,5	0,1,2,3,4,5,6,7

Offset: 0F70 Access: Read//Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SUP0								SUP1								SUP2								SUP3							
W																																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	1	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	1

Figure 16-160. Message Unit Segmentation Execution Privilege Register (MUSEPR0)

Table 16-237. MUSEPR0 Field Descriptions

Bits	Name	Description
31–24	SUP0	Segmentation unit 0 arbitration group 7–0 execution privileges. A set bit indicates that the arbitration group is allowed to execution on this segmentation unit. for example, 0x03 allows only arbitration group 1-0 execution privileges.
23–16	SUP1	Segmentation unit 1 arbitration group 7–0 execution privileges.
15–8	SUP2	Segmentation unit 2 arbitration group 7–0 execution privileges.
7–0	SUP3	Segmentation unit 3 arbitration group 7–0 execution privileges.

Offset: 0FEC Access: Read//Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SUP4								SUP5								—															
W																																
Reset	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-161. Message Unit Segmentation Execution Privilege Register 1 (MUSEPR1)

Table 16-238. MUSEPR1 Field Descriptions

Bits	Name	Description
31–24	SUP4	Segmentation unit 4 arbitration group 7-0 execution privileges.
23–16	SUP5	Segmentation unit 5 arbitration group 7-0 execution privileges.
15–0	—	Reserved

16.4.2.109 Message Unit Reassembly Context Assignment Registers 0–2 (MURCAR0–2)

The message unit reassembly context assignment registers controls the management of inbound hardware reassembly contexts on a per flow basis. The total number of system available reassembly contexts is 16. A flow with one or more dedicated resources does not share the generic pool. The amount of dedicated and generic contexts should not exceed the total available number of reassembly contexts or undefined results may occur. The total number of open contexts at one given time can be determined through register MUSR[OHWC].

Offset: 0F80 Access: Read//Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—				FLOW0				—				FLOW1				—				FLOW2				—				FLOW3			
Reset	All zeros																															

Figure 16-162. Message Unit Reassembly Context Assignment Register 0 (MURCAR0)

Table 16-239. MURCAR0 Field Descriptions

Bits	Name	Description
31–29	—	Reserved
28–24	FLOW0	Dedicated reassembly contexts for flow 0. h00 Shares generic context pool with other flows h01 1 dedicated reassembly context h02 2 dedicated reassembly contexts ... h1F 31 dedicated reassembly contexts
23–21	—	Reserved
20–16	FLOW1	Dedicated reassembly contexts for flow 1.
15–13	—	Reserved
12–8	FLOW2	Dedicated reassembly contexts for flow 2.
7–5	—	Reserved
4–0	FLOW3	Dedicated reassembly contexts for flow 3.

Offset: 0F84 Access: Read/Write

	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
R W	—	FLOW4	—	FLOW5	—	FLOW6	—	FLOW7
Reset	All zeros							

Figure 16-163. Message Unit Reassembly Context Assignment Register 1 (MURCAR1)

Table 16-240. MURCAR1 Field Descriptions

Bits	Name	Description
31–29	—	Reserved
28–24	FLOW4	Dedicated reassembly contexts for flow 4. h00 Shares generic context pool with other flows h01 1 dedicated reassembly context h02 2 dedicated reassembly contexts ... h1F 31 dedicated reassembly contexts
23–21	—	Reserved
20–16	FLOW5	Dedicated reassembly contexts for flow 5.
15–13	—	Reserved
12–8	FLOW6	Dedicated reassembly contexts for flow 6.
7–5	—	Reserved
4–0	FLOW7	Dedicated reassembly contexts for flow 7.

Offset: 0F88 Access: Read/Write

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

Figure 16-164. Message Unit Reassembly Context Assignment Register 2 (MURCAR2)

Table 16-241. MURCAR2 Field Descriptions

Bits	Name	Description
31–29	—	Reserved
28–24	GEN	Generic reassembly contexts available for all flows without dedicated resources (that is, zero). h00 0 generic reassembly contexts h01 1 generic reassembly context h02 2 generic reassembly contexts ... h1F 31 generic reassembly contexts
23–0	—	Reserved

16.4.2.110 IP Block Revision Register 0 (IPBRR0) for eMSG

Offset: 0x0bf8 Access: Read only

	31															16	15							8	7							0
R	IP_ID																IP_MJ								IP_MN							
W																																
Reset	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Figure 16-165. IP Block Revision Register 0 (IPBRR0)

The IP block revision register 0 (IPBRR0), shown in **Figure 16-165**, is used to identify the eMSG block and revision. **Table 16-242** describes the IPBRR0 fields.

Table 16-242. IPBRR0 Field Descriptions

Bits	Name	Description
31–16	IP_ID	IP block ID = 0x0A20
15–8	IP_MJ	Major revision = 0x01
7–0	IP_MN	Minor revision = 0x00

16.4.2.111 IP Block Revision Register 1 (IPBRR1) for eMSG

Offset: 0x0BFC Access: Read only

	31							24	23							16	15							8	7							0
R	—								IP_INT								IP_MNT								IP_CFG							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1

Figure 16-166. IP Block Revision Register 1 (IPBRR1)

The IP block revision register 1 (IPBRR1), shown in **Figure 16-166**, is used to identify eMSG block configuration. **Table 16-243** describes the IPBRR1 fields.

Table 16-243. IIPBRR1 Field Descriptions

Bits	Name	Description
31–24	—	Reserved = 0x00
23–16	IP_INT	IP block integration options = 0x00

Table 16-243. IIPBRR1 Field Descriptions (Continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>15–8</td><td>IP_MNT</td><td>IP block maintenance version = 0x00</td></tr>
<tr><td>7–0</td><td>IP_CFG</td><td>IP block configuration options
<table>
<tr><th>Bits</th><th>Configuration</th></tr>
<tr><td>7–6</td><td>Classification unit blocks. Each block supports 8 rules.
00 4 blocks
01 8 blocks (default)
10 16 blocks
11 32 blocks</td></tr>
<tr><td>5–4</td><td>Inbound reassembly contexts.
00 24 reassembly contexts (default)
01 40 reassembly contexts
10 56 reassembly contexts
11 72 reassembly contexts</td></tr>
<tr><td>3</td><td>Inbound reassembly units.
0 4 reassembly units (default)
1 2 reassembly units</td></tr>
<tr><td>2</td><td>Reserved = 0x0 by default.</td></tr>
<tr><td>1–0</td><td>Outbound segmentation units.
00 4 segmentation units
01 6 segmentation units (default)
10 8 segmentation units
11 Reserved</td></tr>
</table></td></tr>
</table>

16.5 Programming Restrictions

The MSC8157E has the following programming restrictions regarding the programming of the eMSG unit:

- The message manager performs pre-fetching of descriptors when sending outbound RapidIO Type5, Type6, or Type9 transactions. This occurs as a fetch of the first scatter/gather table entry followed by the message descriptor when the format is defined as scatter/gather. The message manager allows for out-of-order pre-fetching between transactions such that a Type9 may fetch the scatter/gather table entry, followed by a Type11 fetching the message descriptor. In this case, the Type11 transaction is ready for segmentation before Type 9. In the event a scatter/gather type transaction receives the message descriptor at the same time a newer transaction is sent to an outbound segmentation unit, there is a possibility the scatter/gather transaction is lost. The loss is due to using multiple arbitration groups and mixing data formats. Limiting to a single arbitration group, however, can affect performance. If an application must mix data formats, use a single arbitration group. If only one type of data format is used, there are no other restrictions.

- The eMSG unit can transmit a duplicate message segment when the same arbitration group (typically equals the SRIO flow level) is allowed to use more than one segmentation unit and the messages are not all multicast type. For a single segment message, the duplication does not produce an error during eMSG inbound reassembly. For multi-segment messages, eMSG inbound reassembly produce a time-out error when only a single segment is received for a multi-segment message. Use one of the following two options to avoid this situation:

1. Restrict messages to single segment and enable multicast. Messages intended for only one destination must be set to a multicast of one with the correct bit set in the multicast list.
2. Restrict a given arbitration group to use only one segmentation unit. For example, Arb group 0 uses SU 0, Arb group 1 uses SU 1, and so on. There is a varying performance impact when using fewer Arb groups than the number of segmentation units. Execution privileges can be set in MMSEPR. This option also requires the following rules:
   a. A message context cannot be reused across arbitration groups.
   b. A dummy message must be sent through all segmentation units used for messages OR you must avoid the use of mailbox 0, letter 0, or destination 0.

17 PCI Express Controller

The PCI Express controller supports communication with PCI Express devices connected to the MSC8157E device. The PCI Express interface is designed to comply with the *PCI Express Base Specification, Revision 2.0* (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 MSC8157E device through the High Speed Serial Interface (HSSI) to a 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 MSC8157E 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.

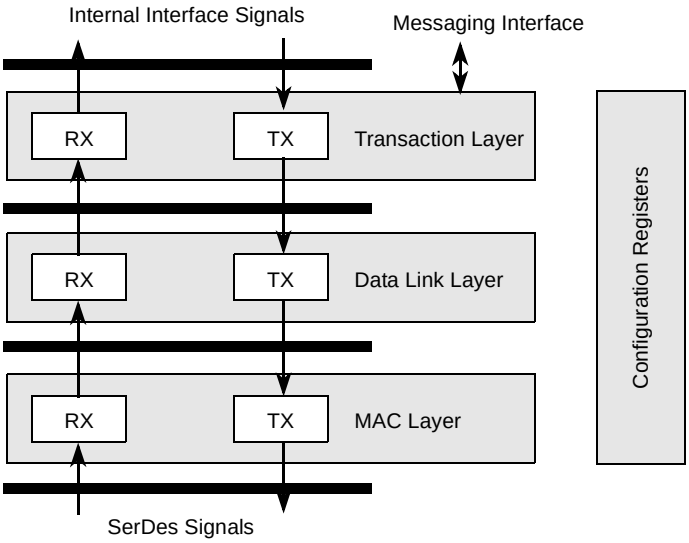


Figure 17-1. PCI Express Controller Block Diagram

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 to issue. 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.

17.1.1.1 Training to 2.5 Gbaud After Reset

The RCW bit PFREQ has no effect on the speed at which the PCI Express runs after power-on reset. The PCI Express controller runs at 5 Gbaud after the system exits reset if the external PCI Express controller is a Gen2 device and the PCI Express controller cannot train to 2.5 Gbaud after exiting power-on reset if the external device supports Gen2 speed even if value of RCW Bit PFREQ = 1 (2.5 Gbaud). To run the PCI Express controller at 2.5 Gbaud, write the specified values to the following fields in PEX_LSCR (offset 0x108):

- PEX_LSCR[LSRS] = 0b0001
- PEX_LSCR[LSM] = 0b1
- PEX_LSCR[LSR] = 0b1

This allows the PCI Express controller to train to 2.5 Gbaud.

17.1.1.2 Link Width Downtraining

When PCI Express controller is downtrained as a x2 or a x1 link from a x4 or a x2 link, the PCI Express controller may flag a correctable error: CED set in the PCI Express Device Status Register and RXE (Receiver error status) set in the PCI Express Correctable Error Status Register. Therefore, after training and before initiating any transaction towards the PCI Express interface, clear the CED field in the PCI Express Device Status Register and the RXE field in the PCI Express Correctable Error Status Register, to allow real error reports in the functional mode.

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 to issue. 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.

Note: Ensure that all inbound PCI Express read accesses are 4-byte aligned. Read errors may appear on unaligned accesses.

17.2 Features

The PCI Express controller includes the following features:

- Designed to comply with the *PCI Express Base Specification, Revision 2.0* with backward compatibility with Revision 1.1.
- 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, one 64-bit message signalled interrupt (MSI) window (RC mode only), and one configuration window that can be alternately configured as a general purpose window.
- Supports 32- or 64-bit MSIs.
- Supports four outbound translation windows and one default window.
- Supports eight non-posted and four (with Serial RapidIO transactions) or six (without Serial RapidIO transactions) 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.
- Software-controllable, dynamic link width and speed.

17.3 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.

17.4 Functional Description

The PCI Express protocol relies on a requestor/completer relationship where one device requests that some desired action be performed by some target device and the target device completes the task and responds. Usually the requests and responses occur through a network of links, but to the requestor and to the completer, the intermediate components are transparent.

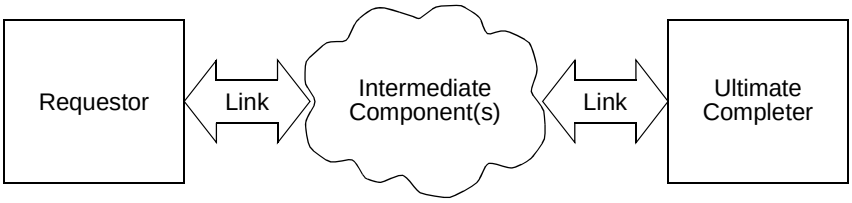


Figure 17-2. Requestor/Completer Relationship

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**.

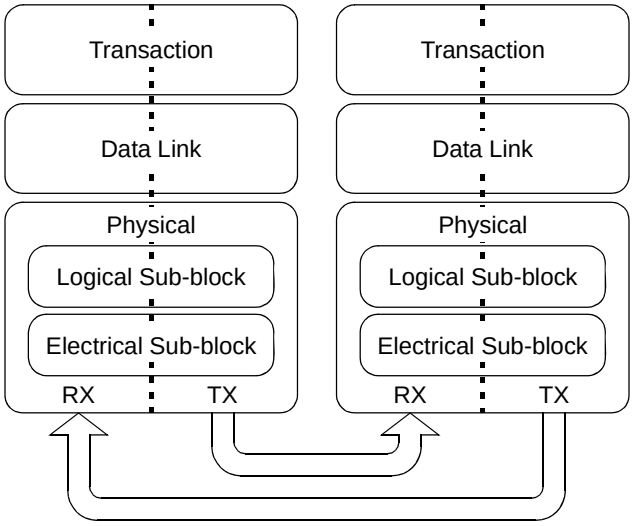


Figure 17-3. PCI Express High-Level Layering

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.

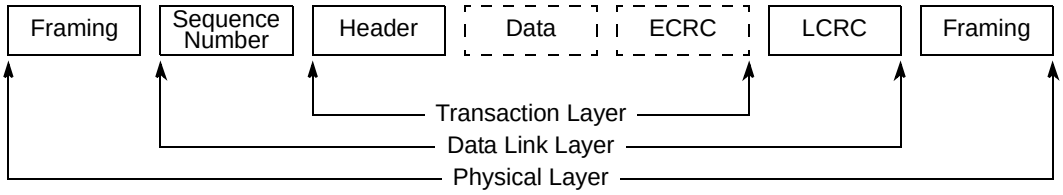


Figure 17-4. PCI Express Packet Flow

17.4.1 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

17.4.1.1 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.

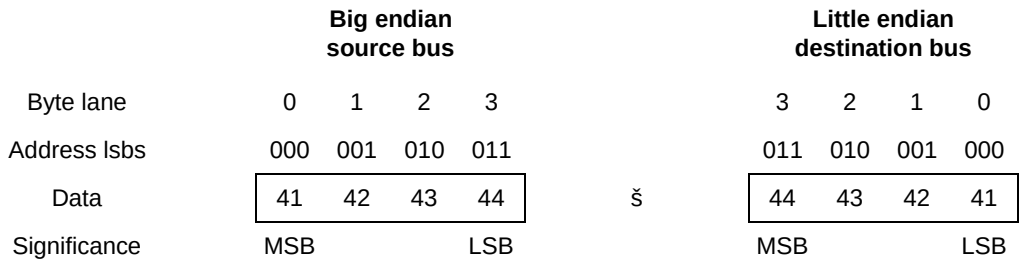


Figure 17-5. Address Invariant Byte Ordering—4 bytes Outbound

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.

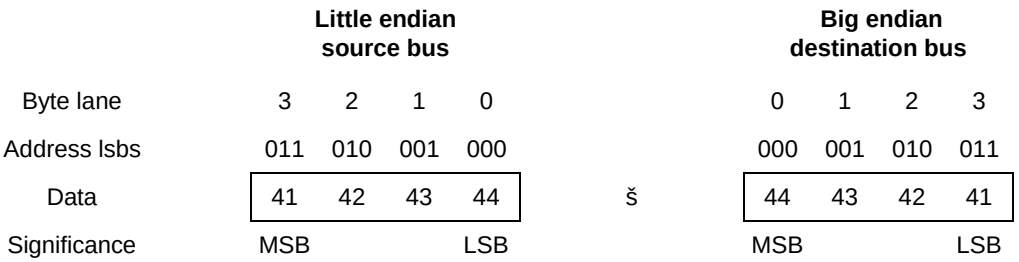


Figure 17-6. Address Invariant Byte Ordering—4 bytes Inbound

Figure 17-7 shows an outbound transfer of an 8-byte scalar, 0x5455_1617_CDCE_2728, using address invariance.

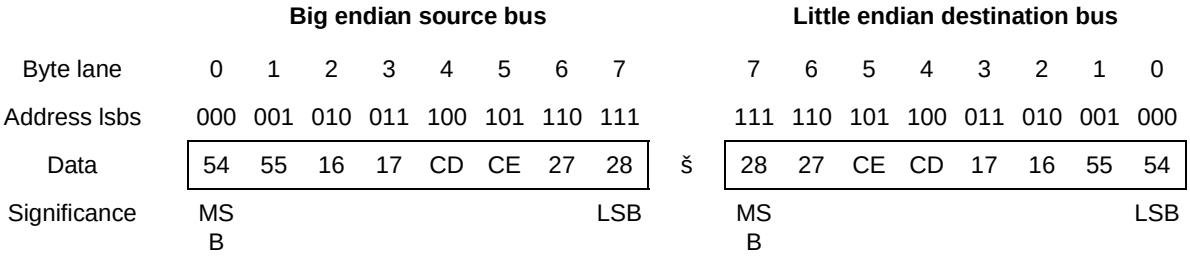


Figure 17-7. Address Invariant Byte Ordering—8 bytes Outbound

Figure 17-8 shows an inbound transfer of a 2-byte scalar, 0x5837, using address invariance.

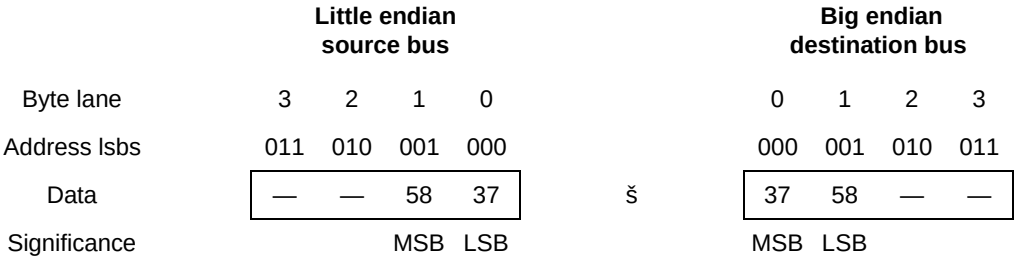


Figure 17-8. Address Invariant Byte Ordering—2 bytes Inbound

Note: In all of these examples, the original addresses of the individual bytes within the scalars (as created by the source) are preserved.

17.4.1.2 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.

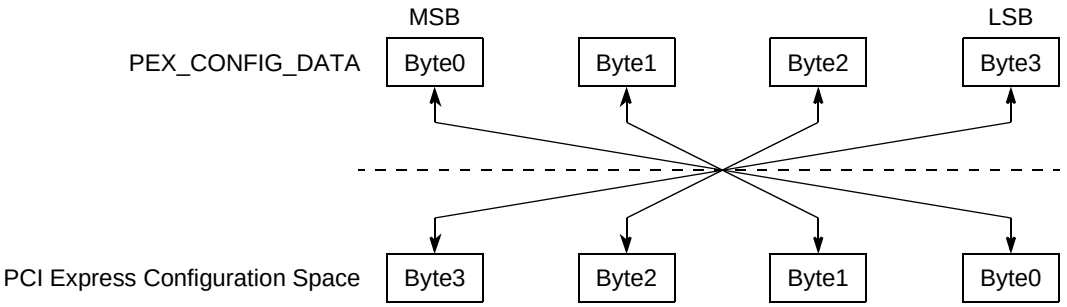


Figure 17-9. PEX_CONFIG_DATA Byte Ordering

17.4.1.3 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.

17.4.1.4 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.

17.4.1.5 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

17.4.1.6 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 17.5.1.10, *PCI Express Configuration Space Access, on page 17-71***, 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.

17.4.1.7 Serialization of Configuration and I/O Writes

Configuration and I/O writes originating 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 re-issues the configuration write transaction. It keeps retrying the request until either a status other than CRS is returned or the transaction times out.

Note: 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.

17.4.2 Messages

Software message generation is supported in both RC and EP modes.

17.4.2.1 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 17.5.1.2.4**, *PCI Express Power Management Command Register (PEX_PMCR)*, on page 17-39 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

17.4.2.2 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

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
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.

Table 17-8. PCI Express EP Inbound Message Handling (Continued)

Name	Code[7–0]	Routing[2–0]	Action
Vendor_Defined Type 0	0111 1110	—	No action taken
Vendor_Defined Type 1	0111 1111	—	No action taken

17.4.3 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.

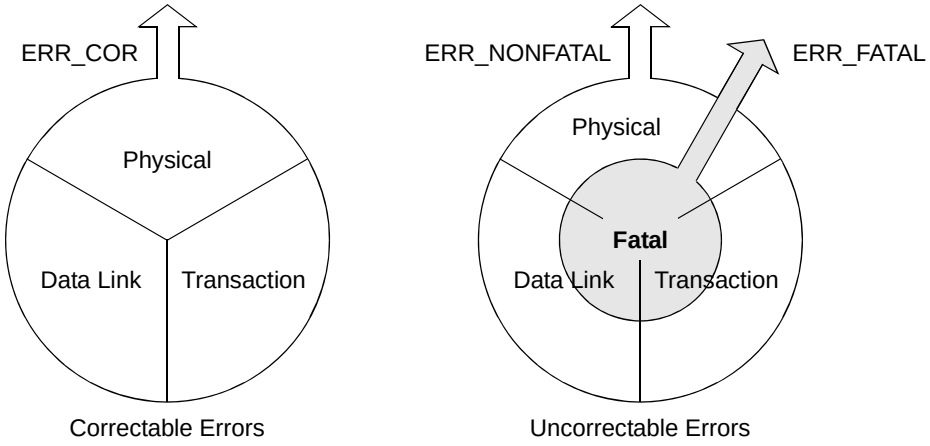


Figure 17-10. PCI Express Error Classification

17.4.3.1 PCI Express Error Logging and Signaling

Figure 17-11 shows the PCI Express-defined sequence of operations related to signalling 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.

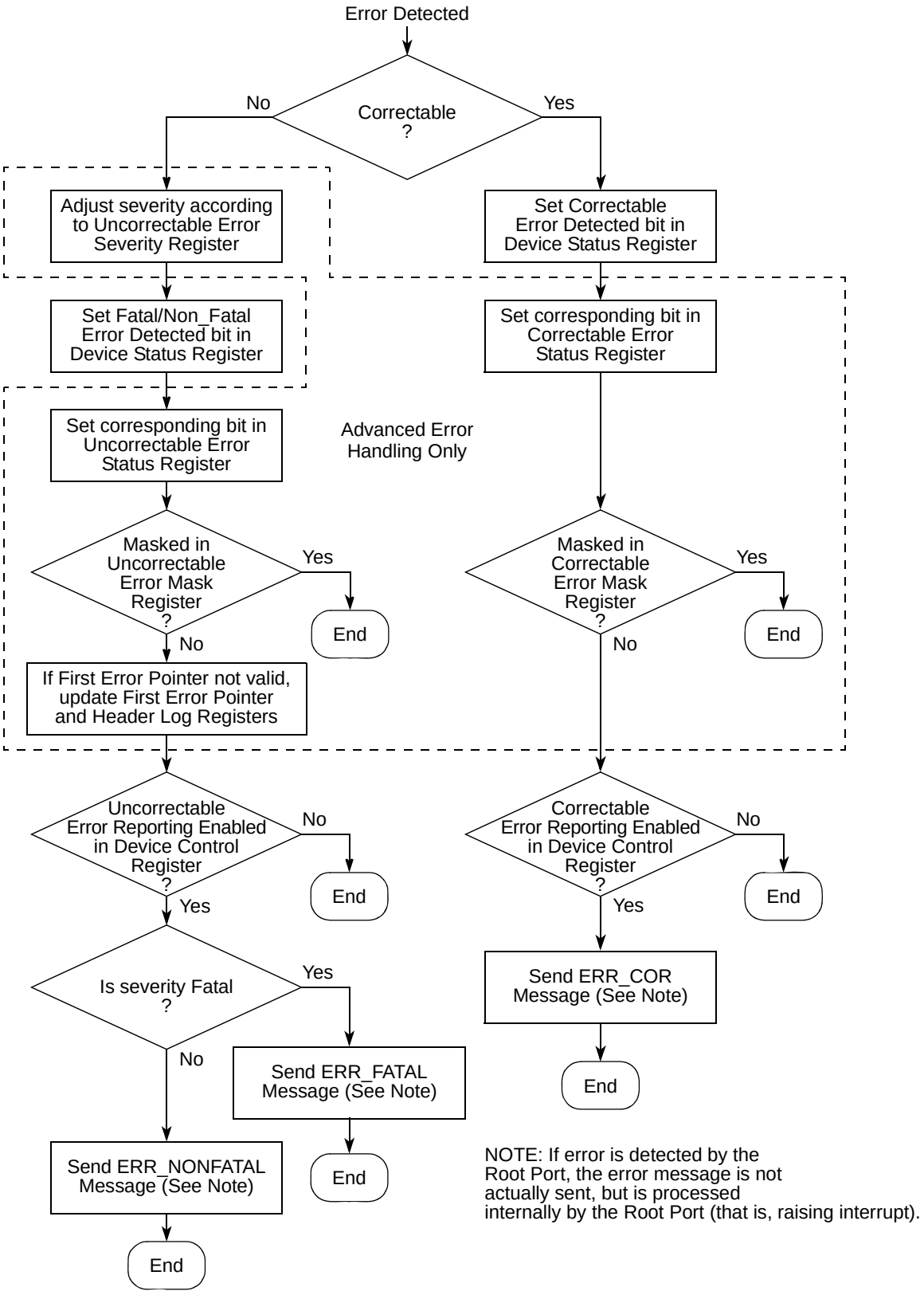


Figure 17-11. PCI Express Device Error Signaling Flowchart

17.4.3.2 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 signalling 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 signalled 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] (signalled system error) is set.	PCI Express Secondary Status Interrupt Mask Register[4] (mask signalled 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.

17.4.3.3 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.

17.4.4 Interrupts

Handling of INTx and message signalled 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 EP device, the controller can generate interrupt signals. **Table 17-11** and **Table 17-12** summarize the interrupt functionality for EP 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 signalling mechanism for PCI Express. However, in RC mode, the PCI Express controller supports the INTx virtual-wire interrupt signalling 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.

17.4.5 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

17.4.6 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.

The PCI Express controller includes support for an L0 low-power interconnection state in which the link is taken to an electrical idle condition and can be taken back to the L0 state without a full device reset. The controller and the PHY should be able to negotiate exiting from the electrical idle state without errors. However, a correctable replay timer time-out may occur and be detected after exit from electrical idle. The error is indicated by the RTTO bit in the PCI Express Correctable Error Status Register. The condition only affects the controller when configured for x2 or x4 PCI Express link widths. Use one of the following methods to prevent this error:

- Disable ASPM by configuring the PCI Express Link Control Register ASPM_CTL field to 0b00.
- The ACK Replay Timeout Register is at offset 0x438 from the CCSR base. Change the RTV (Replay Timeout Value) field (bits 27:13) to 0xFFF.

17.4.7 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**.

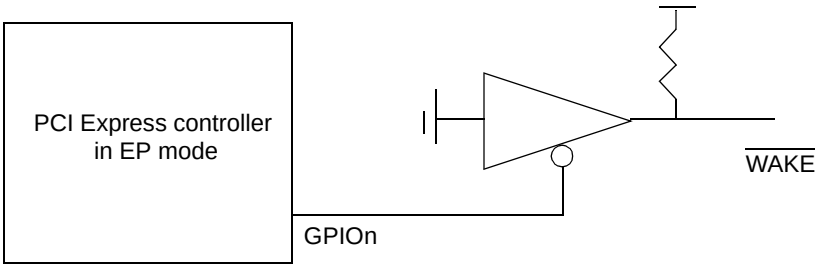


Figure 17-12. $\overline{\text{WAKE}}$ Generation Example

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.

17.4.8 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.

17.4.8.1 Hot Reset During Training and Polarity

During training, the PCI Express controller detects the proper lane polarity and records it in the CSR Link Status Register (offset 0x420). If the link partner initiates a hot reset sequence, after the sequence is finished, the detected polarity status is cleared, and the PCI Express controller is not able to complete the training if the polarity inversion is required. If link training is unsuccessful following a hot reset sequence, software should perform the following:

- Write the value 0x88400000 to address 0xFFFB7F00 to reset the PCI controller.
- Wait 1 ms.
- Write the value 0x80400000 to address 0xFFFB7F00 to allow the link training process to start from the DETECT state for proper detection of polarity inversion.

17.4.8.2 Hot Reset During EP Mode

When a hot reset event is detected while the PCI Express Controller is configured in EP mode, the following may occur:

- The TLP received right before the Hot Reset event may be discarded
- Data corruption may occur on the first inbound memory transaction received after the Hot Reset event.

Depending on the type of the first inbound memory transaction received after Hot Reset, data corruption may occur as follows:

- If it is a memory write, the transaction may finish with data corruption at the target.
- If it is a memory read, the transaction may be decoded incorrectly and the return data might be incorrect.

To prevent this occurrence, do the following:

- When possible, before issuing the Hot Reset, the upstream RC or Switch should quiesce the system first to ensure no inbound traffic is flowing into the Freescale PCI Express EP controller. This prevents the above mentioned “inbound memory write followed immediately by Hot Reset event” sequence from occurring. The exact requirement and action required to quiesce the system depends on the system, application and software used. For example, some requirements may include, but are not be limited to, using a software semaphore at the RC system side to stop the new memory requests targeting the downstream Freescale PCI Express EP controller from the RC system's software API layer or DMA controller. Once such “quiescing system” actions have been finished, the RC system can send a configuration write cycle to clear the Memory Space bit in the Freescale PCI Express EP controller Command Register, followed by a several microsecond delay to allow all previously received inbound memory writes to propagate through the EP controller. A Hot Reset command can then be applied.
- If an “inbound memory write followed immediately by Hot Reset event” sequence cannot be avoided, do the following: immediately after the Hot Reset event, the upstream RC can issue a dummy memory write followed by another write or read to the read-only PEX_IP_BLK_REV1 memory-mapped register in the downstream Freescale device with PCI Express controller configured in EP Mode. The RC can then re-transfer the last memory write TLP that occurred right before the Hot Reset event and resume other normal traffic.

17.4.9 Link Down

Typically, a link down condition occurs after a hot reset event; however, it is possible for the link to go down unexpectedly without a hot reset event. When this occurs, a link down condition is detected (PEX_PME_MSG_DR[LDD]=1). Link down is treated similarly to a hot reset condition.

Subsequently, while the link is down, all new posted outbound transactions are discarded. All new non-posted ATMU transactions are errored out. Non-posted configuration transactions issued using PEX_CONFIG_ADDR/PEX_CONFIG_DATA toward the link returns 0xFFFF_FFFF (all 1s). As soon as the link is up again, the sending of transaction resumes.

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.

17.4.10 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 17.5.1.13.18**, *Configuration Ready Register—0x4B0*, on page 17-127 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.

17.4.11 Enabling Automatic Retraining of Link

As part of the initialization, software should clear PEX_FC_UPDATE_TOR[DIS_LR] to allow automatic retraining of the link if the link partner is down or the device has not received DLLP updates within the interval specified in PEX_FC_UPDATE_TOR[FC_UPDATE_TIMEOUT].

17.4.12 Spread Spectrum Reference Clock Feature

Spread spectrum is supported by PCI Express. However, spread spectrum for the PCI Express cannot use the PLL that serves the SRIO or SGMII interfaces when they are not turned off. Moreover, the PCI Express lane can accept a spread spectrum input even if its own reference clock (REF_CLK) is not spread; that is, you can connect two devices, one with a spread spectrum reference clock and the other without a spread spectrum reference clock, and they can communicate without failures.

17.5 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 17.5.1**, *PCI Express Memory Mapped Registers*, on page 17-27, 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 17.5.1.10**, *PCI Express Configuration Space Access* and its subsections.

17.5.1 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 base address 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 17-30**
0x004	PEX_CONFIG_DATA—PCI Express configuration data register	R/W	0x0000_0000	**page 17-31**
0x00C	PEX_OTB_CPL_TOR—PCI Express outbound completion timeout register	R/W	0x0013_FFFF	**page 17-32**
0x010	PEX_CONF_RTY_TOR—PCI Express configuration retry timeout register	R/W	0x0400_FFFF	**page 17-33**
0x014	PEX_CONFIG—PCI Express configuration register	R/W	0x0004_0028	**page 17-34**
0x018	PEX_INT_STAT—PCI Express interrupt status register	R	0x0000_0000	**page 17-35**
PCI Express Power Management Event & Message Registers				
0x020	PEX_PME_MES_DR—PCI Express PME & message detect register	w1c	0x0000_0000	**page 17-36**
0x024	PEX_PME_MES_DISR—PCI Express PME & message disable register	R/W	0x0000_0000	**page 17-37**
0x028	PEX_PME_MES_IER—PCI Express PME & message interrupt enable register	R/W	0x0000_0000	**page 17-38**
0x02C	PEX_PMCR—PCI Express power management command register	R/W	0x0000_0000	**page 17-39**
0x100	PEX_LWCR—PCI Express link width control register	R/W	0x0000_1000	**page 17-39**
0x104	PEX_LWSR—PCI Express link width status register	R	0x0000_0000	**page 17-40**
0x108	PEX_LSCR—PCI Express link speed control register	R/W	0x0000_2000	**page 17-41**
0x10C	PEX_LSSR—PCI Express link speed status register	R	0x0000_000B	**page 17-42**
PCI Express IP Block Revision Registers				
0xBF8	IP block revision register 1 (PEX_IP_BLK_REV1)	R	0x0208_0202	**page 17-43**
0xBFC	IP block revision register 2 (PEX_IP_BLK_REV2)	R	0x0000_0000	**page 17-43**

Table 17-15. PCI Express Memory-Mapped Register Map (Continued)

Offset	Register	Access	Reset	Page
PCI Express ATMU Registers				
Outbound Window 0 (Default)				
0xC00	PEXOTAR0—PCI Express outbound translation address register 0 (default)	R/W	0x0000_0000	**page 17-45**
0xC04	PEXOTEAR0—PCI Express outbound translation extended address register 0 (default)	R/W	0x0000_0000	**page 17-46**
0xC10	PEXOWAR0—PCI Express outbound window attributes register 0 (default)	Mixed	0x8004_4023	**page 17-48**
Outbound Window 1				
0xC20	PEXOTAR1—PCI Express outbound translation address register 1	R/W	0x0000_0000	**page 17-45**
0xC24	PEXOTEAR1—PCI Express outbound translation extended address register 1	R/W	0x0000_0000	**page 17-46**
0xC28	PEXOWBAR1—PCI Express outbound window base address register 1	R/W	0x0000_0000	**page 17-47**
0xC30	PEXOWAR1—PCI Express outbound window attributes register 1	R/W	0x0004_4023	**page 17-48**
Outbound Window 2				
0xC40	PEXOTAR2—PCI Express outbound translation address register 2	R/W	0x0000_0000	**page 17-45**
0xC44	PEXOTEAR2—PCI Express outbound translation extended address register 2	R/W	0x0000_0000	**page 17-46**
0xC48	PEXOWBAR2—PCI Express outbound window base address register 2	R/W	0x0000_0000	**page 17-47**
0xC50	PEXOWAR2—PCI Express outbound window attributes register 2	R/W	0x0004_4023	**page 17-48**
Outbound Window 3				
0xC60	PEXOTAR3—PCI Express outbound translation address register 3	R/W	0x0000_0000	**page 17-45**
0xC64	PEXOTEAR3—PCI Express outbound translation extended address register 3	R/W	0x0000_0000	**page 17-46**
0xC68	PEXOWBAR3—PCI Express outbound window base address register 3	R/W	0x0000_0000	**page 17-47**
0xC70	PEXOWAR3—PCI Express outbound window attributes register 3	R/W	0x0000_0000	**page 17-48**
Outbound Window 4				
0xC80	PEXOTAR4—PCI Express outbound translation address register 4	R/W	0x0000_0000	**page 17-45**
0xC84	PEXOTEAR4—PCI Express outbound translation extended address register 4	R/W	0x0000_0000	**page 17-46**
0xC88	PEXOWBAR4—PCI Express outbound window base address register 4	R/W	0x0000_0000	**page 17-47**
0xC90	PEXOWAR4—PCI Express outbound window attributes register 4	R/W	0x0004_4023	**page 17-48**
Inbound MSI Window				
0xD00	PEXMSIITAR—PCI Express MSI inbound translation address register	R	0x000f_ff00	**page 17-54**
0xD08	PEXMSIIWBAR—PCI Express MSI inbound window base address register	R/W	0x0000_0000	**page 17-55**
0xD0C	PEXMSIIWBEAR—PCI Express MSI inbound window base extended address register	R/W	0x0000_0000	**page 17-55**
0xD10	PEXMSIIWAR—PCI Express MSI inbound window attributes register	R/W	0x00F4_4013	**page 17-56**
Inbound Window 3				
0xD80	PEXITAR3—PCI Express inbound translation address register 3	R/W	0x000F_FF00	**page 17-51**
0xD88	PEXIWBAR3—PCI Express inbound window base address register 3	R/W	0x0000_0000	**page 17-51**
0xD8C	PEXIWBEAR3—PCI Express inbound window base extended address register 3	R/W	0x0000_0000	**page 17-52**
0xD90	PEXIWAR3—PCI Express inbound window attributes register 3	R/W	0x00F4_4023	**page 17-53**
Inbound Window 2				

Table 17-15. PCI Express Memory-Mapped Register Map (Continued)

Offset	Register	Access	Reset	Page
0xDA0	PEXITAR2—PCI Express inbound translation address register 2	R/W	0x000F_FF00	**page 17-51**
0xDA8	PEXIWBAR2—PCI Express inbound window base address register 2	R/W	0x0000_0000	**page 17-51**
0xDAC	PEXIWBEAR2—PCI Express inbound window base extended address register 2	R/W	0x0000_0000	**page 17-52**
0xDB0	PEXIWAR2—PCI Express inbound window attributes register 2	R/W	0x00F4_4023	**page 17-53**
Inbound Window 1				
0xDC0	PEXITAR1—PCI Express inbound translation address register 1	R/W	0x000F_FF00	**page 17-51**
0xDC8	PEXIWBAR1—PCI Express inbound window base address register 1	R/W	0x0000_0000	**page 17-51**
0xDD0	PEXIWAR1—PCI Express inbound window attributes register 1	R/W	0x00F4_4023	**page 17-53**
Inbound Window 0				
0xDE0	PEXITAR0—PCI Express inbound translation address register 1	R/W	0x000F_FF00	**page 17-51**
0xDE8	PEXIWBAR0—PCI Express inbound window base address register 1	R/W	0x0000_0000	**page 17-51**
0xDF0	PEXIWAR0—PCI Express inbound window attributes register 1	R/W	0x80F4_4013	**page 17-53**
PCI Express Error Management Registers				
0xE00	PEX_ERR_DR—PCI Express error detect register	w1c	0x0000_0000	**page 17-58**
0xE08	PEX_ERR_EN—PCI Express error interrupt enable register	R/W	0x0000_0000	**page 17-61**
0xE10	PEX_ERR_DISR—PCI Express error disable register	R/W	0x0000_0000	**page 17-63**
0xE20	PEX_ERR_CAP_STAT—PCI Express error capture status register	Mixed	0x0000_0000	**page 17-65**
0xE28	PEX_ERR_CAP_R0—PCI Express error capture register 0	R/W	0x0000_0000	**page 17-66**
0xE2C	PEX_ERR_CAP_R1—PCI Express error capture register 1	R/W	0x0000_0000	**page 17-67**
0xE30	PEX_ERR_CAP_R2—PCI Express error capture register 2	R/W	0x0000_0000	**page 17-69**
0xE34	PEX_ERR_CAP_R3—PCI Express error capture register 3	R/W	0x0000_0000	**page 17-70**

17.5.1.1 PCI Express Configuration Access Registers

17.5.1.1.1 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 17-16**.

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).	
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.	

17.5.1.1.2 PCI Express Configuration Data Register (PEX_CONFIG_DATA)

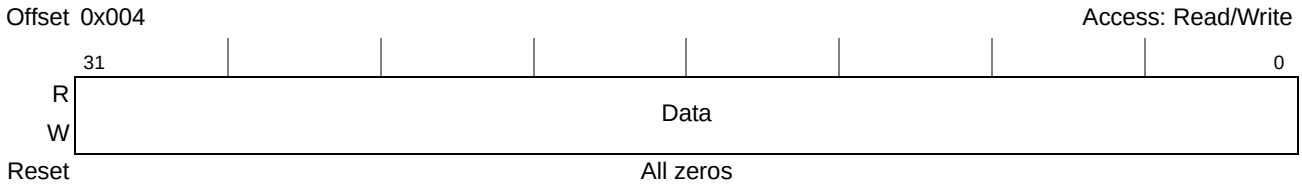


Figure 17-13. PCI Express Configuration Data Register (PEX_CONFIG_DATA)

The PCI Express configuration data register, show in **Figure 17-13**, 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 17.4.1.2**, *Byte Order for Configuration Transactions*, on page 17-10 for more information. The fields of the PCI Express configuration data register are described in **Table 17-17**.

Table 17-17. 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).

17.5.1.1.3 PCI Express Outbound Completion Timeout Register (PEX_OTB_CPL_TOR)

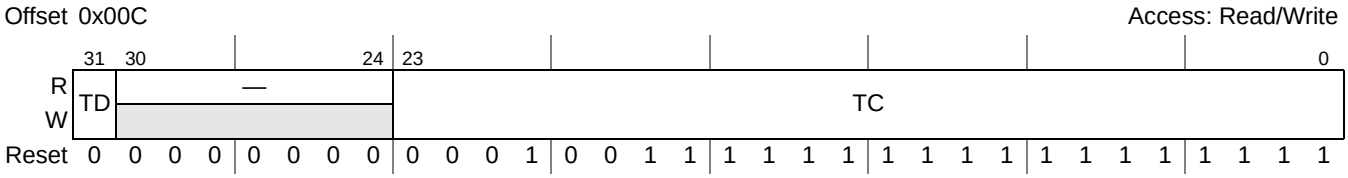


Figure 17-14. PCI Express Outbound Completion Timeout Register (PEX_OTB_CPL_TOR)

The PCI Express outbound completion timeout register, shown in **Figure 17-14**, contains the maximum wait time for a response to come back as a result of an outbound non-posted request before a timeout condition occurs. This register is also used for timeout requests that were not acknowledged (PEX_ERR_DET[PAT] which is a unrecoverable error. The outbound completion timeout register 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. The fields of the PCI Express outbound completion timeout register are described in **Table 17-18**.

Table 17-18. 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. The completion timeout is determined by TC and the setting of the completion timeout disable bit (CPL_TOD) in the Device Control 2 Register. **Note:** The maximum computed timeout value based on TC cannot exceed 1.072 s at 500 MHz. When Device_Control _2[CPL_TOD] is set, the timeout the TC unit is 8 x the PCI Express controller clock period; that is, one TC unit is 16 ns at 500 MHz. The following are examples of timeout periods based on 500 MHz: • 0x000_0000 Reserved • 0x010_FFFF 17.82 ms at 500 MHz controller clock. • 0xFFF_FFFF 268.43 ms at 500 MHz controller clock When Device Control 2[CPL_TOD] is cleared, the timeout value is based on TC and Device Control 2[CPL_TO_VAL]. When completion timeout value (CPL_TO_VAL) is: • 0000 the timeout value is TC • 0001 the timeout value is TC/64 • 0010 the timeout value is TC/4 • 0101 the timeout value is TC • 0110 the timeout value is TC×4 • 1001 the timeout value is TC×16 • 1010 the timeout value is TC×64 For all other values of completion timeout the timeout value is TC.

17.5.1.1.4 PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR)

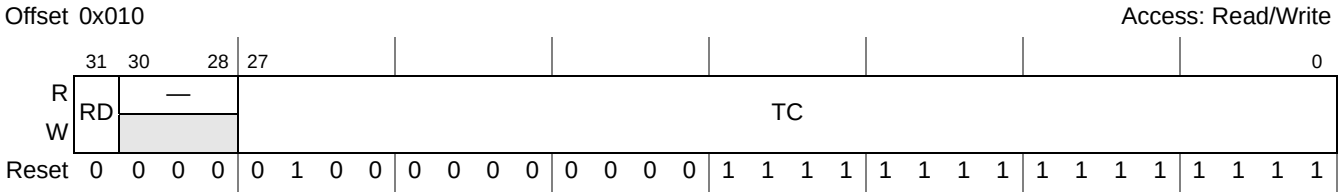


Figure 17-15. PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR)

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. This register is a programmable register that represents 1/8 timeout period of a configuration request is scheduled to PCI Express and a CRS response is received. While waiting for the response to come back, the counter is decremented. If a CRS response continues to be received when the timer expires, then a timeout error occurs. The logic continues to issue the same configuration request with a CRS status completion until a status other than CRS is returned or it has exceed the Configuration Timeout period. A minimum of 2^{26} PCI Express clocks is required before timeout. The fields of the PCI Express configuration retry timeout register are described in **Table 17-19**.

Table 17-19. 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 response counter. The value is 1/8 of the total timeout value. One TC unit is 8 times the PCI Express controller clock, that is, one TC unit is 16 ns at 500 MHz. The following are examples of timeout periods based on 500 MHz: • 0x000_0000 Reserved • 0x010_FFFF 1.07 s at 500 MHz controller clock. • 0xFFF_FFFF 4.30 s at 500 MHz controller clock

17.5.1.1.5 PCI Express Configuration Register (PEX_CONFIG)

Offset 0x014 Access: Read/Write

	31				19	18	16	15	14	13	12				5	4	3	2	1	0
R W	—									OB_CK	—					SAC	—		SP	SCC
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0

Figure 17-16. PCI Express Configuration Register (PEX_CONFIG)

The PCI Express configuration register, shown in **Figure 17-4**, contains various control switches for the controller. The fields of the PCI Express configuration register are described in **Table 17-20**.

Table 17-20. 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 17.5.1.12.11**, *PCI Express Link Control Register—0x5C,* on page 17-103 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 17.5.1.12.6**, *PCI Express Capabilities Register—0x4E,* on page 17-99 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 17.5.1.12.12**, *PCI Express Link Status Register—0x5E,* on page 17-103 for more information.

17.5.1.1.6 PCI Express Interrupt Status Register (PEX_INT_STAT)

Offset 0x018 Access: Read only

	31															16
R	—															
W																
Reset	All zeros															

	15	14	13	8	7	6	5	4	0
R	INTM	INTE	—		INTG0	INTG1	INTG2	—	
W									
Reset	All zeros								

Figure 17-17. PCI Express Interrupt Source Register (PEX_INT_STAT)

The PCI Express interrupt status register, shown in **Figure 17-17**, shows the current interrupt sources that are raising an interrupt to the core. This register is a read only register. Clearing the interrupt source will clear the corresponding bit in the interrupt status register.The fields of the PCI Express PME and message detect register are described in **Table 17-21**.

Table 17-21. PEX_INT_STAT Descriptions

Bits	Name	Description
31–16	—	Reserved
15	INTM	Interrupt from PEX_PME_MES_DR register. 1 Interrupt is pending 0 No interrupt
14	INTE	Interrupt from PEX_ERR_DR register. 1 Interrupt is pending 0 No interrupt
13–8	—	Reserved
7	INTG0	Interrupt from group 0. This includes interrupt generated by Root Status Register [PME Status]. 1 Interrupt is pending 0 No interrupt
6	INTG1	Interrupt from group 1. This includes interrupts generated from receiving any error messages (Fatal, Non-fatal, Correctable). 1 Interrupt is pending 0 No interrupt
5	INTG2	Interrupt from group 2. This includes interrupts generated from command status register (Master Abort, Target Abort, and so forth) 1 Interrupt is pending 0 No interrupt
4–0	—	Reserved

17.5.1.2 PCI Express Power Management Event and Message Registers

17.5.1.2.1 PCI Express PME and Message Detect Register (PEX_PME_MES_DR)

Offset 0x020 Access: w1c

	31–16	15	14	13	12	11	10	9	8	7	6–4	3–0
R	—	PTO	—	ENL23	EXL23	—	HRD	LDD	AILDD	LUD	—	
W		w1c		w1c	w1c		w1c	w1c	w1c	w1c		
Reset	All zeros											

Figure 17-18. 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.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	—	Reserved. Note that during normal operation, this bit may be set (falsely). The bit may be ignored and cleared (w1c) without consequence.
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
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	AILDD	Ack-timeout induced link down detected. This bit indicates that a link down condition has been detected due to an internal timeout condition. The controller is reset and then cleans up all outstanding transactions. Link retraining takes place once the controller has cleaned itself up. No LDD will be set if this bit is set. 1 Ack timeout link down has been detected 0 Ack timeout link down not detected
7	LUD	Link up detected. This bit indicates that a link up condition has been detected. It means that link training has been successful. 1 Link up detected 0 Link up not detected

Table 17-22. PEX_PME_MES_DR Field Descriptions (Continued)

Bits	Name	Description
6–0	—	Reserved

17.5.1.2.2 PCI Express PME and Message Disable Register (PEX_PME_MES_DISR)

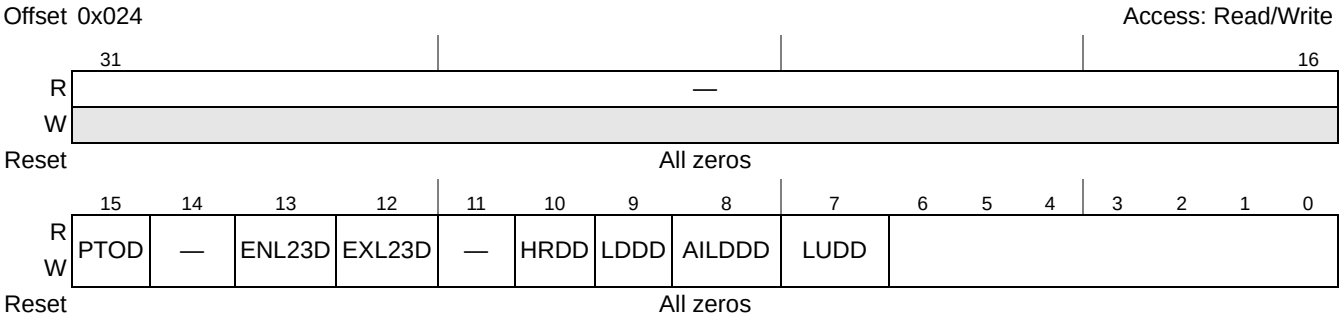


Figure 17-19. PCI Express PME and Message Disable Register (PEX_PME_MES_DISR)

The PCI Express PME and message disable register, shown in Figure 17-19, when set, prevents the detection of the corresponding bits in the PCI Express PME and message detect register. 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	—	Reserved
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	AILDDD	Ack time-out induced link down detected disable. When set, this bit disables the setting of PEX_PME_MES_DR[AILDD] bit. 1 Disable ack time-out link down state detection 0 Enable ack time-out link down state detection
7	LUDD	Link up detected disable. When set, this bit disables the setting of PEX_PME_MES_DR[LUD] bit. 1 Disable link up detection 0 Enable link up detection
6–0	—	Reserved

17.5.1.2.3 PCI Express PME and Message Interrupt Enable Register (PEX_PME_MES_IER)

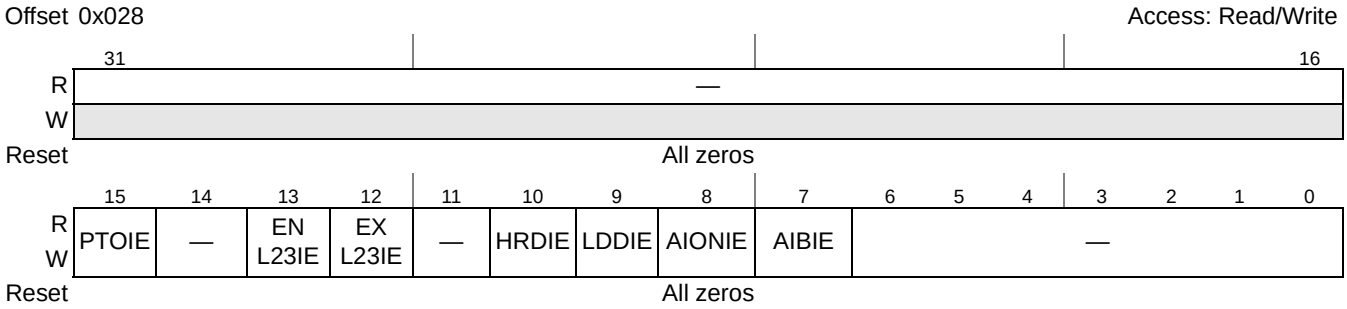


Figure 17-20. PCI Express PME and Message Interrupt Enable Register (PEX_PME_MES_IER)

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. **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	—	Reserved
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
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	AILDDIE	Ack time-out induced link down detected interrupt enable. When set and PEX_PME_MES_DR[AILDD]=1 generates an interrupt. 1 Enable ack time-out link down state interrupt generation 0 Disable ack time-out link down state interrupt generation
7	LUDIE	Link up detected interrupt enable. When set and PEX_PME_MES_DR[LUD]=1 generates an interrupt. 1 Enable link up detected interrupt generation 0 Disable link up detected interrupt generation
6–0	—	Reserved

17.5.1.2.4 PCI Express Power Management Command Register (PEX_PMCR)

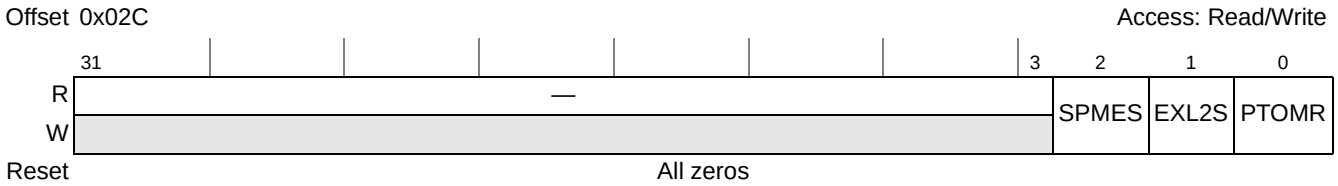


Figure 17-21. PCI Express Power Management Command Register (PEX_PMCR)

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. 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 sets the PME status bit and if PME is enabled (see **Section 17.5.1.12.3**, *PCI Express Power Management Status and Control Register—0x48*, on page 17-98, for more information) it transmits a PM_PME message upstream. This bit should not be used when in RC mode. This bit is self-clearing.
1	EXL2S	Exit L2 state. When set exits the link state out of L2/L3 ready state in order to send new requests. The request is only made when entered_L2/L3 ready state is active. This bit is self-clearing. When the link has exited L2/L3 ready state, the status bit Exit_L2/L3 ready state is set. This bit should not be used when in EP mode.
0	PTOMR	PME_Turn_Off message request. When set broadcasts a PME turn_off message. This bit should not be used when in EP mode. This bit is self-clearing

17.5.1.3 PCI Express Link Width Control Register (PEX_LWCR)

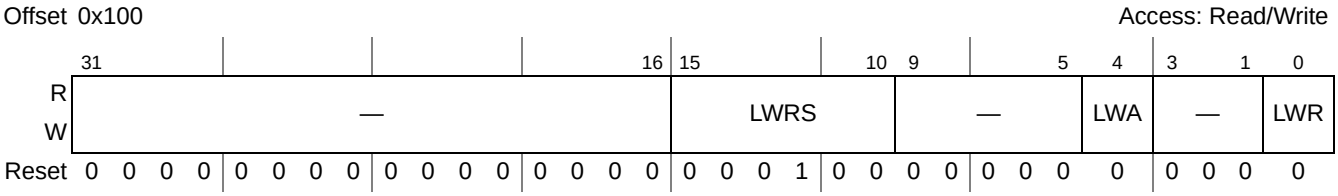


Figure 17-22. PCI Express Link Width Control Register (PEX_LWCR)

The PCI Express link width control register provides software a mechanism to dynamically changing the link width.

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 001000 - x8 link All other encodings are reserved and should not be used.

Table 17-26. PEX_LWCR Field Descriptions (Continued)

Bits	Name	Description
9–5	—	Reserved
4	LWA	Link width auto. This bit is settable 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.

17.5.1.4 PCI Express Link Width Status Register (PEX_LWSR)

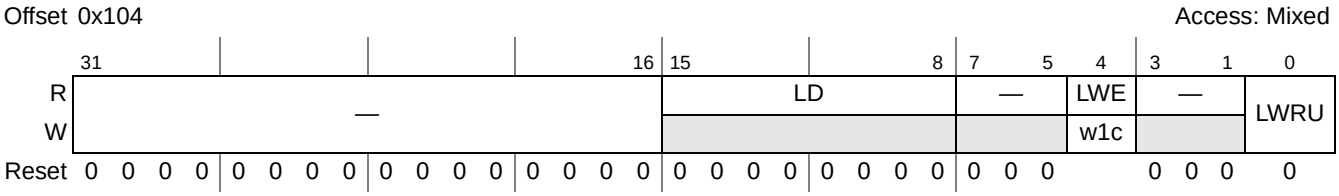


Figure 17-23. PCI Express Link Width Status Register (PEX_LWSR)

The PCI Express link width status register provides software a mechanism to dynamically changing the link width.

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 upconfig 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.

17.5.1.5 PCI Express Link Speed Control Register (PEX_LSCR)

Offset 0x108 Access: Read/Write

	31–16	15–12	11–5	4	3–2	1	0
R W	—	LSRS	—	LSA	—	LSM	LSR
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 1 0	0 0 0 0 0 0 0	0	0 0	0	0

Figure 17-24. PCI Express Link Speed Control Register (PEX_LSCR)

The PCI Express link speed control register provides software a mechanism to dynamically changing the link speed.

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. Its default value is set by configuration pins. Software should first determine the link partner's advertised 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 settable by software. It is gated with the Hardware Autonomous Speed Disable bit.
3-2	—	Reserved
1	LSM	Link speed mask. This bit when set indicate that 5.0 Gb/s speed is not supported. Its default value is set by configuration signals. If the controller is advertising that 5.0 Gb/s is supported and its link partner is also supporting 5.0Gb/s, then it is not possible to train down to 2.5 Gb/s unless this bit is set.
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.

Note: The RCW bit PFREQ has no effect on the speed at which the PCI Express runs after power-on reset. The PCI Express controller runs at 5 Gbaud after the system exits reset if the external PCI Express controller is a Gen2 device and the PCI Express controller cannot train to 2.5 Gbaud after exiting power-on reset if the external device supports Gen2 speed even if value of RCW Bit PFREQ = 1 (2.5 Gbaud). To run the PCI Express controller at 2.5 Gbaud, write the specified values to the following fields in PEX_LSCR (offset 0x108):

- PEX_LSCR[LSRS] = 0b0001
- PEX_LSCR[LSM] = 0b1
- PEX_LSCR[LSR] = 0b1

This allows the PCI Express controller to train to 2.5 Gbaud.

17.5.1.6 PCI Express Link Speed Status Register (PEX_LSSR)

Offset 0x10C — Access: Read only

	31 … 5	4	3	2	1	0
R	—	LSE	DAS		LPAS	
W		w1c				
Reset	All zeros	0	1	0	1	1

Figure 17-25. PCI Express Link Speed Status Register (PEX_LSSR)

The PCI Express link speed request status register provides software status of the link speed request.

Table 17-29. PEX_LSSR Field Descriptions

Bits	Name	Description
31–5	—	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_LSCR[LSRS] = 4'b0010 and PEX_LSSR[LPAS] = 2'b01 • PEX_PMS_MES_DR[LSD]=1 and current speed != PEX_LSCR[LSRS]
3–2	DAS	Device advertised speed. Its default value is set by configuration signals, 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 Reserved 11 5.0 Gb/s

17.5.1.7 PCI Express IP Block Revision Registers

17.5.1.7.1 IP Block Revision Register 1 (PEX_IP_BLK_REV1)

Offset 0xBF8 Access: Read only

	31															16	15							8	7							0
R	IP_ID																IP_MJ								IP_MN							
W																																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	

Figure 17-26. IP Block Revision Register 1

The IP block revision register 1 is shown in **Figure 17-9**. **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

17.5.1.7.2 IP Block Revision Register 2 (PEX_IP_BLK_REV2)

Offset 0xBFC Access: Read only

	31							24	23							16	15							8	7							0
R	—								IP_INT								—								IP_CFG							
W																																
Reset	All zeros																															

Figure 17-27. IP Block Revision Register 2

The IP block revision register 2 is shown in **Figure 17-12**. **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

17.5.1.8 PCI Express ATMU Registers

17.5.1.8.1 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, when PEX_CONFIG[OB_CK] is set, the translated address of an outbound transaction must fall into either the memory base/limit range, the prefetchable memory base/limit range, or I/O base/limit range. The transaction will be terminated if the translated address does not fall within these ranges.

Figure 17-28 shows the outbound transaction flow.

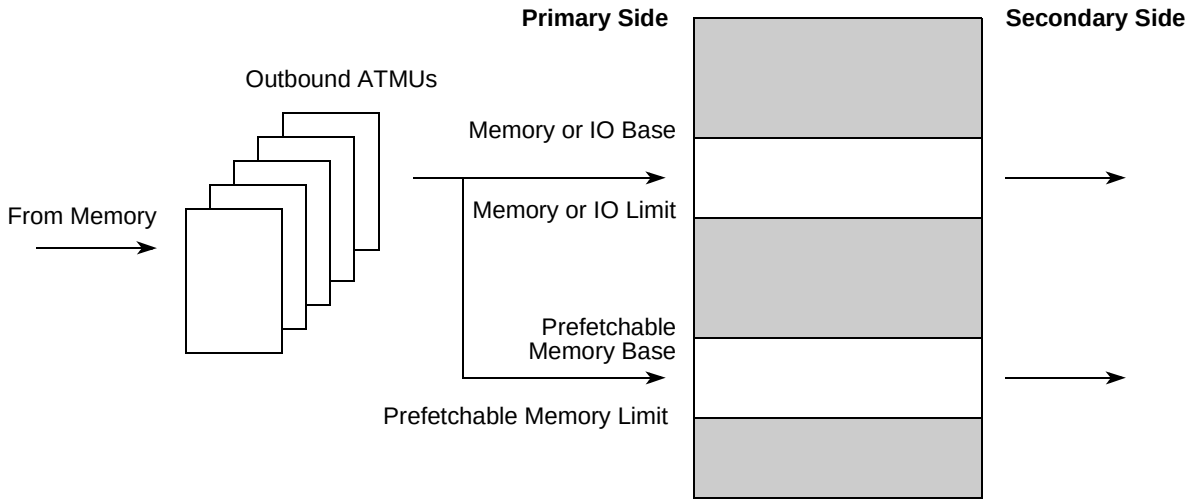


Figure 17-28. RC Outbound Transaction Flow

17.5.1.8.2 PCI Express Outbound Translation Address Registers (PEXOTAR*n*)

Offset Window 0: 0xC00
Window 1: 0xC20
Window 2: 0xC40
Window 3: 0xC60
Window 4: 0xC80

Access: Read/Write

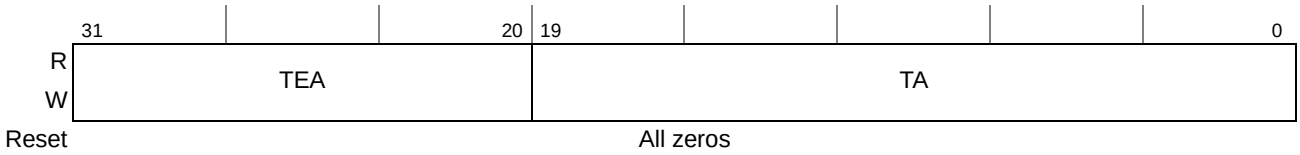


Figure 17-29. PCI Express Outbound Translation Address Registers (PEXOTAR*n*)

The PCI Express outbound translation address registers, shown in **Figure 17-29**, select the starting addresses in the system address space for window hits within the PCI Express outbound address translation windows. The new translated address is created by concatenating the transaction offset to this translation address. **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].

17.5.1.8.3 PCI Express Outbound Translation Extended Address Registers (PEXOTEAR*n*)

Offset Window 0: 0xC04
Window 1: 0xC24
Window 2: 0xC44
Window 3: 0xC64
Window 4: 0xC84

Access: Read/Write

	31	20	19	0
R W	—		TEA	
Reset	All zeros			

Figure 17-30. PCI Express Outbound Translation Extended Address Registers (PEXOTEAR*n*)

The PCI Express outbound translation extended address registers, shown in **Figure 17-30**, contain the most-significant bits of a 64 bit translation address. **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].

17.5.1.8.4 PCI Express Outbound Window Base Address Registers (PEXOWBAR*n*)

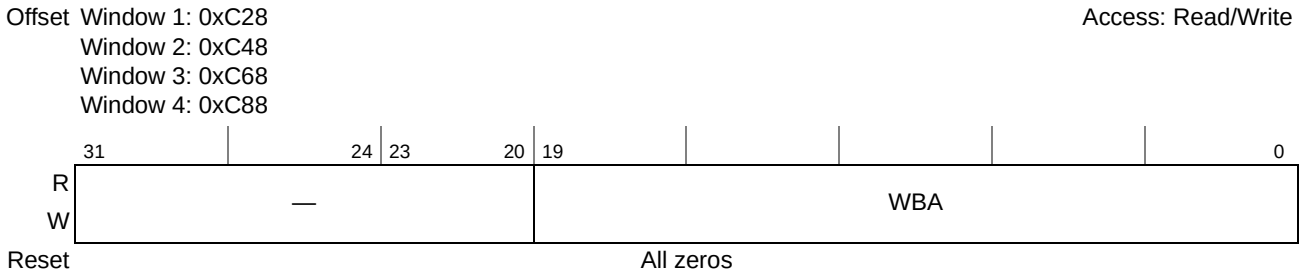


Figure 17-31. PCI Express Outbound Window Base Address Registers (PEXOWBAR*n*)

The PCI Express outbound window base address registers, shown in **Figure 17-31**, select the base address for the windows which are translated to the external address space. Addresses for outbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces the transaction is forwarded through a default register set. **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
31–20	—	Reserved
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 [31:12].

17.5.1.8.5 PCI Express Outbound Window Attributes Registers (PEXOWAR*n*)

Offset 0xC10 Access: Mixed

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EN			ROE	NS	—			TC			—	RTT				WTT				—						OWS					
W																																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1

Figure 17-32. PCI Express Outbound Window Attributes Register 0 (PEXOWAR0)

The PCI Express outbound window attributes registers, shown in **Figure 17-32** and **Figure 17-33**, define the window sizes to translate and other attributes for the translations. 64 Gbytes is the largest window size allowed. **Figure 17-32** shows the outbound window attributes register 0 (PEXOWAR0). **Figure 17-33** shows the PCI Express outbound window attributes registers 1–4 (PEXOWAR*n*).

Offset Window 1: 0xC30
Window 2: 0xC50
Window 3: 0xC70
Window 4: 0xC90
Access: Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EN			ROE	NS				TC				RTT				WTT				—						OWS					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1
Win3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-33. 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

17.5.1.8.6 PCI Express Inbound ATMU Registers

There are differences between RC and EP implementations of inbound ATMU registers as described in the following sections.

17.5.1.8.7 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 window can be used as either a fixed Mbyte window used for inbound memory transactions that access memory-mapped registers or it can be programmed as a general purpose window for local memory access.
- Inbound window BAR1 at configuration address 0x14 (32-bit)
- 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.

17.5.1.8.8 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 17-34 shows the inbound transaction flow in RC mode.

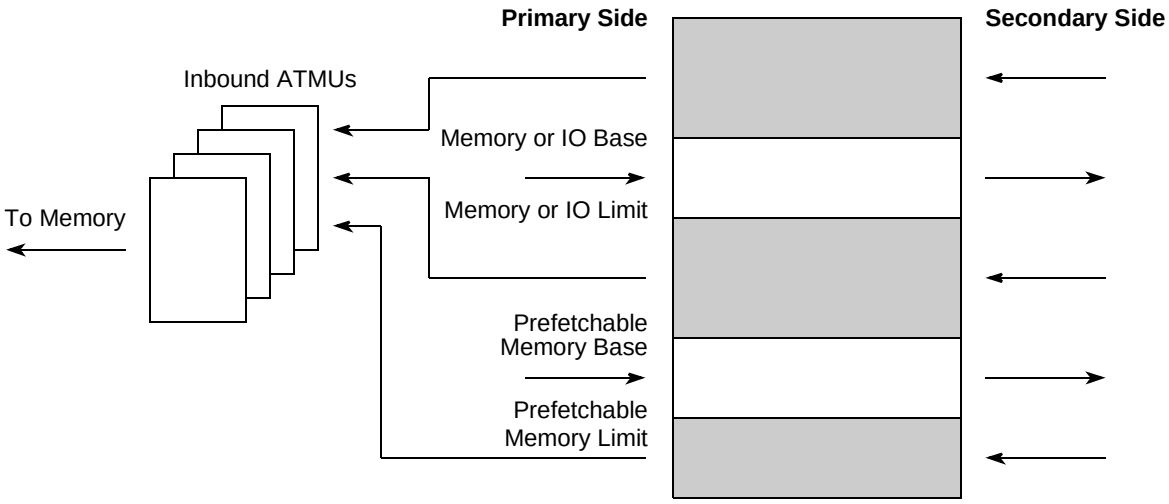


Figure 17-34. RC Inbound Transaction Flow

17.5.1.8.9 PCI Express Inbound Translation Address Registers (PEXITAR*n*)

Offset Window 0: 0xDE0
Window 1: 0xDC0
Window 2: 0xDA0
Window 3: 0xD80

Access: Read/Write

	31	24	23	20	19				0
R	—				TA				
W									
Reset	0x000F_FF00								

Figure 17-35. PCI Express Inbound Translation Address Registers (PEXITAR*n*)

The PCI Express inbound translation address registers, shown in **Figure 17-35**, contain the translated internal platform address to be used. **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
31–20	—	Reserved
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 [31:12].

17.5.1.8.10 PCI Express Inbound Window Base Address Registers (PEXIWBARn)

Offset Window 0: 0xDE8, Window 1: 0xDC8
Window 2: 0xDA8, Window 3: 0xD88

Access: Read/Write

	0	11	12	31
R	WBEA (Windows 2 and 3 only)		WBA	
W				
Reset	All zeros			

Figure 17-36. PCI Express Inbound Window Base Address Registers (PEXIWBAR*n*)

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 17-37 describes the fields of the PCI Express inbound window base address registers.

Table 17-37. PEXIWBAR*x* Field Descriptions

Bit	Name	Description
WBEA 31–20	WBEA	Window base extended address. This field corresponds to PCI Express address bits [31:20]. Note that the extended address is supported for windows 2 and 3 only; for PEXIWBAR0 and PEXIWBAR1, these bits are reserved and must be 0.
WBA 19–0	WBEA	Window base extended address. This field corresponds to PCI Express address bits [19:0].

17.5.1.8.11 PCI Express Inbound Window Base Extended Address Registers (PEXIWBEAR*n*)

PEXIWBEAR2 PCI Express Inbound Window Base Extended Address Registers 2–3 Offset 0xDAC
PEXIWBEAR3 Offset 0xDBC

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 17-38** describes the fields of the PCI Express inbound window base extended address registers.

Table 17-38. 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]

17.5.1.8.12 PCI Express Inbound Window Attributes Registers (PEXIWAR*n*)

PEXIWAR0	PCI Express Inbound	Offset 0xDF0
PEXIWAR1	Window Attributes	Offset 0xDD0
PEXIWAR2	Registers 1–3	Offset 0xDB0
PEXIWAR3		Offset 0xD90

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	0	0	0	0	0	0	1	1	1	1	0	1	0	0
Win0	1	0	0	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
Win0	0	1	0	0	0	0	0	0	0	0	0	1	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 17-39** describes the fields of the PCI Express inbound window attributes registers.

Table 17-39. 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.	
PF 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 15.4** for information about the OCN-to-MBus bridges.	0000 OCN-to-MBus Bridge 1 1110 OCN-to-MBus Bridge 0 1111 CCSR space 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 17-39. 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.

Note: When using the same OCN-to-MBus (O2M) for Read and Write transactions, the write transactions may write incorrect data for specific access sequences. To preclude this scenario, use one bridge for Write transactions and the other bridge for read transactions.

17.5.1.8.13 PCI Express MSI Inbound Translation Address Register (PEXMSIITAR)—RC Mode Only

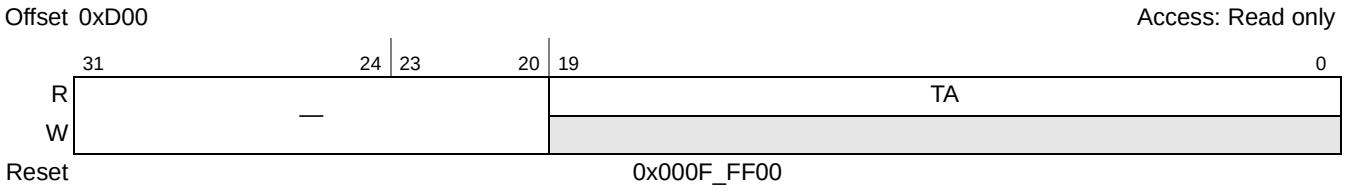


Figure 17-37. PCI Express MSI Inbound Translation Address Register (PEXMSIITAR)

The PCI Express MSI inbound translation address register, shown in **Figure 17-37**, contains the translated internal platform address to be used. The translated address is always mapped to CCSR space similarly to PEXITAR0. This register only applies in RC mode.**Table 17-40** describes the fields of the PCI Express MSI inbound translation address register.

Table 17-40. PCI Express MSI Inbound Translation Address Register Field Descriptions

Bits	Name	Description
31–20	—	Reserved
19–0	TA	Translation address. Target address which indicates the starting point of the inbound translated address. Mapped to the CCSR range.

17.5.1.8.14 PCI Express MSI Inbound Window Base Address Register (PEXMSIIWBAR)—RC Mode Only

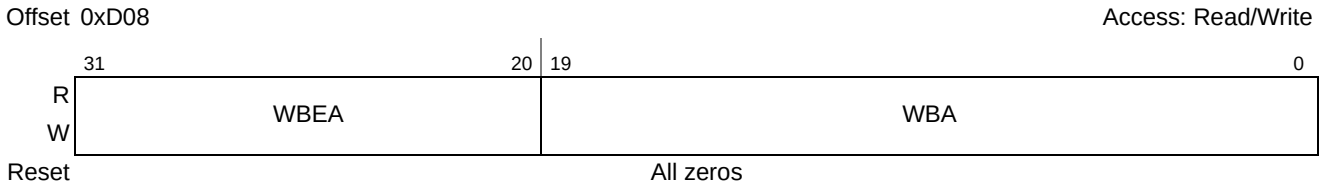


Figure 17-38. PCI Express MSI Inbound Window Base Address Register (PEXMSIIWBAR)

The PCI Express MSI inbound window base address register, shown in **Figure 17-38**, selects the base address for the windows which are translated to an alternate target address space. In root complex (RC) mode, addresses for the 64-bit MSI inbound transactions are compared to this window. This register only applies in RC mode. **Table 17-41** describes the fields of the PCI Express MSI inbound window base address registers.

Table 17-41. PCI Express MSI Inbound Window Base Address Register Field Descriptions

Bits	Name	Description
31–20	WBEA	Window base extended address. This field corresponds to PCI Express address bits [43:32].
19–0	WBA	Window base address. Source address which is the starting point for the inbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to PCI Express address bits [31:12].

17.5.1.8.15 PCI Express MSI Inbound Window Base Extended Address Registers (PEXMSIIWBEAR)—RC Mode Only

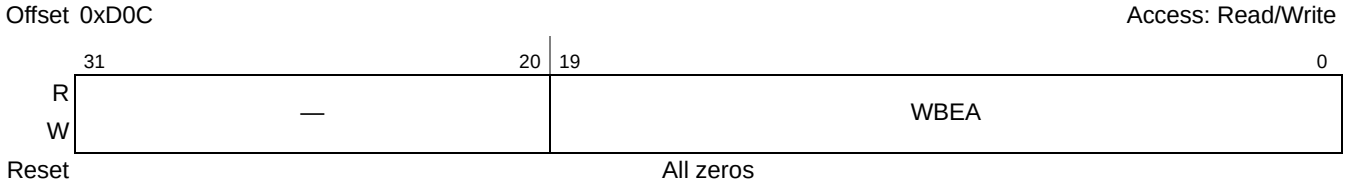


Figure 17-39. PCI Express MSI Inbound Window Base Extended Address Register (PEXMSIIWBEAR)

The PCI Express MSI inbound window base extended address register, shown in **Figure 17-39**, contains the most-significant bits of a 64-bit MSI base address. This register only applies in RC mode. **Table 17-42** describes the fields of the PCI Express MSI inbound window base extended address registers.

Table 17-42. PCI Express MSI Inbound Window Base Extended Address Register Field Descriptions

Bits	Name	Description
31–20	—	Reserved
19–0	WBEA	Window base extended address. This field corresponds to PCI Express address bits [63:44]

17.5.1.8.16 PCI Express MSI Inbound Window Attributes Registers (PEXMSIIWAR)—RC Mode Only

Offset 0xD10 Access: Mixed

	31	30	29	28	24	23	20	19	16	15	12	11	6	5	0
R	EN		PF	—		TRGT		RTT		WTT		—		IWS	
W															
Reset	0	0	0	0	0 0 0 0	1	1 1 1	0	1 0 0	0	1 0 0	0 0 0 0	0 0	1 0	0 0 1 1

Figure 17-40. PCI Express MSI Inbound Window Attributes Register (PEXMSIIWAR)

The PCI Express MSI inbound window attributes register, shown in **Figure 17-40**, defines the window sizes to translate and other attributes for the translations. The size is fixed at -M window. This register only applies in RC mode. **Table 17-43** describes the fields of the PCI Express MSI inbound window attributes register.

Table 17-43. PCI Express MSI Inbound Window Attribute Register Field Descriptions

Bits	Name	Description
31	EN	Enable. This bit controls the enabling/disabling of the MSI translation window. 0 Disable inbound window translation 1 Enable inbound window translation
30	—	Reserved
29	PF	Prefetchable. This bit indicates that the address space is prefetchable. This bit corresponds to the prefetchable bit in the BAR in the PCI Express type 0 header. This bit drives the BAR's prefetchable bit in EP mode. 0 Not prefetchable 1 Prefetchable
28–24	—	Reserved
23–20	.TRGT	Target Interface. This field selects the interface to which the inbound transaction is sent. 1111 CCSR space All others reserved.
19–16	RTT	Read transaction type. Read transaction type to send to target interface. 0100 Read All others reserved.
15–12	WTT	Write transaction type. Write transaction type to send to target interface. 0100 Write All others reserved.

Table 17-43. PCI Express MSI Inbound Window Attribute Register Field Descriptions

Bits	Name	Description
11–6	—	Reserved
5–0	IWS	Inbound window size. Inbound translation window size N which is the encoded $2^{(N + 1)}$-bytes window size. The smallest window size is 4 Kbytes. For EP mode, this field directly controls the size of the BARs. This field is driven by config pin 000000 Reserved ... 001010 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size 100100 Reserved ... 111111 Reserved

17.5.1.9 PCI Express Error Management Registers

17.5.1.9.1 PCI Express Error Detect Register (PEX_ERR_DR)

Offset 0xE00 Access: w1c

	31	30	24	23	22	21	20	19	18	17	16
R	ME	—		PCT	PAT	PCAC	PNM	CDNSC	CRSNC	ICCA	IACA
W	w1c			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	All zeros										

	15	14	13	12	11	10	9	8	7	6	5	4	0
R	CRST	MIS	IOIS	CIS	CIEP	IOIEP	OAC	IOIA	IMBA	IIOBA	LDDE	—	
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c		
Reset	All zeros												

Figure 17-41. PCI Express Error Detect Register (PEX_ERR_DR)

The PCI Express error detect register, shown in **Figure 17-41**, 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.**Table 17-44** describes the fields of the PCI Express error detect register.

Table 17-44. 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	PAT	PCI Express Ack time-out. An internal request timeout was seen. This is considered as a internal fatal error. Once this bit is set, all outstanding transactions will be canceled. No new transactions will be accepted until this bit is clear. It is expected that software needs to perform the proper clean-up operation (that is, hot reset) to get the link to function again before clearing this bit. 1 An Ack time-out on the PCI Express link was detected. 0 No Ack time-out on the PCI Express link detected.

Table 17-44. PCI Express Error Detect Register Field Descriptions (Continued)

Bits	Name	Description
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.
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 (that is, UR, CA or CRS status). 1 Completion with data non successful packet was detected. 0 No completion with data non successful packet detected.
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 17.5.1.1.4**, *PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR),* on page 17-33 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 17.4.2.1**, *Outbound ATMU Message Generation,* on page 17-13 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
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

Table 17-44. PCI Express Error Detect Register Field Descriptions (Continued)

Bits	Name	Description
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	IMBA	Invalid memory base address. An outbound memory transaction with a translated address not hitting into the Memory Base/Limit register or Prefetchable Memory Base/Limit register. Only valid in RC mode. This feature is enabled by setting PEX_CONFIG[OB_CK]. 1 Invalid memory base address was detected 0 No invalid memory base address detected
6	IIOBA	Invalid I/O base address. An outbound I/O transaction with a translated address not hitting into the IO Base/Limit register or IO Upper Base/Limit register. Only valid in RC mode. This feature is enabled by setting PEX_CONFIG[OB_CK]. 1 Invalid I/O base address was detected 0 No invalid I/O base address detected
5	LDDE	Link down data error. An outbound read transaction was issued through an ATMU window when the link was down. 1 Link down data error detected 0 No Link down data error detected
4–0	—	Reserved

17.5.1.9.2 PCI Express Error Interrupt Enable Register (PEX_ERR_EN)

Offset 0xE08 Access: Read/Write

	31–24	23	22	21	20	19	18	17	16
R W	—	PCT IE	PAT IE	PCAC IE	PNM IE	CDNSC IE	CRSNC IE	ICCA IE	IACA IE
Reset	All zeros								

	15	14	13	12	11	10	9	8	7	6	5	4–0
R W	CRST IE	MIS IE	IOIS IE	CIS IE	CIEP IE	IOIEP IE	OAC IE	IOIA IE	IMBA IE	IIOBA IE	LDDE IE	—
Reset	All zeros											

Figure 17-42. PCI Express Error Interrupt Enable Register (PEX_ERR_EN)

The PCI Express error interrupt enable register, shown in **Figure 17-42**, allows interrupts to be generated when the corresponding PCI Express error detect register bits are set. **Table 17-45** describes the fields of the PCI Express error interrupt enable register.

Table 17-45. 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	PATIE	PCI Express Ack time-out interrupt enable. When set and PEX_ERR_DR[PAT]=1 will generate an interrupt. 1 Enable PCI Express ack time-out interrupt generation 0 Disable PCI Express ack time-out interrupt generation
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
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.

Table 17-45. PCI Express Error Interrupt Enable Register Field Descriptions (Continued)

Bits	Name	Description
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	IMBAIE	Invalid memory base address interrupt enable. When set and PEX_ERR_DR[IMBA]=1 will generate an interrupt. 1 Enable memory base address interrupt generation 0 Disable memory base address interrupt generation
6	IIOBAIE	Invalid I/O base address interrupt enable. When set and PEX_ERR_DR[IIOBA]=1 will generate an interrupt. 1 Enable I/O base address interrupt generation 0 Disable I/O base address interrupt generation
5	LDDEIE	Link down data error interrupt enable. When set and PEX_ERR_DR[LDDEIE]=1 will generate an interrupt. 1 Enable link down data error interrupt generation 0 Disable link down data error interrupt generation
4–0	—	Reserved

17.5.1.9.3 PCI Express Error Disable Register (PEX_ERR_DISR)

Offset0xE10 Access: Read/Write

	31	30	24	23	22	21	20	19	18	17	16
R W	MED	—		PCTD	PATD	PCACD	PNMD	CDNSCD	CRSNCD	ICCAD	IACAD
Reset	All zeros										

	15	14	13	12	11	10	9	8	7	6	5	4	0
R W	CRSTD	MISD	IOISD	CISD	CIEPD	IOIEPD	OACD	IOIAD	IMBAD	IIOBAD	LDDED	—	
Reset	All zeros												

Figure 17-43. PCI Express Error Disable Register (PEX_ERR_DISR)

The PCI Express error disable register, shown in **Figure 17-43**, controls the setting of the PCI Express error detect register's bits. **Table 17-46** describes the fields of the PCI Express error disable register.

Table 17-46. 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	PATD	PCI Express Ack time-out disable. When set will disable the setting of PEX_ERR_DR[PAT] bit. 1 Disable PCI Express ack time-out detection 0 Enable PCI Express ack time-out detection
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 17-46. 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	IMBAD	Invalid memory base address disable. When set will disable the setting of PEX_ERR_DR[IMBA]. 1 Enable memory base address detection 0 Disable memory base address detection
6	IIOBAD	Invalid I/O base address disable. When set will disable the setting of PEX_ERR_DR[IIOBA]. 1 Enable I/O base address detection 0 Disable I/O base address detection
5	LDDED	Link down data error disable. When set will disable the setting of PEX_ERR_DR[LDDE]. 1 Enable link down data error detection 0 Disable link down data error detection
4–0	—	Reserved

17.5.1.9.4 PCI Express Error Capture Status Register (PEX_ERR_CAP_STAT)

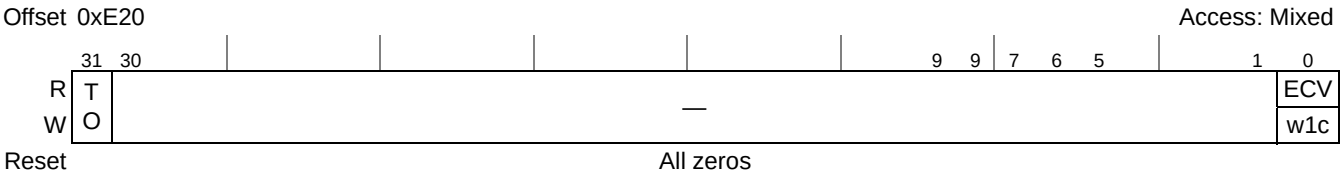


Figure 17-44. PCI Express Error Capture Status Register (PEX_ERR_CAP_STAT)

The PCI Express error capture status register, shown in **Figure 17-44**, 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. **Table 17-47** describes the fields of the PCI Express error capture status register.

Table 17-47. PCI Express Error Capture Status Register Field Descriptions

Bits	Name	Description
31	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.
30–1	—	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.

17.5.1.9.5 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 17-45**.

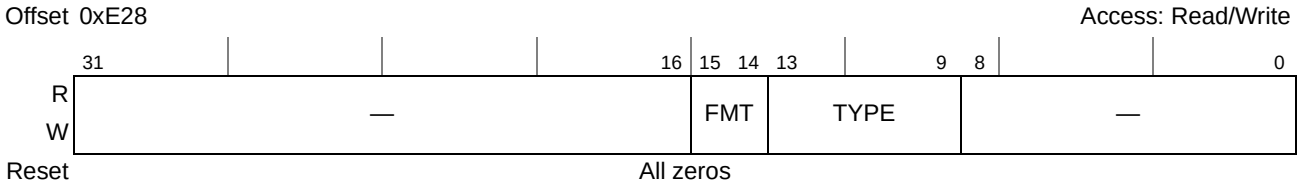


Figure 17-45. PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0) Internal Source, Outbound Transaction

Table 17-48 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 17-48. 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 2.0 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 2.0 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 17-46**.

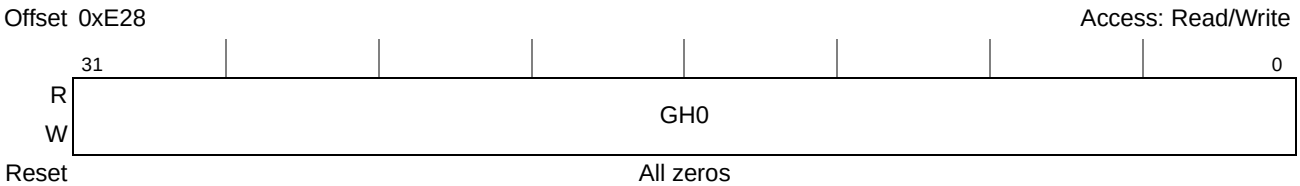


Figure 17-46. PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0) External Source, Inbound Transaction

Table 17-49 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 17-49. 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]

17.5.1.9.6 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 17-47**.

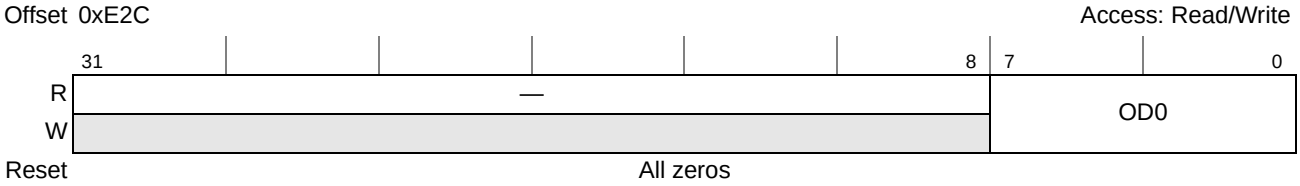


Figure 17-47. PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1)
Internal Source, Outbound Transaction

Table 17-50 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 17-50. 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 17-48**.

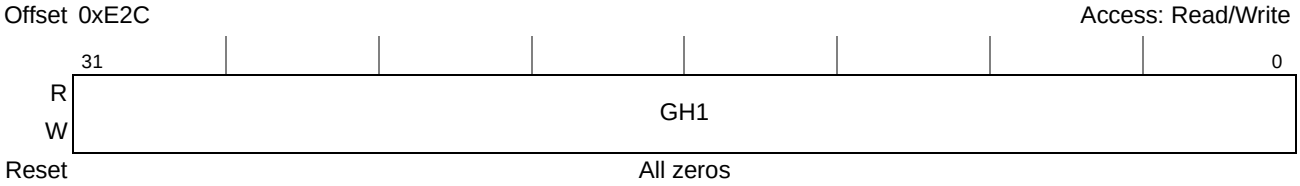


Figure 17-48. PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1)
External Source, Inbound Transaction

Table 17-51 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 17-51. 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]

17.5.1.9.7 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 17-49**.

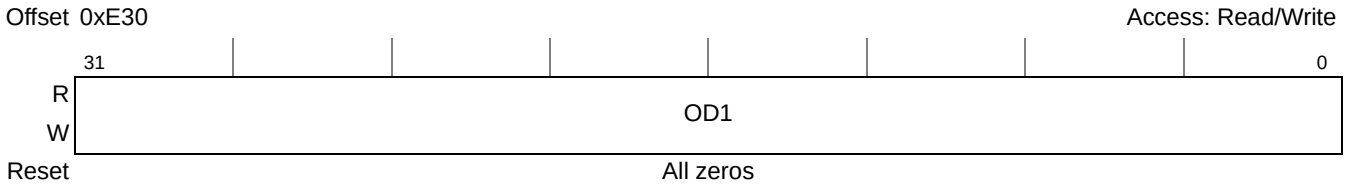


Figure 17-49. PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2) Internal Source, Outbound Transaction

Table 17-50 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 17-52. 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 17-48**.

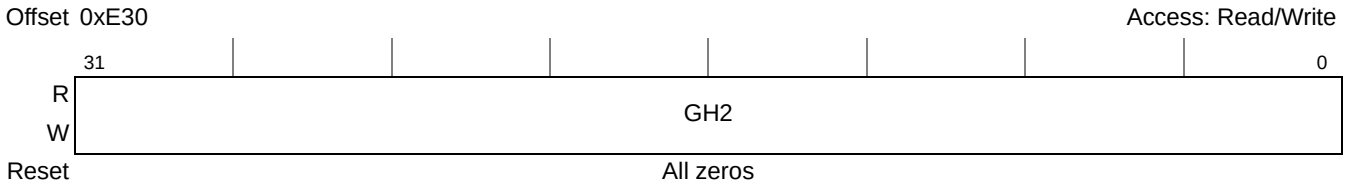


Figure 17-50. PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2) External Source, Inbound Transaction

Table 17-51 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 17-53. 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

17.5.1.9.8 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 17-51**.

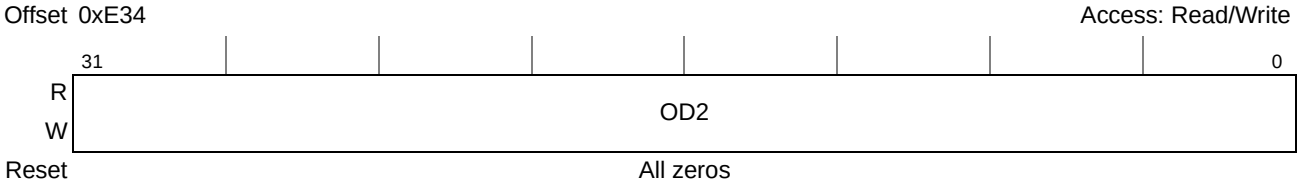


Figure 17-51. PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3)
Internal Source, Outbound Transaction

Table 17-50 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 17-54. 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 17-48**.

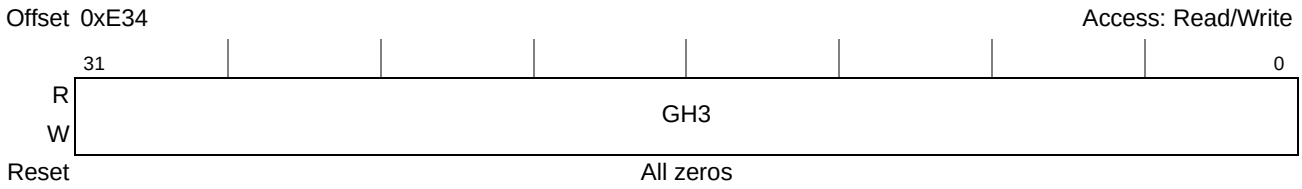


Figure 17-52. PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3)
External Source, Inbound Transaction

Table 17-51 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 17-55. 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.

17.5.1.10 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)

17.5.1.10.1 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 17.4.1.2**, *Byte Order for Configuration Transactions*, on page 17-10 for more information.

Note that external configuration transactions should not be attempted until the link has successfully trained.

17.5.1.10.2 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.

17.5.1.10.3 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.

17.5.1.10.4 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.

17.5.1.11 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 17-53**.

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 17-53. 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.

17.5.1.11.1 Common PCI Compatible Configuration Header Registers

This section details the registers that are common to both type 0 and type 1 configuration headers.

17.5.1.11.2 PCI Express Vendor ID Register—Offset 0x00

The vendor ID register, shown in **Figure 17-54**, is used to identify the manufacturer of the device.

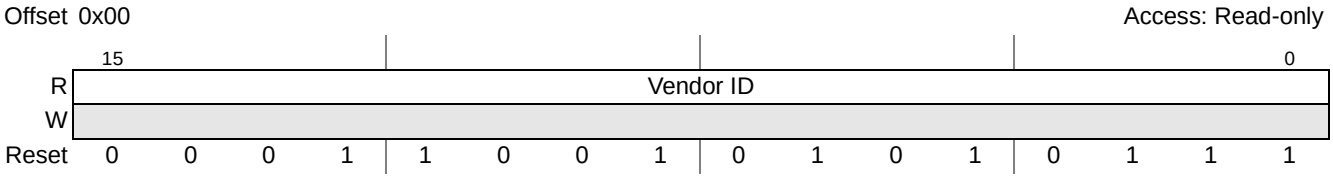


Figure 17-54. PCI Express Vendor ID Register

Table 17-56 describes the vendor ID register fields.

Table 17-56. PCI Express Vendor ID Register Field Description

Bits	Name	Description
15–0	Vendor ID	0x1957 (Freescale)

17.5.1.11.3 PCI Express Device ID Register—Offset 0x02

The device ID register, shown in **Figure 17-55**, is used to identify the device.

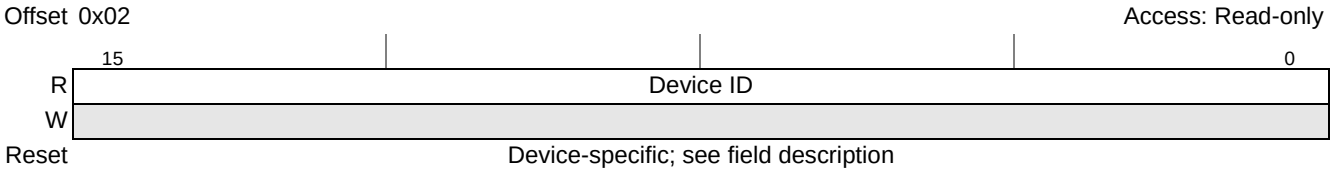


Figure 17-55. PCI Express Device ID Register

Table 17-57 describes the device ID register fields.

Table 17-57. PCI Express Device ID Register Field Description

Bits	Name	Description
15–0	Device ID	Value is 0x182A

17.5.1.11.4 PCI Express Command Register—Offset 0x04

The command register, shown in **Figure 17-56**, provides control over the ability to generate and respond to PCI Express cycles.

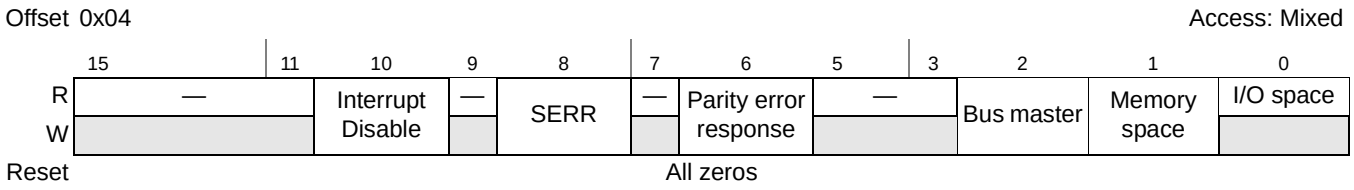


Figure 17-56. PCI Express Command Register

Table 17-58 describes the bits of the command register.

Table 17-58. 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 17.5.1.12.8**, *PCI Express Device Control Register—0x54,* on page 17-101 and the advance error reporting capability structure described in **Section 17.5.1.13.1** through **Section 17.5.1.13.12**.
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 17.5.1.12.8**, *PCI Express Device Control Register—0x54,* on page 17-101 and the advance error reporting capability structure described in **Section 17.5.1.13.1** through **Section 17.5.1.13.12**.
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.

17.5.1.11.5 PCI Express Status Register—Offset 0x06

The status register, shown in **Figure 17-57**, 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 17-57. PCI Express Status Register

The definition of each bit is given in **Table 17-59**.

Table 17-59. 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	Signalled 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	Signalled 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 17.5.1.12.8**, *PCI Express Device Control Register—0x54,* on page 17-101 and the advance error reporting capability structure described in **Section 17.5.1.13.1** through **Section 17.5.1.13.12**.

17.5.1.11.6 PCI Express Revision ID Register—Offset 0x08

The revision ID register, shown in **Figure 17-58**, is used to identify the revision of the device.

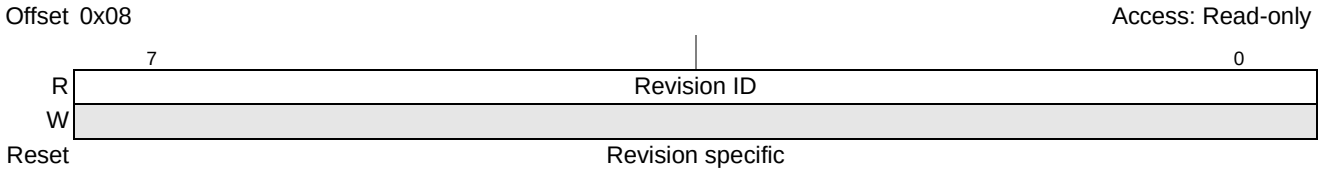


Figure 17-58. PCI Express Revision ID Register

Table 17-60 describes the revision ID register fields.

Table 17-60. PCI Express Revision ID Register Field Descriptions

Bits	Name	Description
7–0	Revision ID	Revision specific.

17.5.1.11.7 PCI Express Class Code Register—Offset 0x09

The class code register, shown in **Figure 17-59**, 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.

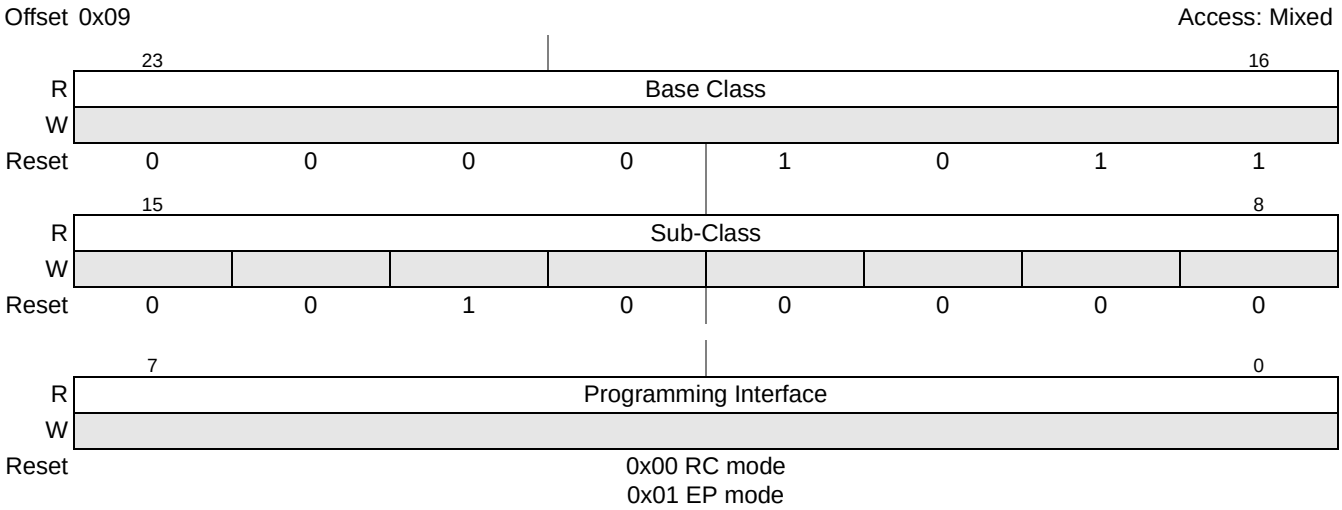


Figure 17-59. PCI Express Class Code Register

Table 17-61 describes the class code register fields.

Table 17-61. 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

17.5.1.11.8 PCI Express Cache Line Size Register—Offset 0x0C

The cache line size register, shown in **Figure 17-60**, is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

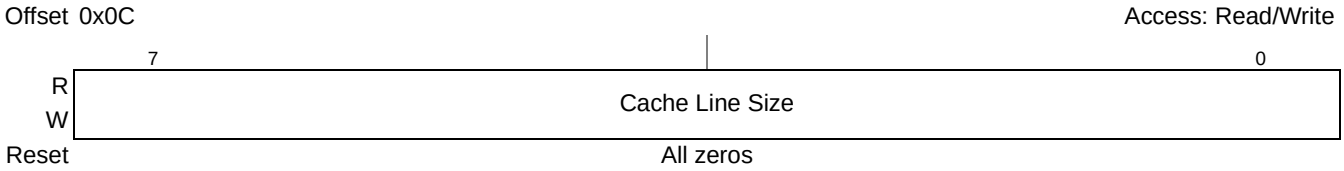


Figure 17-60. PCI Express Bus Cache Line Size Register

Table 17-62 describes the cache line size register.

Table 17-62. 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.

17.5.1.11.9 PCI Express Latency Timer Register—0x0D

The latency timer register, shown in **Figure 17-61**, is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

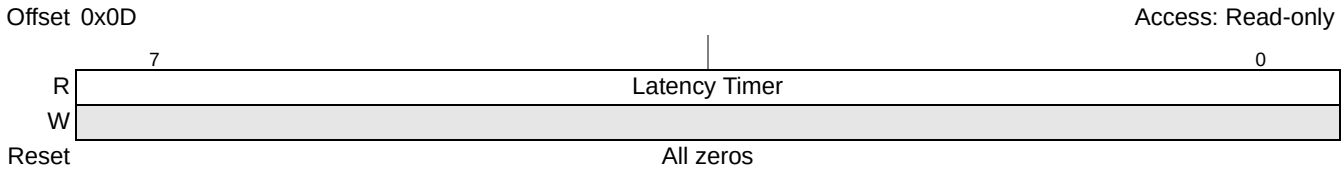


Figure 17-61. PCI Express Bus Latency Timer Register

Table 17-63 describes the PCI Express latency timer register (PLTR).

Table 17-63. 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.

17.5.1.11.10 PCI Express Header Type Register—0x0E

The PCI Express header type register, shown in **Figure 17-60**, is used to identify the layout of the PCI compatible header.

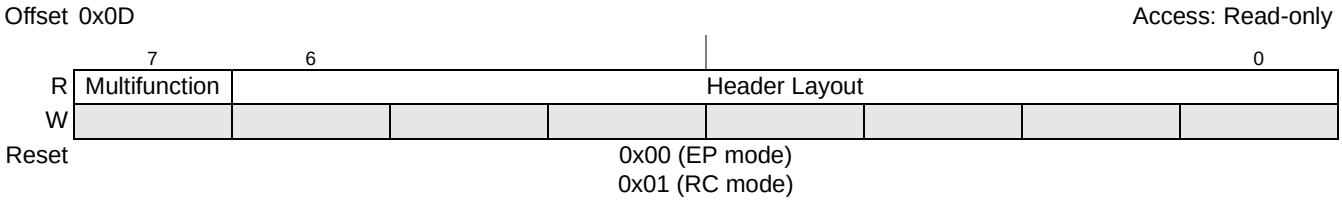


Figure 17-62. PCI Express Bus Latency Timer Register

Table 17-63 describes the PCI Express header type register.

Table 17-64. 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 17-63** for type 0 layout. 0x01 Root Complex. See **Figure 17-75** for type 1 layout. All other encodings reserved.

17.5.1.11.11 PCI Express BIST Register—0x0F

The BIST register is optional and reserved on the PCI Express controller.

17.5.1.11.12 Type 0 Configuration Header

The type 0 header is shown in **Figure 17-63**.

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 17-63. PCI Express PCI-Compatible Configuration Header—Type 0

Section 17.5.1.11.1, *Common PCI Compatible Configuration Header Registers*, on page 17-73 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.

17.5.1.11.13 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 17.5.1.8.6**, *PCI Express Inbound ATMU Registers*, on page 17-50 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 17-64**.

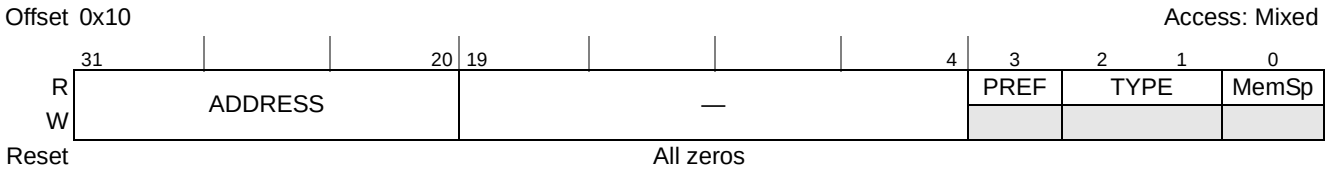


Figure 17-64. PCI Express Base Address Register 0 (PEXCSRBAR)

Table 17-65 describes the PCI Express configuration and status register base address register.

Table 17-65. 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 17-65**.

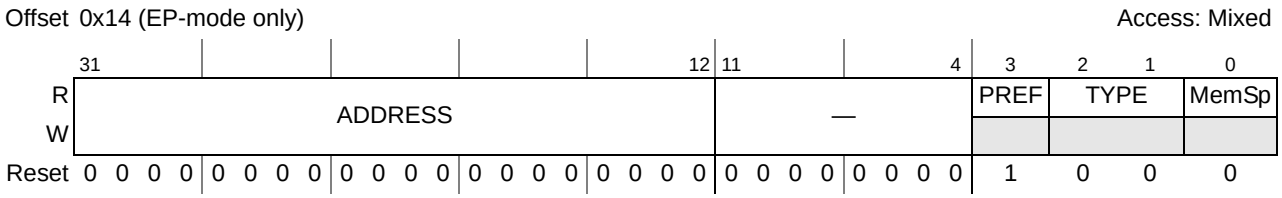


Figure 17-65. 32-Bit Memory Base Address Register (BAR1)

Table 17-66 describes the PCI Express 32-bit memory BAR fields.

Table 17-66. 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 17-66**.

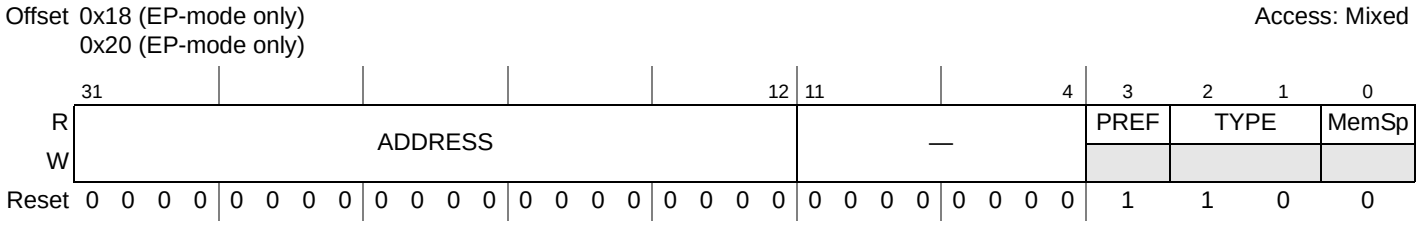


Figure 17-66. 64-Bit Low Memory Base Address Register

Table 17-67 describes the PCI Express 64-bit low memory BAR fields.

Table 17-67. 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 17-67**.

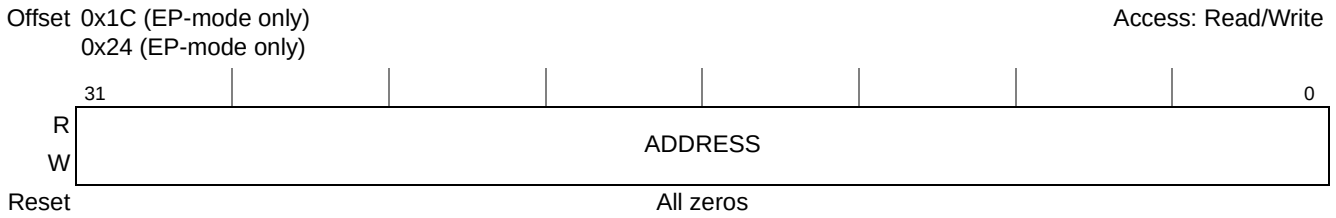


Figure 17-67. 64-Bit High Memory Base Address Register

Table 17-68 describes the PCI Express 64-bit low memory BAR fields.

Table 17-68. 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.

17.5.1.11.14 PCI Express Subsystem Vendor ID Register (EP-Mode Only)—0x2C

The PCI Express subsystem vendor ID register is used to identify the subsystem.

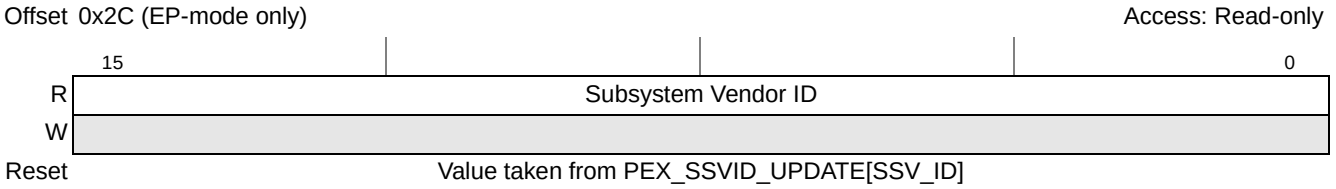


Figure 17-68. PCI Express Subsystem Vendor ID Register

Table 17-69. 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 17.5.1.13.17**, *PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478,* on page 17-126 for more information.

17.5.1.11.15 PCI Express Subsystem ID Register (EP-Mode Only)—0x2E

The PCI Express subsystem ID register is used to identify the subsystem.

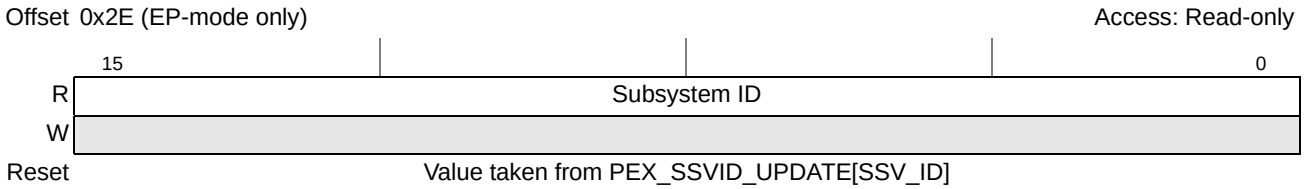


Figure 17-69. PCI Express Subsystem ID Register

Table 17-70. 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 17.5.1.13.17**, *PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478,* on page 17-126 for more information.

17.5.1.11.16 Capabilities Pointer Register—0x34

The capabilities pointer identifies additional functionality supported by the device.

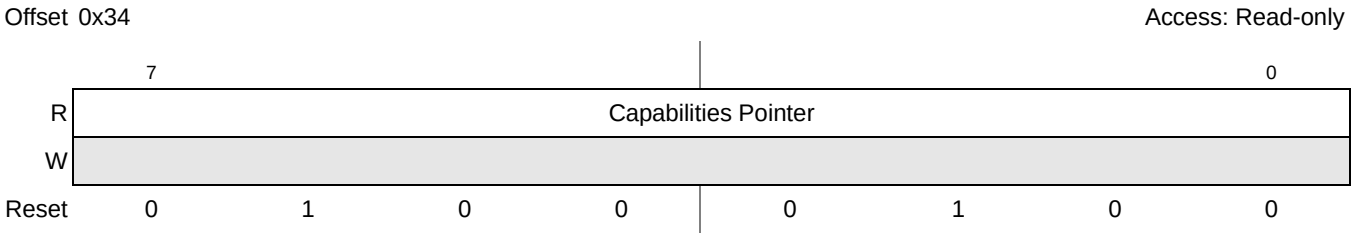


Figure 17-70. Capabilities Pointer Register

Table 17-71. 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 17.5.1.12**, *PCI Compatible Device-Specific Configuration Space,* on page 17-96 for more information.

17.5.1.11.17 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.

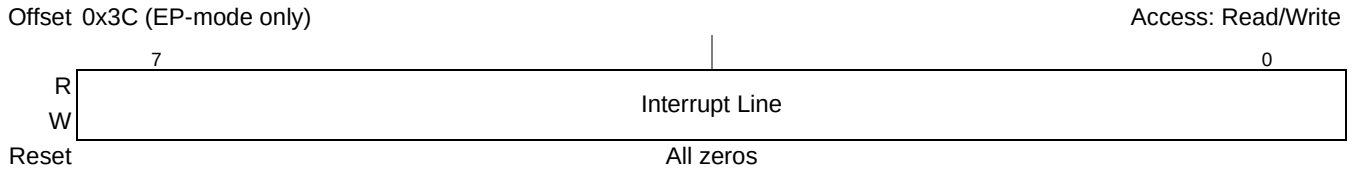


Figure 17-71. PCI Express Interrupt Line Register

Table 17-72. PCI Express Interrupt Line Register Field Description

Bits	Name	Description
7–0	Interrupt Line	Used to communicate interrupt line routing information.

17.5.1.11.18 PCI Express Interrupt Pin Register—0x3D

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.

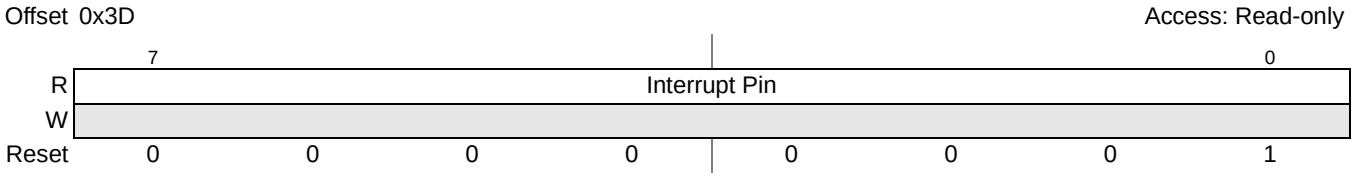


Figure 17-72. PCI Express Interrupt Pin Register

Table 17-73. 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.

17.5.1.11.19 PCI Express Minimum Grant Register (EP-Mode Only)—0x3E

This register does not apply to PCI Express. It is present for legacy purposes.

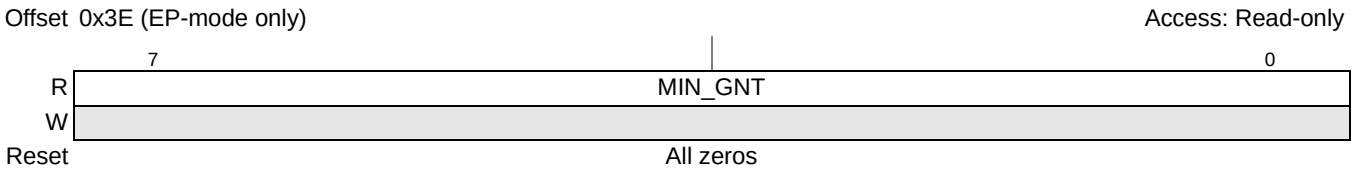


Figure 17-73. PCI Express Maximum Grant Register (MAX_GNT)

Table 17-74. PCI Express Maximum Grant Register Field Description

Bits	Name	Description
7–0	MIN_GNT	Does not apply for PCI Express.

17.5.1.11.20 PCI Express Maximum Latency Register (EP-Mode Only)—0x3F

This register does not apply to PCI Express. It is present for legacy purposes.

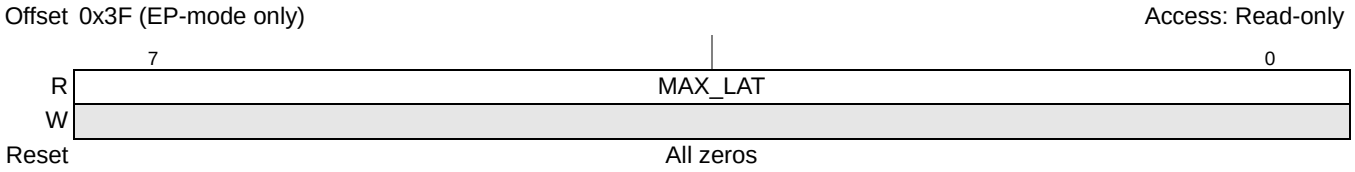


Figure 17-74. PCI Express Maximum Latency Register (MAX_LAT)

Table 17-75. PCI Express Maximum Latency Register Field Description

Bits	Name	Description
7–0	MAX_LAT	Does not apply for PCI Express.

17.5.1.11.21 Type 1 Configuration Header

The type 1 header is shown in **Figure 17-75**.

Reserved

				Address Offset (Hex)
Device ID		Vendor ID		00
Status		Command		04
Class Code			Revision ID	08
BIST	Header Type	Latency Timer	Cache Line Size	0C
Base Address Register 0				10
				14
Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number	18
Secondary Status		I/O Limit	I/O Base	1C
Memory Limit		Memory Base		20
Prefetchable Memory Limit		Prefetchable Memory Base		24
Prefetchable Base Upper 32 Bits				28
Prefetchable Limit Upper 32 Bits				2C
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits		30
			Capabilities Pointer	34
Expansion ROM Base Address				38
Bridge Control		Interrupt Pin	Interrupt Line	3C

Figure 17-75. PCI Express PCI-Compatible Configuration Header—Type 1

Section 17.5.1.11.1, *Common PCI Compatible Configuration Header Registers*, on page 17-73 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.

17.5.1.11.22 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 17-64**.

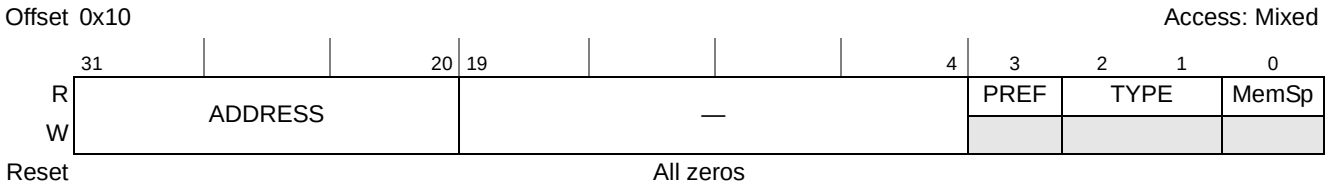


Figure 17-76. PCI Express Base Address Register 0 (PEXCSRBAR)

Table 17-65 describes the PCI Express configuration and status register base address register.

Table 17-76. 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

17.5.1.11.23 PCI Express Primary Bus Number Register—Offset 0x18

The primary bus number register is shown in **Figure 17-77**.

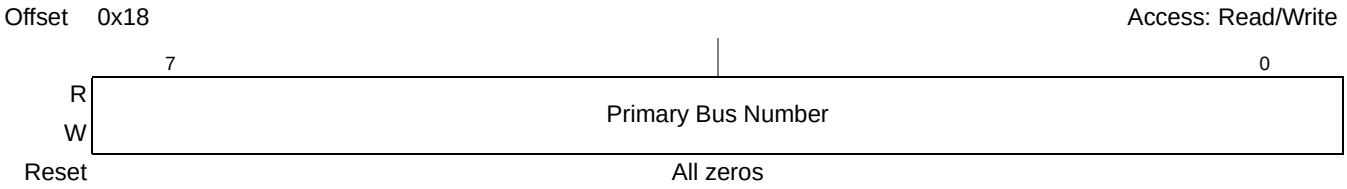


Figure 17-77. PCI Express Primary Bus Number Register

Table 17-77 describes the primary bus number register fields.

Table 17-77. 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.

17.5.1.11.24 PCI Express Secondary Bus Number Register—Offset 0x19

The secondary bus number register is shown in **Figure 17-78**.

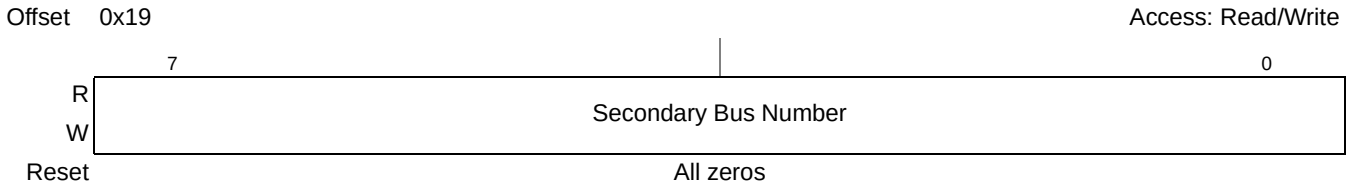


Figure 17-78. PCI Express Secondary Bus Number Register

Table 17-78 describes the secondary bus number register fields.

Table 17-78. 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.

17.5.1.11.25 PCI Express Subordinate Bus Number Register—Offset 0x1A

The subordinate bus number register is shown in **Figure 17-79**.

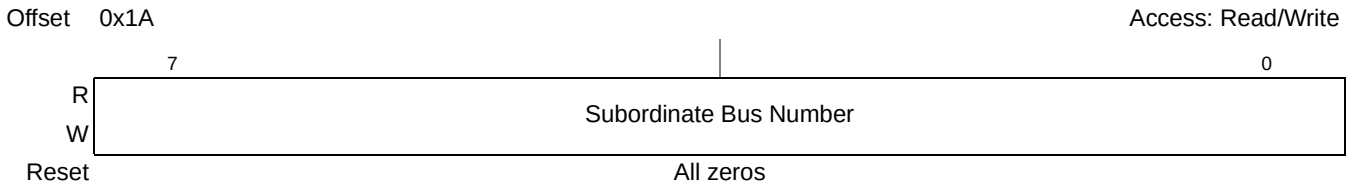


Figure 17-79. PCI Express Subordinate Bus Number Register

Table 17-79 describes the subordinate bus number register fields.

Table 17-79. 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.

17.5.1.11.26 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.

17.5.1.11.27 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 17-79**.

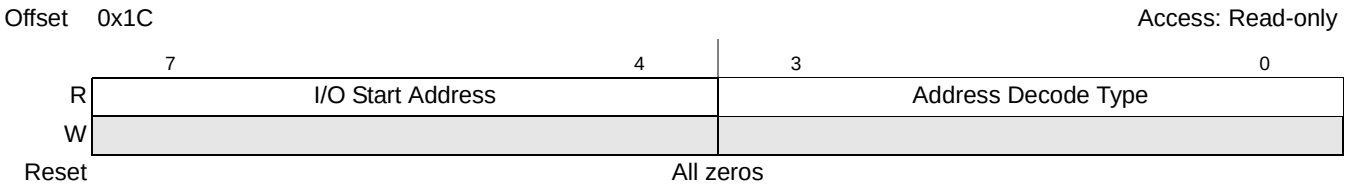


Figure 17-80. PCI Express I/O Base Register

Table 17-79 describes the I/O base register fields.

Table 17-80. 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.

17.5.1.11.28 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 17-79**.

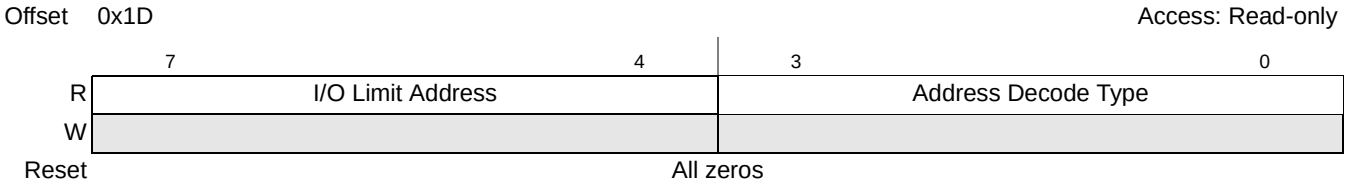


Figure 17-81. PCI Express I/O Limit Register

Table 17-79 describes the I/O limit register fields.

Table 17-81. 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.

17.5.1.11.29 PCI Express Secondary Status Register—0x1E

The PCI Express secondary status register is shown in **Figure 17-82**. 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 17.5.1.13.20**, *Secondary Status Interrupt Mask Register (RC-Mode Only)—0x5A0,* on page 17-129 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 17-82. PCI Express Secondary Status Register

Table 17-82 describes the PCI Express secondary status register fields.

Table 17-82. 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

Note: After Gen2 training and before initiating any transactions toward the PCI Express controller, always clear the DPE bit to clear any receive packet errors caused by the training.

17.5.1.11.30 PCI Express Memory Base Register—0x20

The memory base register is shown in **Figure 17-83**.

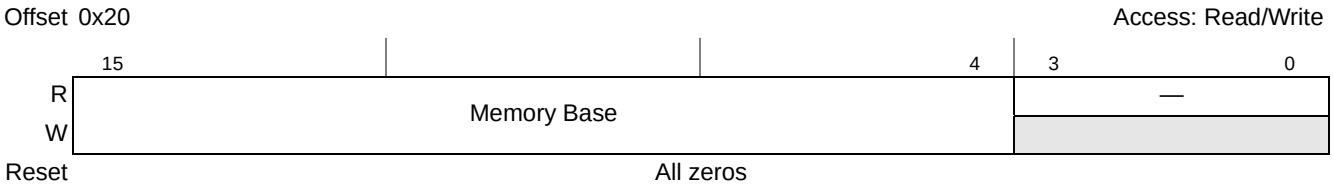


Figure 17-83. PCI Express Memory Base Register

Table 17-83 describes the memory base register fields.

Table 17-83. 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

17.5.1.11.31 PCI Express Memory Limit Register—0x22

The memory limit register is shown in **Figure 17-84**.

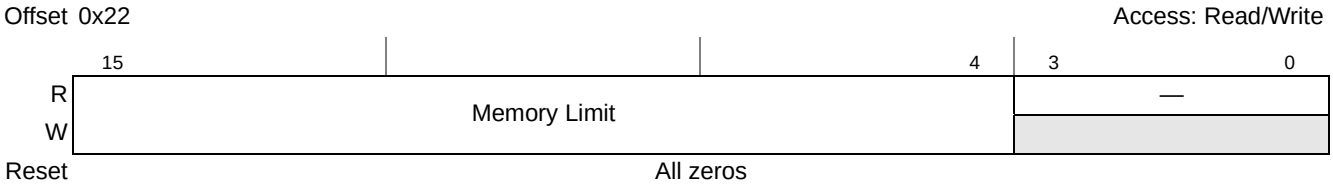


Figure 17-84. PCI Express Memory Limit Register

Table 17-84 describes the memory base register fields.

Table 17-84. 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

17.5.1.11.32 PCI Express Prefetchable Memory Base Register—0x24

The prefetchable memory base register is shown in **Figure 17-85**.

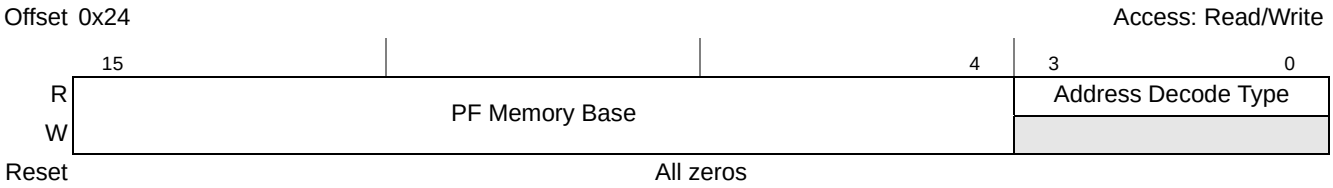


Figure 17-85. PCI Express Prefetchable Memory Base Register

Table 17-85 describes the prefetchable memory base register fields.

Table 17-85. 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.

17.5.1.11.33 PCI Express Prefetchable Memory Limit Register—0x26

The prefetchable memory limit register is shown in **Figure 17-86**.

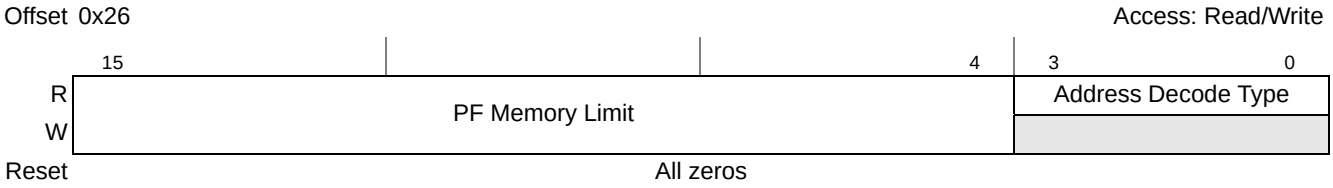


Figure 17-86. PCI Express Prefetchable Memory Limit Register

Table 17-86 describes the prefetchable memory limit register fields.

Table 17-86. 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.

17.5.1.11.34 PCI Express Prefetchable Base Upper 32 Bits Register—0x28

The PCI Express prefetchable memory base upper 32 bits register is shown in **Figure 17-87**.

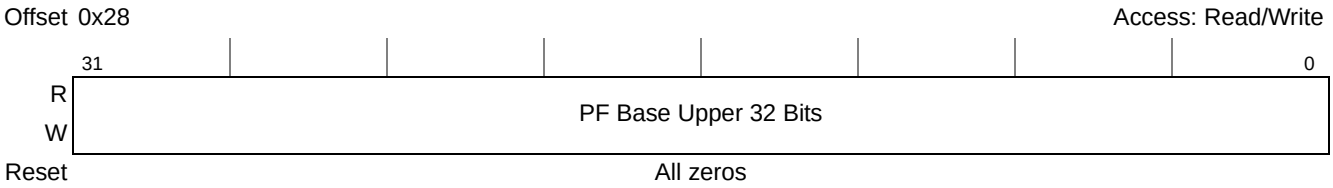


Figure 17-87. PCI Express Prefetchable Base Upper 32 Bits Register

Table 17-87 describes the PCI Express prefetchable memory base upper 32 bits register fields.

Table 17-87. 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.

17.5.1.11.35 PCI Express Prefetchable Limit Upper 32 Bits Register—0x2C

The PCI Express prefetchable memory base upper 32 bits register is shown in **Figure 17-88**.

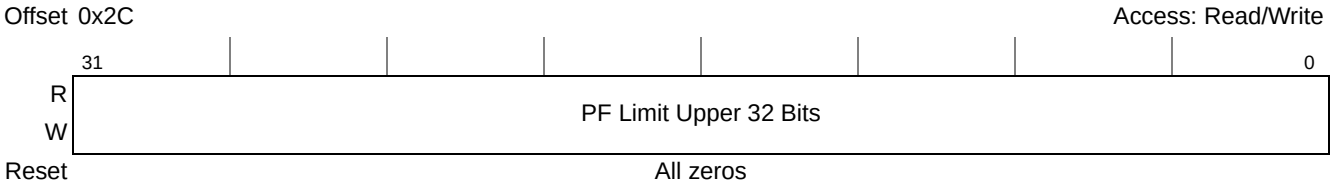


Figure 17-88. PCI Express Prefetchable Limit Upper 32 Bits Register

Table 17-88 describes the PCI Express prefetchable memory limit upper 32 bits register fields.

Table 17-88. 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.

17.5.1.11.36 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 17-89**.

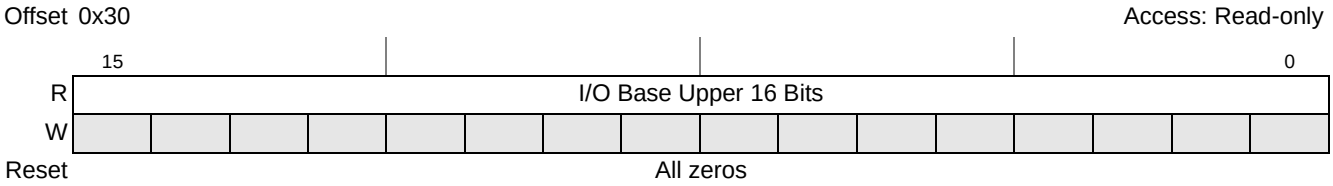


Figure 17-89. PCI Express I/O Base Upper 16 Bits Register

Table 17-89 describes the I/O base upper 16 bits register fields.

Table 17-89. 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.

17.5.1.11.37 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 17-90**.

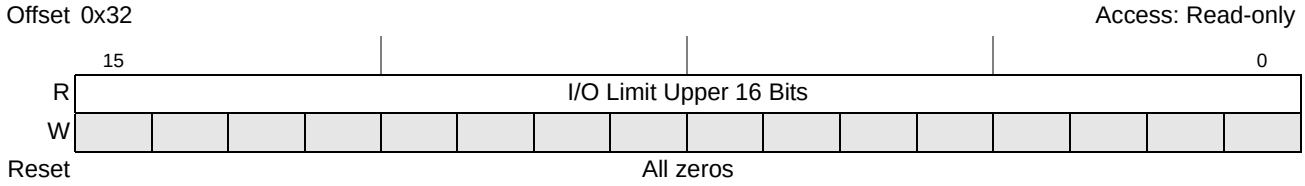


Figure 17-90. PCI Express I/O Limit Upper 16 Bits Register

Table 17-90 describes the I/O limit upper 16 bits register fields.

Table 17-90. 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.

17.5.1.11.38 Capabilities Pointer Register—0x34

The capabilities pointer identifies additional functionality supported by the device.

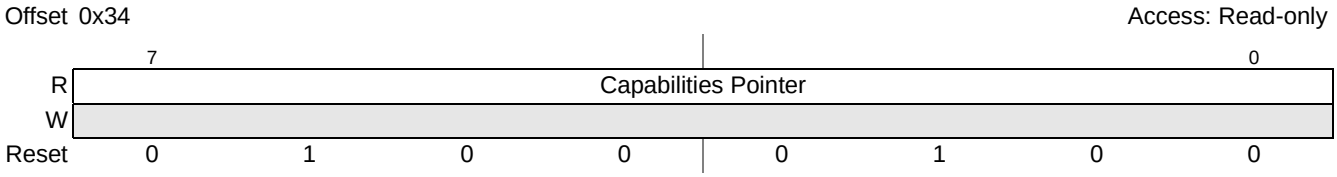


Figure 17-91. Capabilities Pointer Register

Table 17-91. 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 17.5.1.12**, *PCI Compatible Device-Specific Configuration Space,* on page 17-96," for more information.

17.5.1.11.39 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.

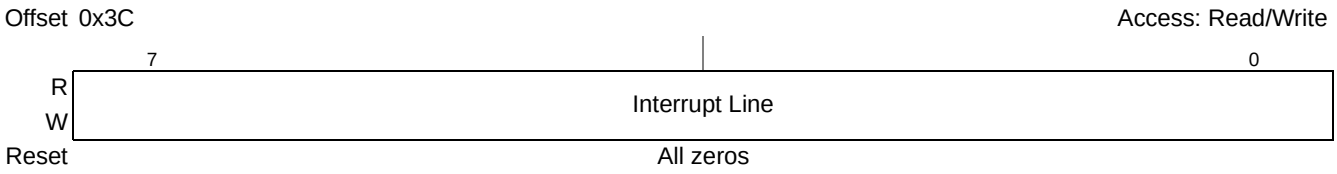


Figure 17-92. PCI Express Interrupt Line Register

Table 17-92. PCI Express Interrupt Line Register Field Description

Bits	Name	Description
7–0	Interrupt Line	Used to communicate interrupt line routing information.

17.5.1.11.40 PCI Express Interrupt Pin Register—0x3D

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.

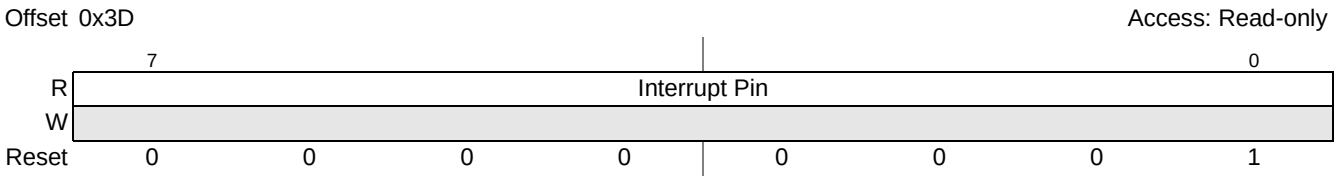


Figure 17-93. PCI Express Interrupt Pin Register

Table 17-93. 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.

17.5.1.11.41 PCI Express Bridge Control Register—0x3E

The PCI Express bridge control register is shown in **Figure 17-94**.

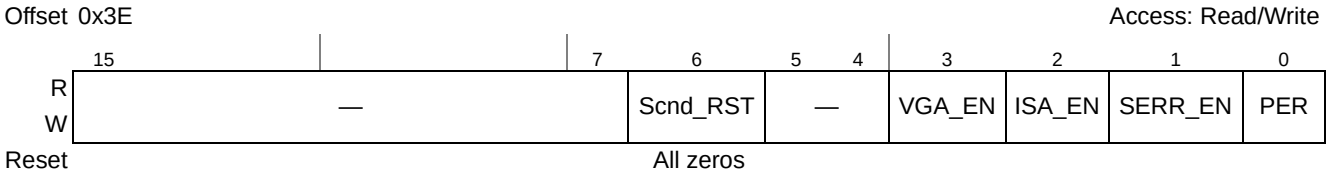


Figure 17-94. PCI Express Bridge Control Register

Table 17-94 describes the PCI Express bridge control register fields.

Table 17-94. 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.

17.5.1.12 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 17.5.1.11**, *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
Device Capabilities 2			70
Device Status 2	Device Control 2		74
Link Capabilities 2			78
Link Status 2	Link Control 2		7C
Slot Capabilities 2			80
Slot Status 2	Slot Control 2		84
MSI Message Control	Next Pointer (NULL)	MSI Message Capability ID	88
MSI Message Address			8C
MSI Upper Message Address			90
	MSI Message Data		94
			98 FF

Figure 17-95. PCI Compatible Device-Specific Configuration Space

17.5.1.12.1 PCI Express Power Management Capability ID Register—0x44

The PCI Express power management capability ID register is shown in **Figure 17-96**.

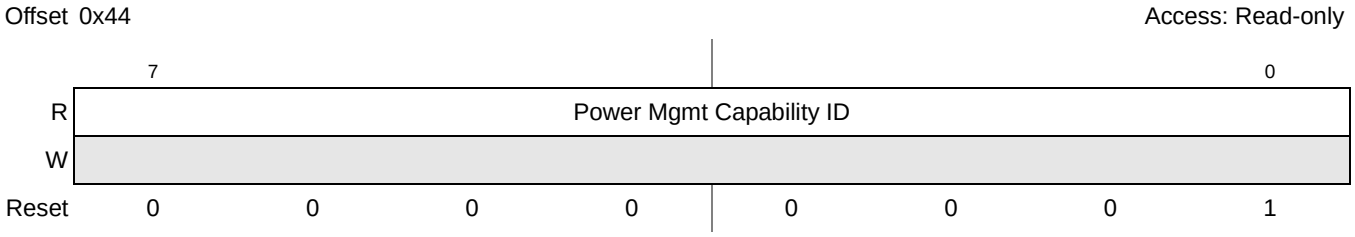


Figure 17-96. PCI Express Power Management Capability ID Register

Table 17-95. PCI Express Power Management Capability ID Register Field Description

Bits	Name	Description
7–0	Power Mgmt Capability ID	Power Management = 0x01

17.5.1.12.2 PCI Express Power Management Capabilities Register—0x46

The PCI Express power management capabilities register is shown in **Figure 17-97**.

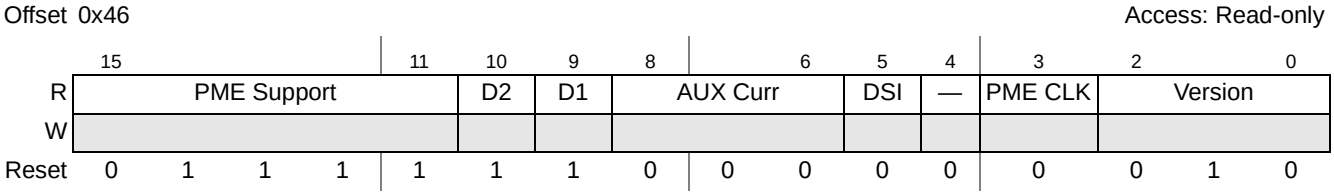


Figure 17-97. PCI Express Power Management Capabilities Register

Table 17-96. 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

17.5.1.12.3 PCI Express Power Management Status and Control Register—0x48

The PCI Express power management status and control register is shown in **Figure 17-98**.

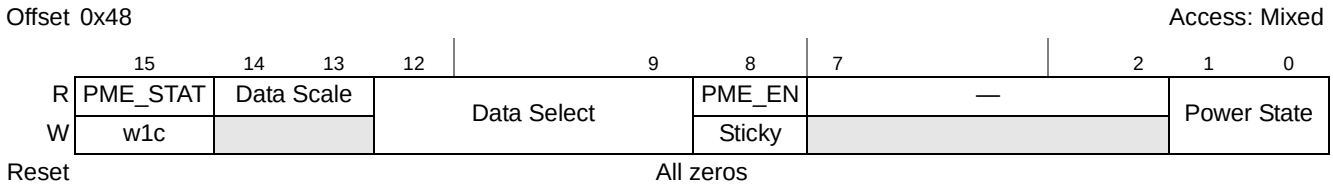


Figure 17-98. PCI Express Power Management Status and Control Register

Table 17-97. 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

17.5.1.12.4 PCI Express Power Management Data Register—0x4B

The PCI Express power management data register is shown in **Figure 17-99**.

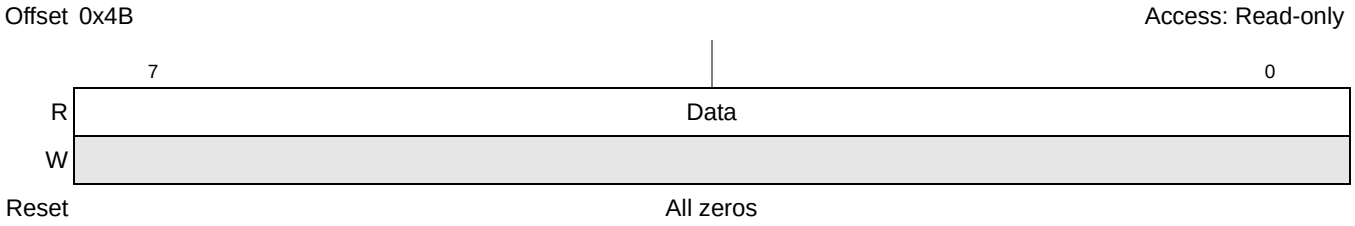


Figure 17-99. PCI Express Power Management Data Register

Table 17-98. PCI Express Power Management Data Register Field Description

Bits	Name	Description
7–0	Data	Obtained from *PCI Express Base Specification, Revision 1.0a*

17.5.1.12.5 PCI Express Capability ID Register—0x4C

The PCI Express capability ID register is shown in **Figure 17-100**.

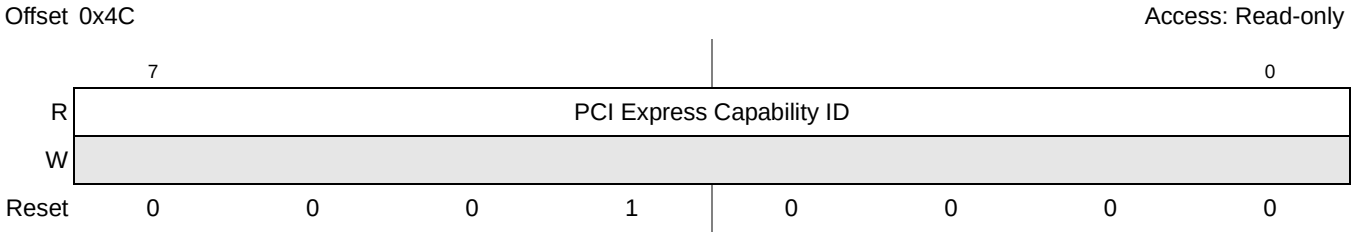


Figure 17-100. PCI Express Capability ID Register

Table 17-99. PCI Express Capability ID Register Field Description

Bits	Name	Description
7–0	PCI Express Capability ID	PCI Express = 0x10

17.5.1.12.6 PCI Express Capabilities Register—0x4E

The PCI Express capabilities register is shown in **Figure 17-101**.

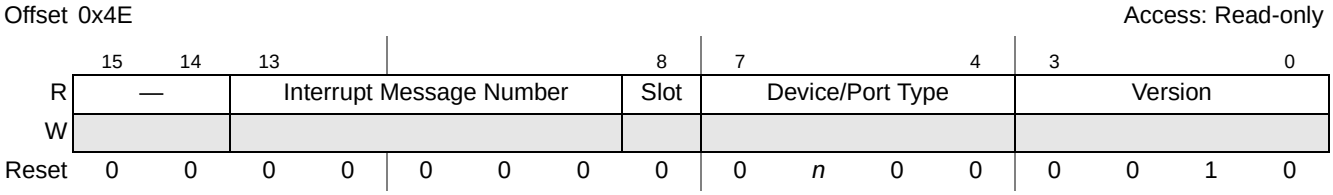


Figure 17-101. PCI Express Capabilities Register

Table 17-100. 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.

17.5.1.12.7 PCI Express Device Capabilities Register—0x50

The PCI Express device capabilities register is shown in **Figure 17-102**.

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 17-102. PCI Express Device Capabilities Register

Table 17-101. 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

17.5.1.12.8 PCI Express Device Control Register—0x54

The PCI Express device control register is shown in **Figure 17-103**.

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 17-103. PCI Express Device Control Register

Table 17-102. 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

17.5.1.12.9 PCI Express Device Status Register—0x56

The PCI Express device status register is shown in **Figure 17-104**.

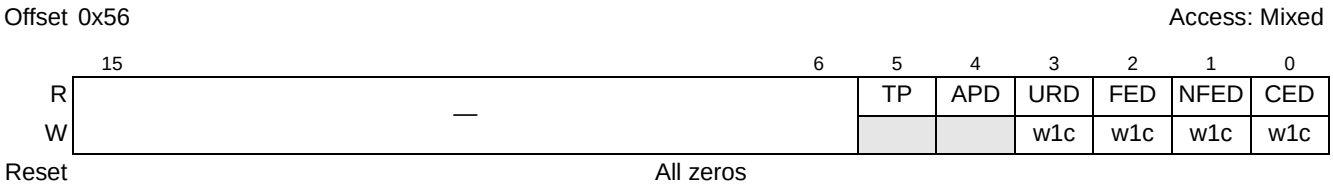


Figure 17-104. PCI Express Device Status Register

Table 17-103. 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

Note: After training and before initiating any transactions toward the PCI Express controller, always clear the CED bit.

17.5.1.12.10 PCI Express Link Capabilities Register—0x58

The PCI Express link capabilities register is shown in **Figure 17-105**.

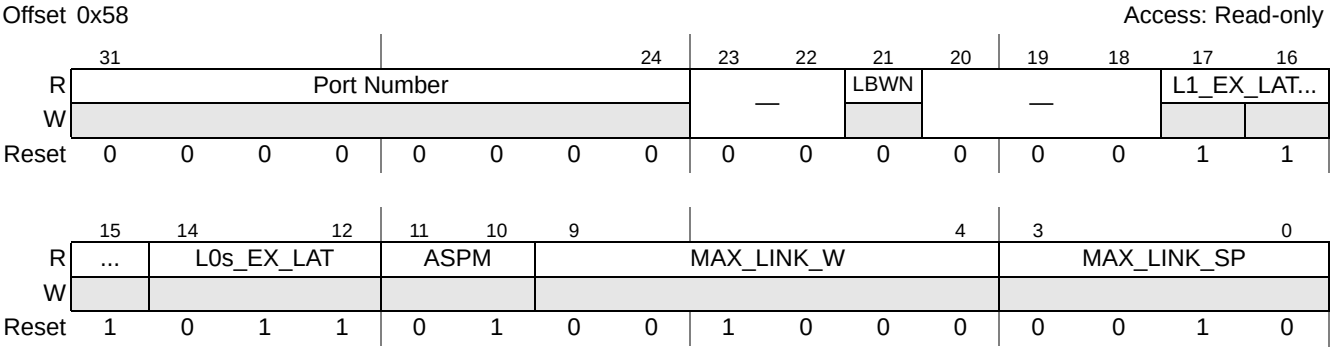


Figure 17-105. PCI Express Link Capabilities Register

Table 17-104. PCI Express Link Capabilities Register Field Description

Bits	Name	Description
31–24	Port Number	
23–22	—	Reserved
21	LBWN	Link bandwidth notification capable.
20–18	—	Reserved
17–15	L1_EX_LAT	L1 exit latency

Table 17-104. PCI Express Link Capabilities Register Field Description (Continued)

Bits	Name	Description
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

17.5.1.12.11 PCI Express Link Control Register—0x5C

The PCI Express link control register is shown in **Figure 17-106**.

Offset 0x5C — Access: Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—				LABIE	LBMIE	HWAWD	—	EXT_SYNC	CCC	RL	LD	RCB	—	ASPM_CTL	
Reset	All zeros															

Figure 17-106. PCI Express Link Control Register

Table 17-105. PCI Express Link Control Register Field Description

Bits	Name	Description
15–12	—	Reserved
11	LABIE	Link autonomous bandwidth interrupt enable.
10	LBMIE	Link bandwidth management interrupt enable.
9	HWAWD	Hardware autonomous width disable.
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

17.5.1.12.12 PCI Express Link Status Register—0x5E

The PCI Express link status register is shown in **Figure 17-107**.

Offset 0x5E — Access: Read-only

	15	13	12	11	10	9	4	3	0
R	LABS	LBMS	—	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 17-107. PCI Express Link Status Register

Table 17-106. PCI Express Link Status Register Field Description

Bits	Name	Description
15	LABS	Link autonomous bandwidth status.
14	LBMS	Link bandwidth management status.
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

17.5.1.12.13 PCI Express Slot Capabilities Register—0x60

The PCI Express slot capabilities register is shown in **Figure 17-108**.

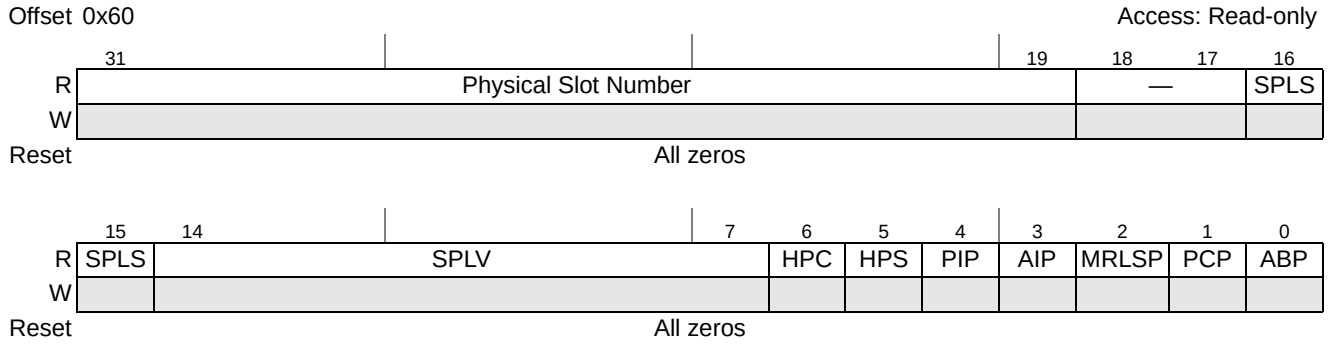


Figure 17-108. PCI Express Slot Capabilities Register

Table 17-107. 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.
3	AIP	Attention indicator present.
2	MRLSP	MRL sensor present.
1	PCP	Power controller present.
0	ABP	Attention button present.

17.5.1.12.14 PCI Express Slot Control Register—0x64

The PCI Express slot control register is shown in **Figure 17-109**.

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 17-109. PCI Express Slot Control Register

Table 17-108. 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.

17.5.1.12.15 PCI Express Slot Status Register—0x66

The PCI Express slot status register is shown in **Figure 17-110**.

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 17-110. PCI Express Slot Status Register

Table 17-109. 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.

17.5.1.12.16 PCI Express Root Control Register (RC Mode Only)—0x68

The PCI Express root control register is shown in **Figure 17-111**.

Offset 0x68 Access: Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—												PMEIE	SEFEE	SENFEE	SECEE
W																
Reset	All zeros															

Figure 17-111. PCI Express Root Control Register

Table 17-110. 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.

17.5.1.12.17 PCI Express Root Status Register (RC Mode Only)—0x6C

The PCI Express root status register is shown in **Figure 17-112**.

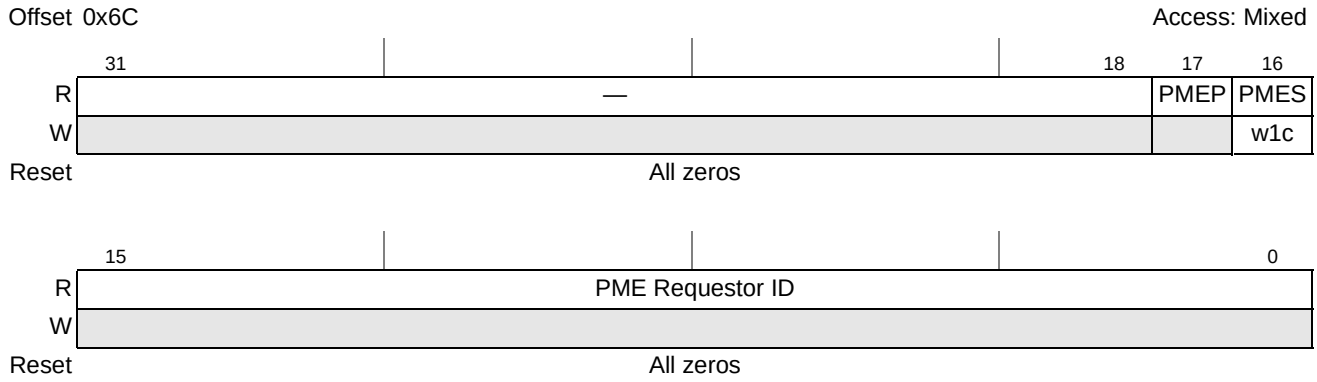


Figure 17-112. PCI Express Root Status Register

Table 17-111. 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.

17.5.1.12.18 PCI Express Device Capabilities 2 Register—0x70

The PCI Express device capabilities 2 register is shown in **Figure 17-113**.

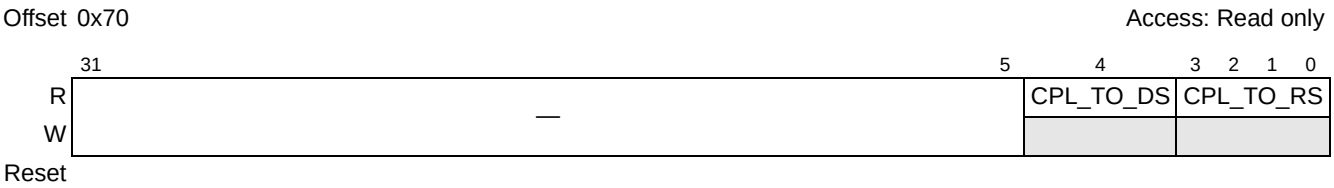


Figure 17-113. PCI Express Device Capabilities 2 Register

Table 17-112. PCI Express Device Capabilities 2 Register Field Description

Bits	Name	Description
31–5	—	Reserved
4	CPL_TO_DS	Completion timeout disable supported
3–0	CPL_TO_RS	Completion timeout ranges supported

17.5.1.12.19 PCI Express Device Control 2 Register—0x74

The PCI Express device control 2 register is shown in **Figure 17-111**.

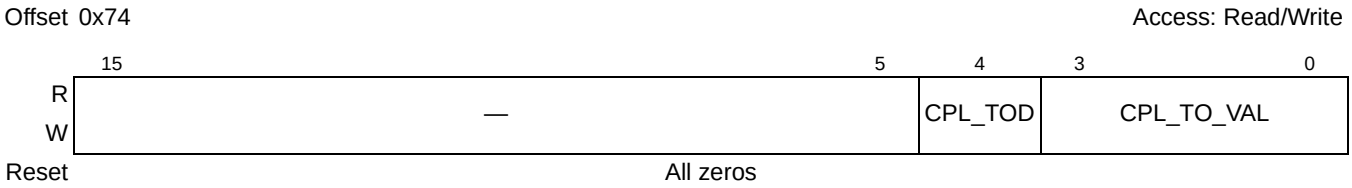


Figure 17-114. PCI Express Device Control 2 Register

Table 17-113. PCI Express Device Control 2 Register Field Description

Bits	Name	Description
15–5	—	Reserved
4	CPL_TOD	Completion timeout disable
3–0	CPL_TO_VAL	Completion timeout value

17.5.1.12.20 PCI Express Link Control 2 Register—0x7C

The PCI Express link control 2 register is shown in **Figure 17-111**.

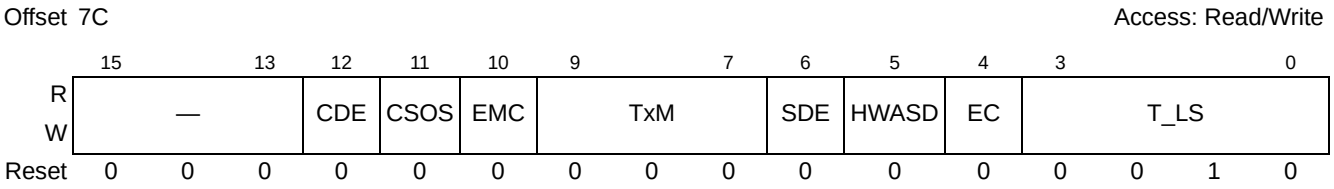


Figure 17-115. PCI Express Link Control 2 Register

Table 17-114. PCI Express Link Control 2 Register Field Description

Bits	Name	Description
15–13	—	Reserved
12	CDE	Compliance de-emphasis
11	CSOS	Compliance SOS
10	EMC	Enter modified compliance
9–7	TxM	Transmit margin
6	SDE	Selectable de-emphasis
5	HWASD	Hardware autonomous speed disable
4	EC	Enter compliance
3–0	T_LS	Target link speed

17.5.1.12.21 PCI Express Link Status 2 Register—0x7E

The PCI Express link status 2 register is shown in **Figure 17-111**.

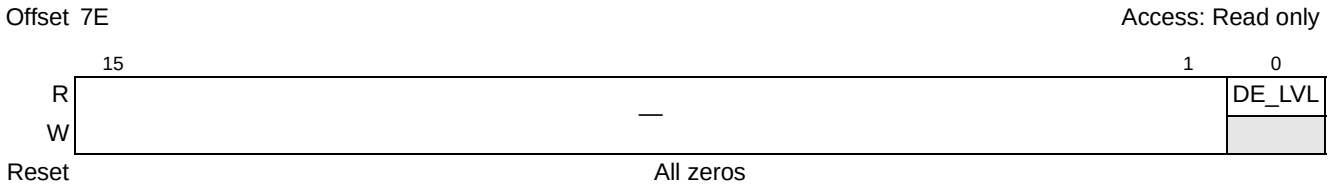


Figure 17-116. PCI Express Link Status 2 Register

Table 17-115. PCI Express Link Status 2 Register Field Description

Bits	Name	Description
15–1	—	Reserved
0	DE_LVL	Current de-emphasis level

17.5.1.12.22 PCI Express MSI Message Capability ID Register (EP Mode Only)—0x88

The PCI Express MSI message capability ID register is shown in **Figure 17-117**.

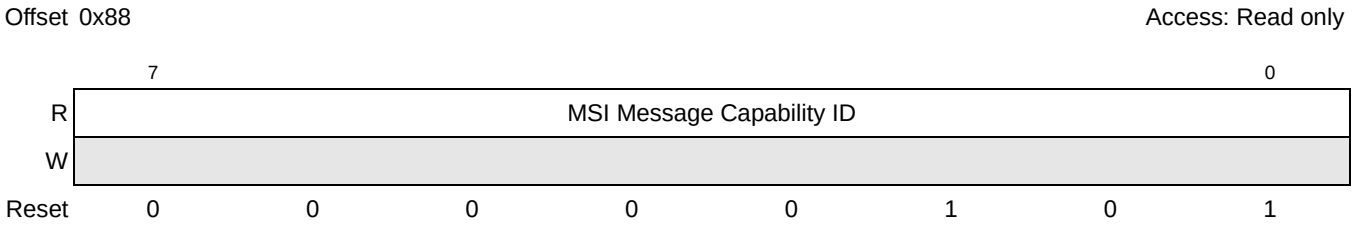


Figure 17-117. PCI Express Capability ID Register

Table 17-116. PCI Express Capability ID Register Field Description

Bits	Name	Description
7–0	MSI Message Capability ID	MSI Message = 0x05

17.5.1.12.23 PCI Express MSI Message Control Register (EP Mode Only)—0x8A

The PCI Express MSI message control register is shown in **Figure 17-118**.

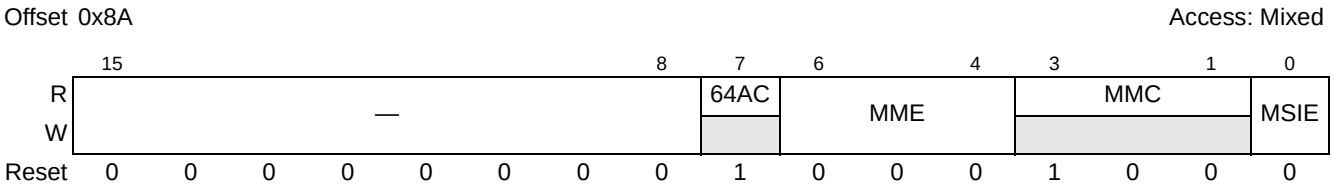


Figure 17-118. PCI Express MSI Message Control Register

Table 17-117. 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.

17.5.1.12.24 PCI Express MSI Message Address Register (EP Mode Only)—0x8C

The PCI Express MSI message address register is shown in **Figure 17-119**.

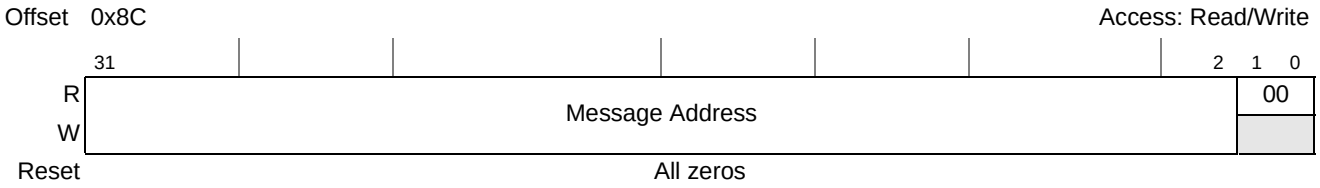


Figure 17-119. PCI Express MSI Message Address Register

Table 17-118. 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.

17.5.1.12.25 PCI Express MSI Message Upper Address Register (EP Mode Only)—0x90

The PCI Express MSI message upper address register is shown in **Figure 17-120**.

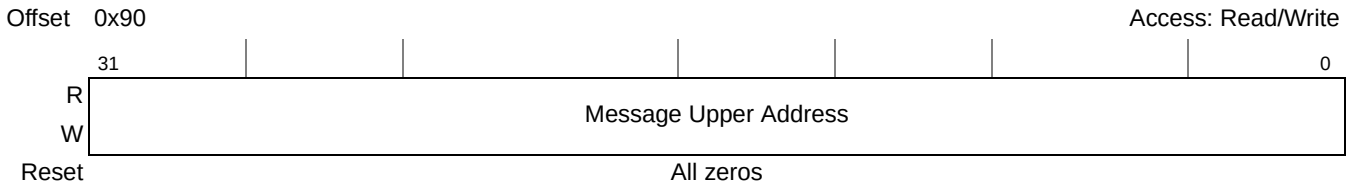


Figure 17-120. PCI Express MSI Message Upper Address Register

Table 17-119. PCI Express MSI Message Upper Address Register Field Description

Bits	Name	Description
31–0	Message Upper Address	System-specified message upper address

17.5.1.12.26 PCI Express MSI Message Data Register (EP Mode Only)—0x94

The PCI Express MSI message data register is shown in **Figure 17-121**.

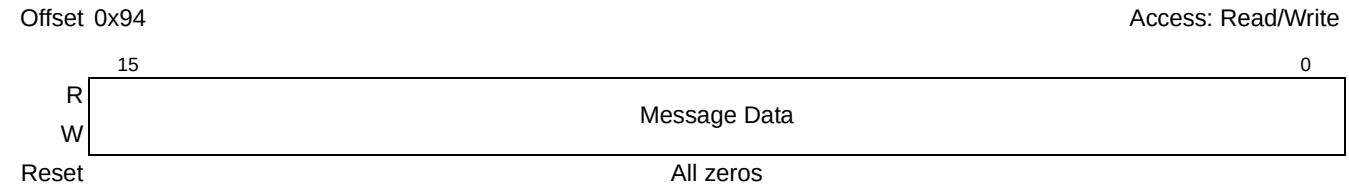


Figure 17-121. PCI Express MSI Message Data Register

Table 17-120. PCI Express MSI Message Data Register Field Description

Bits	Name	Description
15–0	Message Data	System-specified message.

17.5.1.13 PCI Express Extended Configuration Space

Reserved

		Address Offset (Hex)
PCI Compatible Configuration Header (See **Section 17.5.1.11**, *PCI Compatible Configuration Headers* for more information.)		000 03F
PCI-Compatible Device-Specific Configuration Space (See **Section 17.5.1.12**, *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 17-122. PCI Express Extended Configuration Space

17.5.1.13.1 PCI Express Advanced Error Reporting Capability ID Register—0x100

The PCI Express advanced error reporting capability ID register is shown in **Figure 17-123**.

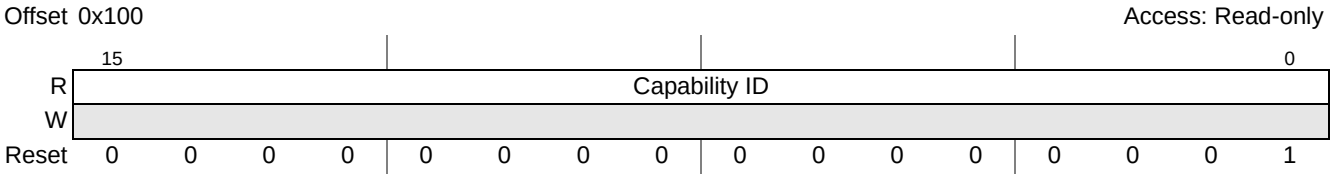


Figure 17-123. PCI Express Advanced Error Reporting Capability ID Register

Table 17-121. PCI Express Advanced Error Reporting Capability ID Register Field Description

Bits	Name	Description
15–0	Capability ID	Advanced error reporting capability = 0x0001

17.5.1.13.2 PCI Express Uncorrectable Error Status Register—0x104

The PCI Express uncorrectable error status register is shown in **Figure 17-124**.

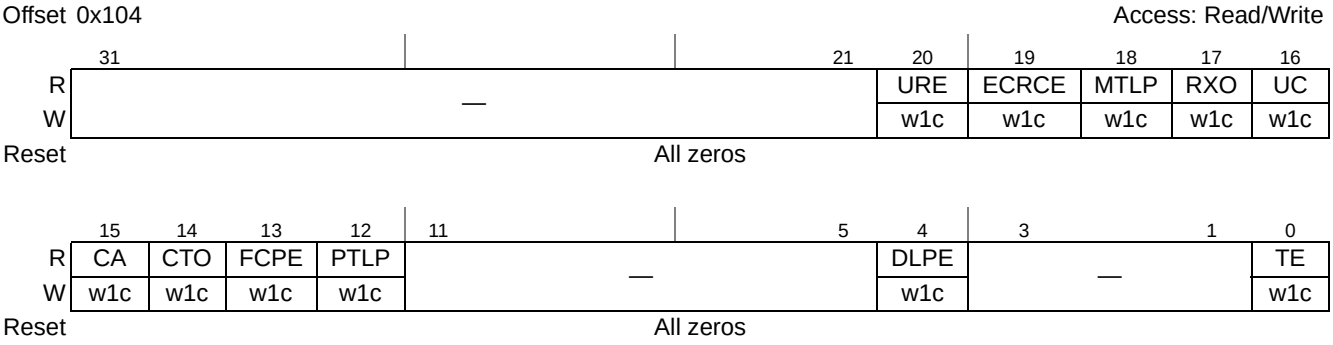


Figure 17-124. PCI Express Uncorrectable Error Status Register

Table 17-122. 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 17-122. 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.

17.5.1.13.3 PCI Express Uncorrectable Error Mask Register—0x108

The PCI Express uncorrectable error mask register is shown in **Figure 17-125**.

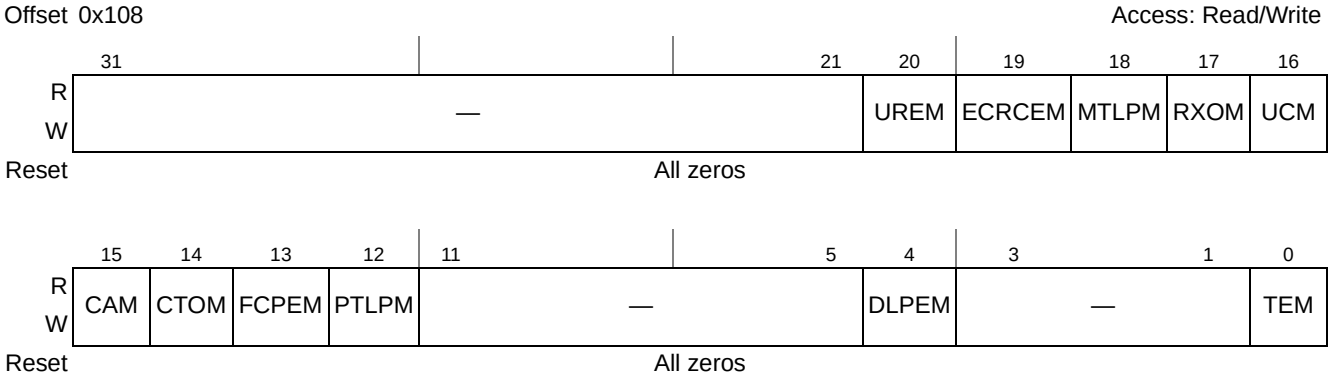


Figure 17-125. PCI Express Uncorrectable Error Mask Register

Table 17-123. 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.

17.5.1.13.4 PCI Express Uncorrectable Error Severity Register—0x10C

The PCI Express uncorrectable error severity register is shown in **Figure 17-126**.

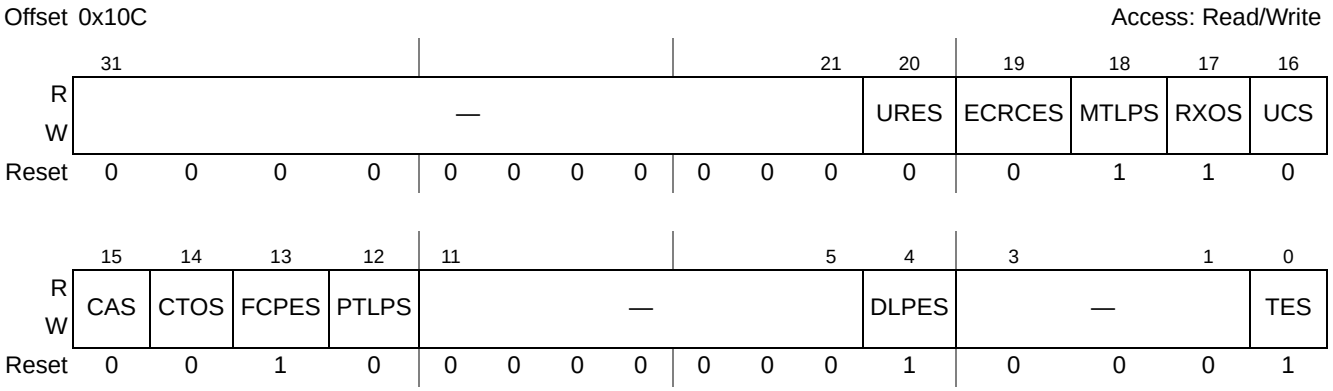


Figure 17-126. PCI Express Uncorrectable Error Severity Register

Table 17-124. 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.

17.5.1.13.5 PCI Express Correctable Error Status Register—0x110

The PCI Express correctable error status register is shown in **Figure 17-127**.

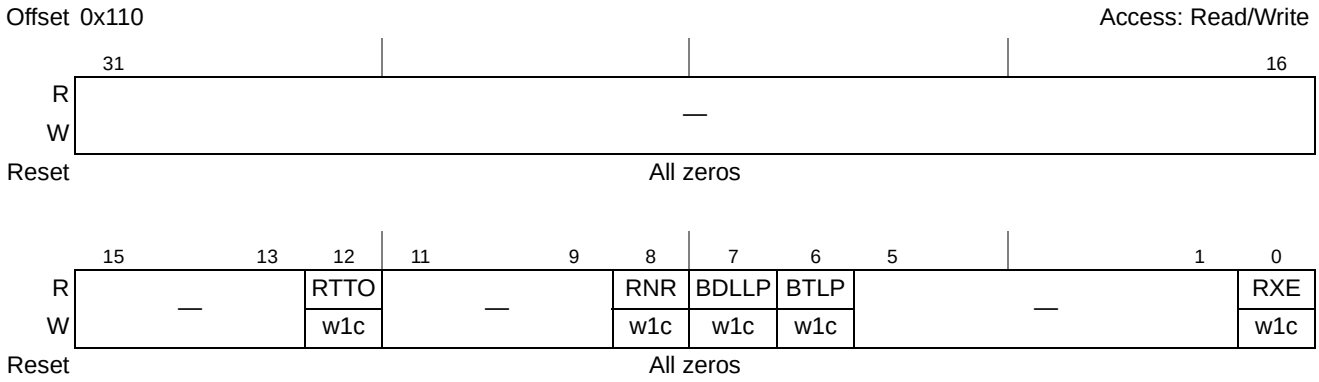


Figure 17-127. PCI Express Correctable Error Status Register

Table 17-125. 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

Note: After training and before initiating any transactions toward the PCI Express controller, always clear the RXE bit.

17.5.1.13.6 PCI Express Correctable Error Mask Register—0x114

The PCI Express correctable error mask register is shown in **Figure 17-128**.

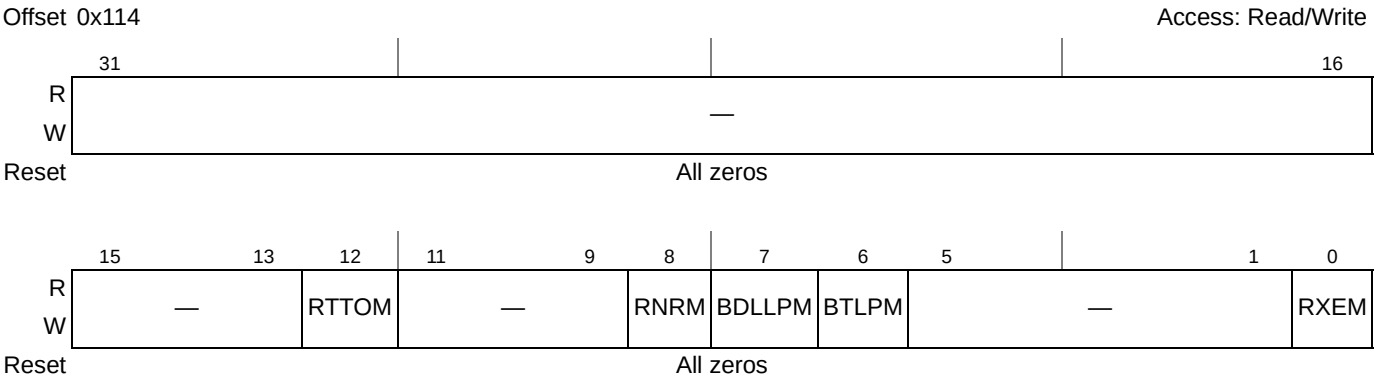


Figure 17-128. PCI Express Correctable Error Mask Register

Table 17-126. 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

17.5.1.13.7 PCI Express Advanced Error Capabilities and Control Register—0x118

The PCI Express advanced error capabilities and control register is shown in **Figure 17-129**.

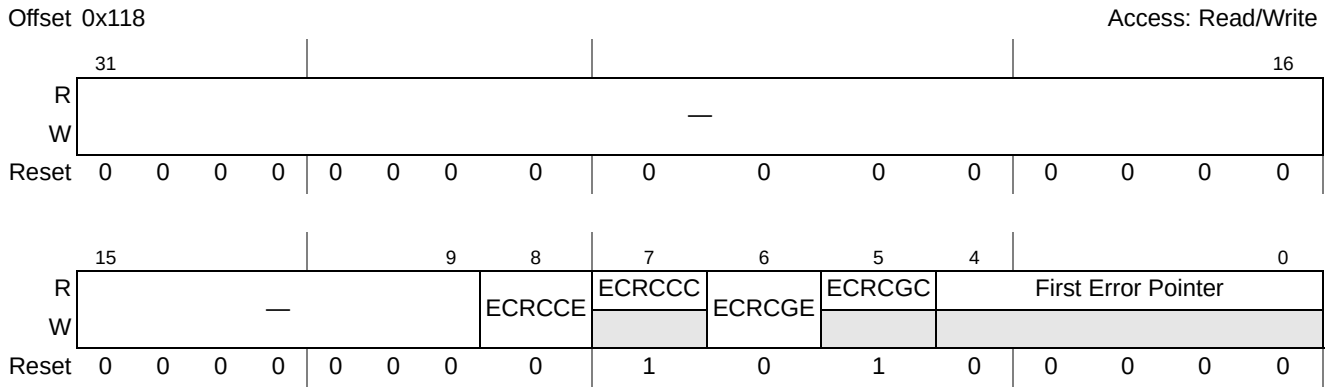


Figure 17-129. PCI Express Advanced Error Capabilities and Control Register

Table 17-127. 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.

17.5.1.13.8 PCI Express Header Log Register—0x11C–0x12B

The PCI Express header log register is shown in **Figure 17-130**.

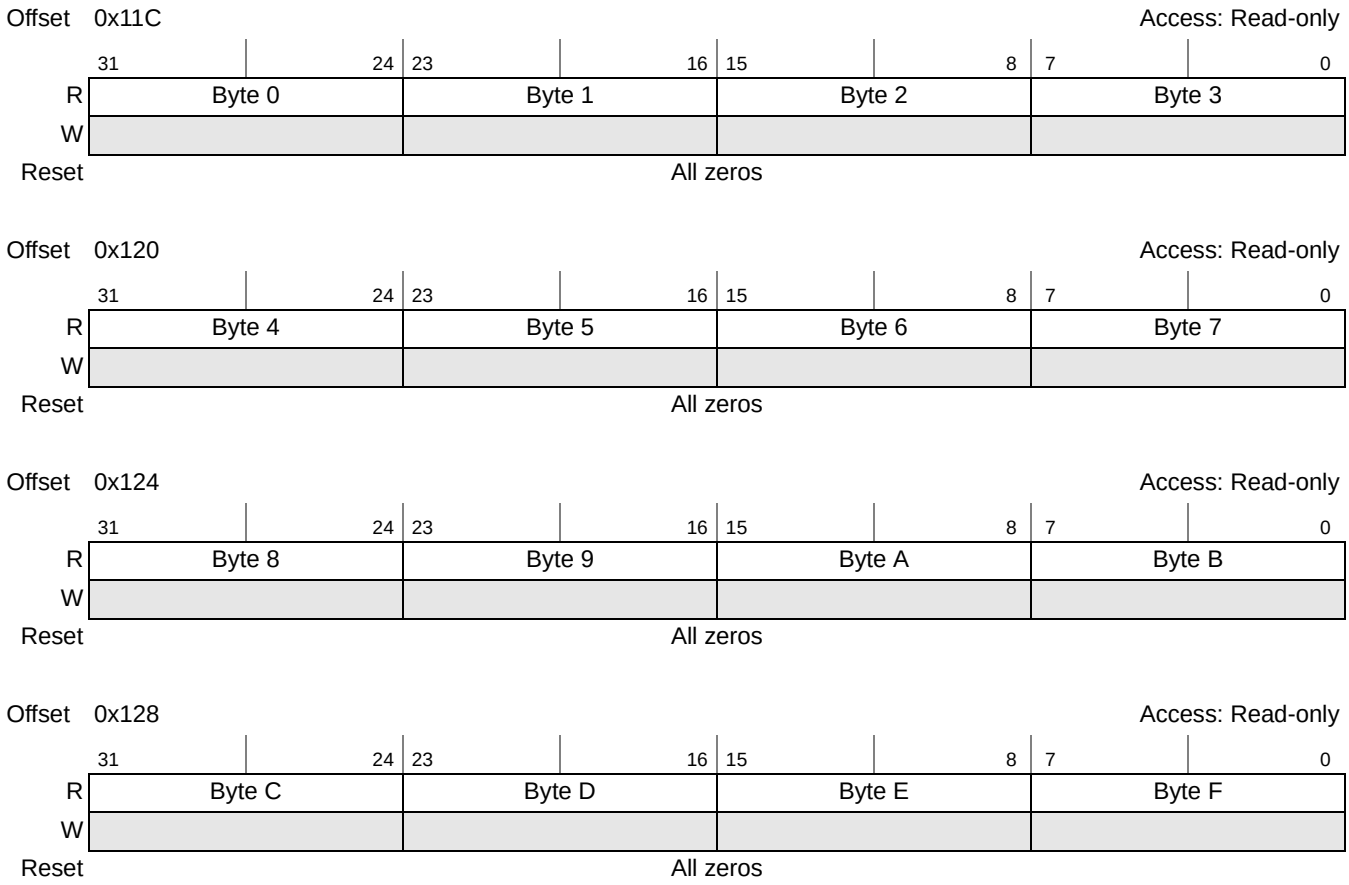


Figure 17-130. PCI Express Header Log Register

Table 17-128. PCI Express Header Log Register Field Description

Bits	Name	Description
127–0	Header Log	Header of TLP associated with error.

17.5.1.13.9 PCI Express Root Error Command Register—0x12C

The PCI Express root error command register is shown in **Figure 17-131**.

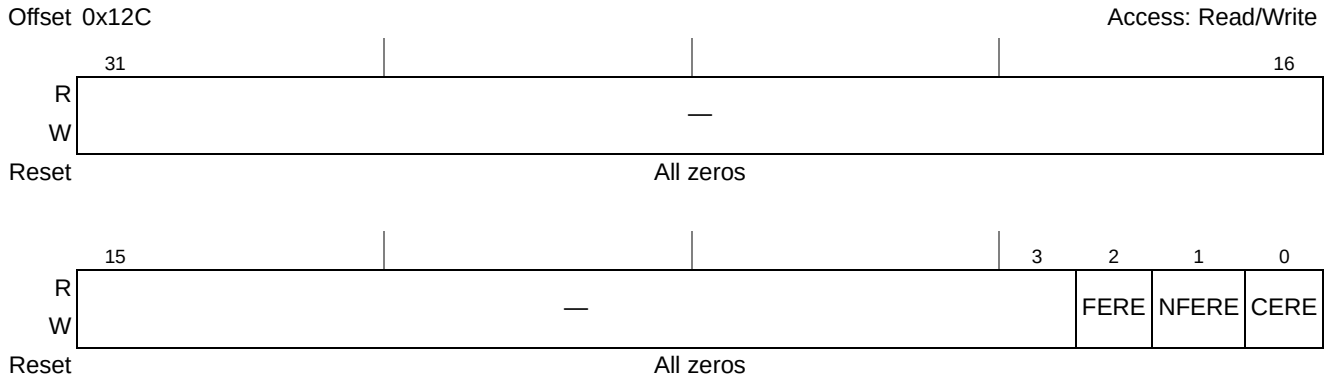


Figure 17-131. PCI Express Root Error Command Register

Table 17-129. 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

17.5.1.13.10 PCI Express Root Error Status Register—0x130

The PCI Express root error status register is shown in **Figure 17-132**.

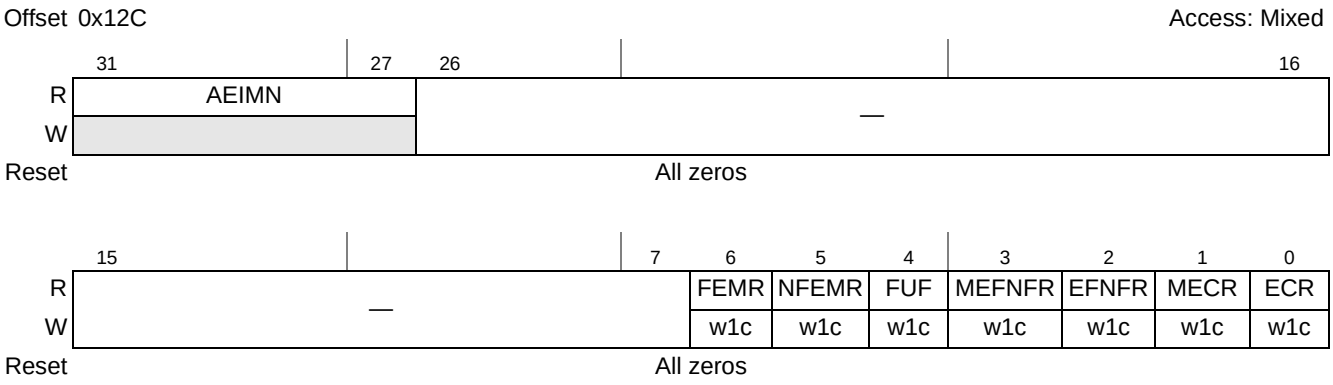


Figure 17-132. PCI Express Root Error Command Register

Table 17-130. 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.

17.5.1.13.11 PCI Express Correctable Error Source ID Register—0x134

The PCI Express correctable error source ID register is shown in **Figure 17-133**.

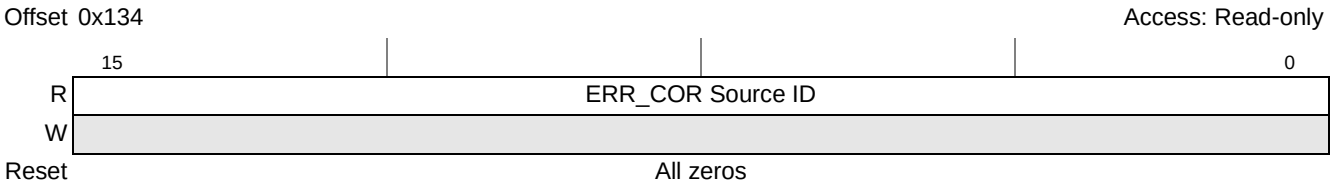


Figure 17-133. PCI Express Correctable Error Source ID Register

Table 17-131. 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.

17.5.1.13.12 PCI Express Error Source ID Register—0x136

The PCI Express error source ID register is shown in **Figure 17-134**.

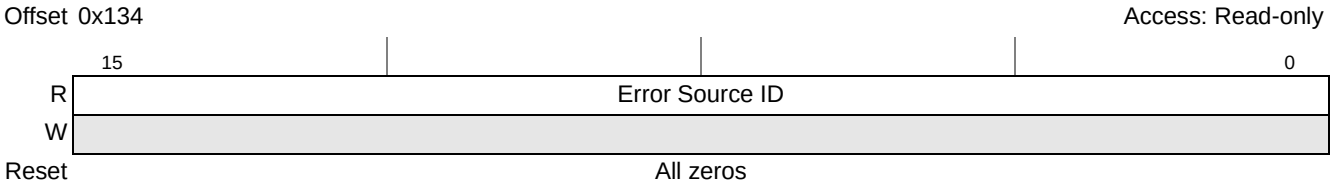


Figure 17-134. PCI Express Correctable Error Source ID Register

Table 17-132. PCI Express Correctable Error Source ID Register Field Description

Bits	Name	Description
15–0	Error Source ID	ERR_FATAL/NONFATAL source ID

17.5.1.13.13 LTSSM State Status Register—0x404

The PCI Express link training and status state machine (LTSSM) state status register, shown in **Figure 17-135**, provides detailed information about link training status. This register is useful for debugging link training failures.

Note: The Status Code in PEX_LTSSM_STAT changes while reacting to the link status. Therefore, software may need to read PEX_LTSSM_STAT multiple times to ensure the value has stabilized.

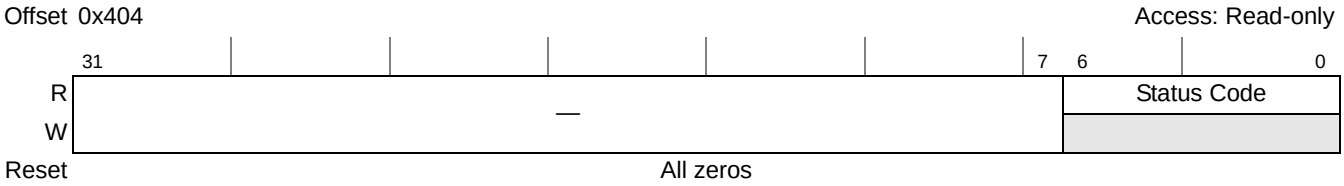


Figure 17-135. PCI Express LTSSM State Status Register (PEX_LTSSM_STAT)

The fields of the PCI Express LTSSM state status register are described in **Table 17-133**.

Table 17-133. PEX_LTSSM_STAT Field Descriptions

Bits	Name	Description
31–7	—	Reserved
6–0	Status code	Status code. See **Table 17-134** for encodings.

Table 17-134 provides the encodings for the status code field of the PEX_LTSSM_STAT register.

Table 17-134. 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)

Table 17-134. PEX_LTSSM_STAT Status Codes (Continued)

Status Code (Hex)	LTSSM State Description	Status Code (Hex)	LTSSM State Description
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)
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

17.5.1.13.14 PCI Express Controller Core Clock Ratio Register—0x440

The PCI Express controller core clock ratio register, shown in **Figure 17-136**, 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.

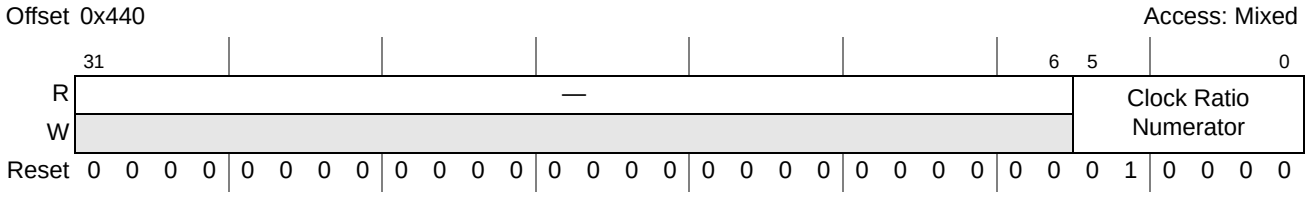


Figure 17-136. PCI Express IP Block Core Clock Ratio Register (PEX_GCLK_RATIO)

The fields of the PCI Express IP block core clock ratio register are described in **Table 17-135**.

Table 17-135. 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.

17.5.1.13.15 PCI Express Power Management Timer Register—0x450

The PCI Express power management timer register, shown in **Figure 17-137**, is used to program the time-in values for entering L0s and L1 power management states.

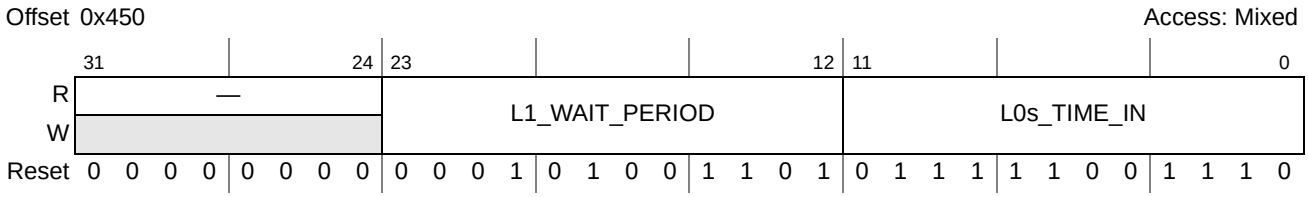


Figure 17-137. PCI Express Power Management Timer Register (PEX_PM_TIMER)

The fields of the PCI Express power management timer register are described in **Table 17-136**.

Table 17-136. 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.

17.5.1.13.16 PCI Express PME Time-Out Register (EP-Mode Only)—0x454

The PCI Express PME time-out register, shown in **Figure 17-138**, 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.

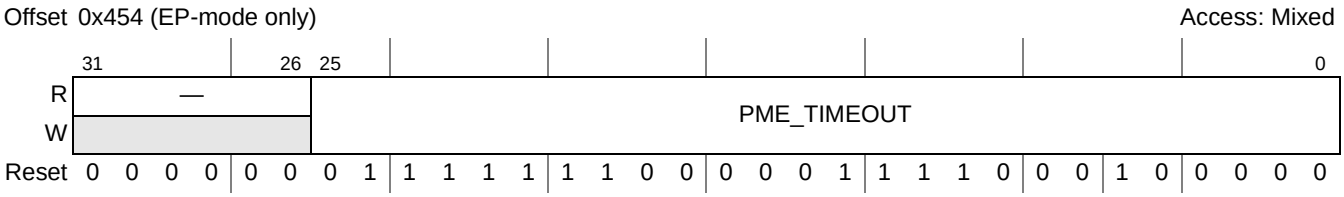


Figure 17-138. PCI Express PME Time-Out Register (PEX_PME_TIMEOUT)

The fields of the PCI Express PME time-out register are described in **Table 17-136**.

Table 17-137. 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.

17.5.1.13.17 PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478

The PCI Express subsystem vendor ID update register, shown in **Figure 17-139**, is used to set the values for the Subsystem ID and Subsystem Vendor ID registers in the Type 0 configuration header.

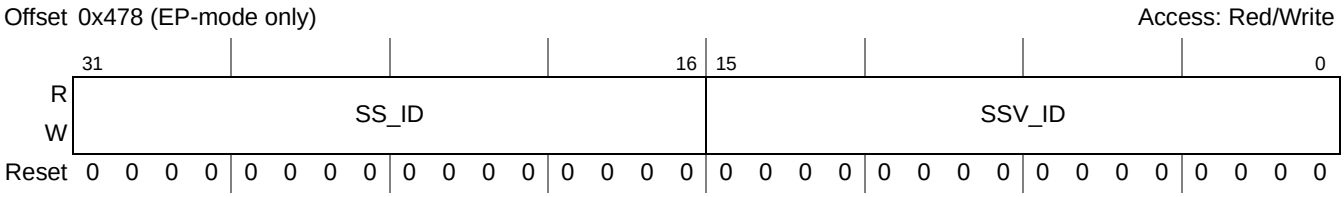


Figure 17-139. PCI Express Subsystem Vendor ID Update Register (PEX_SSVID_UPDATE)

The fields of the PCI Express subsystem vendor ID update register are described in **Table 17-138**.

Table 17-138. 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 17.5.1.13.18**, *Configuration Ready Register—0x4B0*, on page 17-127). That way, when the host begins system enumeration, the correct values will be present in the Type 0 configuration header.

17.5.1.13.18 Configuration Ready Register—0x4B0

The PCI Express configuration ready register, shown in **Figure 17-140**, 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.

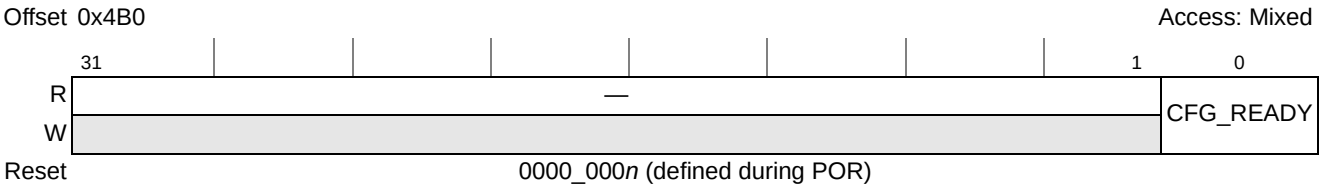


Figure 17-140. PCI Express Configuration Ready Register (PEX_CFG_READY)

The fields of the PCI Express configuration ready register are described in **Table 17-139**.

Table 17-139. 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.

17.5.1.13.19 Flow Control Update Timeout Register—0x4B8

The PCI Express flow control update timeout register, shown in **Figure 17-141**, is used to program the timeout value for update-FC DLLP reception in terms of PCI Express controller core clock cycles.

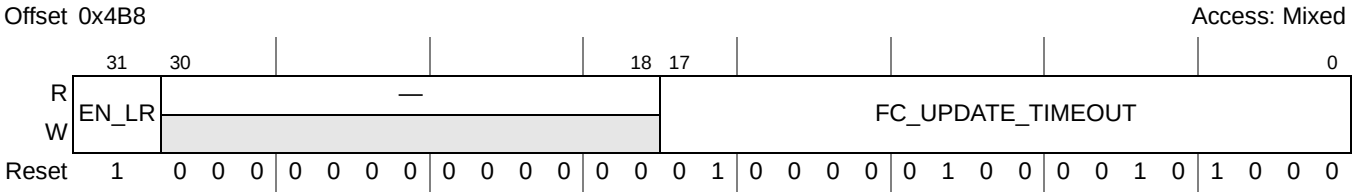


Figure 17-141. PCI Express Flow Control Update Timeout Register (PEX_FC_UPDATE_TOR)

The fields of the PCI Express flow control update timeout register are described in **Table 17-140**.

Table 17-140. PEX_FC_UPDATE_TOR Field Descriptions

Bits	Name	Description
31	EN_LR	Enable link retraining on FC update timeout error. The PCI Express protocol allows for optional link retraining by a device if FC updates are not received from the remote device within the timeout period programmed in bits [17–0] of this register. If the FC updates were lost due to link error issues, this retraining will help to restore normal operation. By default, this retraining is enabled. Since this is an optional feature, the user is allowed to disable such automatic retraining, if desired.
30–18	—	Reserved
17–0	FC_UPDATE_TIMEOUT	FC update timeout value specifies the interval (in PCI Express controller core clock cycles) to wait for the reception of consecutive FC-update DLLPs before indicating a flow-control protocol error. The value is calculated as: Time (in µsec) × PCI Express controller core clock frequency (in MHz) The maximum time value is 200 µsec; the default value (0x10428) is 200 µsec for the default clock frequency of 333 MHz.

17.5.1.13.20 Secondary Status Interrupt Mask Register (RC-Mode Only)—0x5A0

The PCI Express secondary status interrupt mask register, shown in **Figure 17-142**, can be used to disable sideband interrupt generation when error bits in the PCI Express secondary status register are set. See **Section 17.5.1.11.29**, *PCI Express Secondary Status Register—0x1E*, on page 17-89 for more information. By default, interrupt generation due to secondary status errors is disabled.

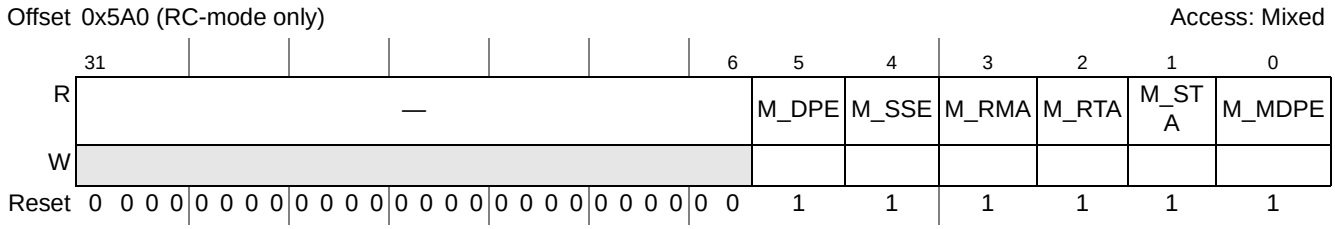


Figure 17-142. PCI Express PCI Interrupt Mask Register (PEX_SS_INTR_MASK)

The fields of the PCI Express secondary status interrupt mask register are described in **Table 17-141**.

Table 17-141. 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

17.5.1.13.21 Gen2 Control Register—0x640

The PEX Gen2 control register, shown in **Figure 17-143**, can be used to control the MAC in Gen2 device.

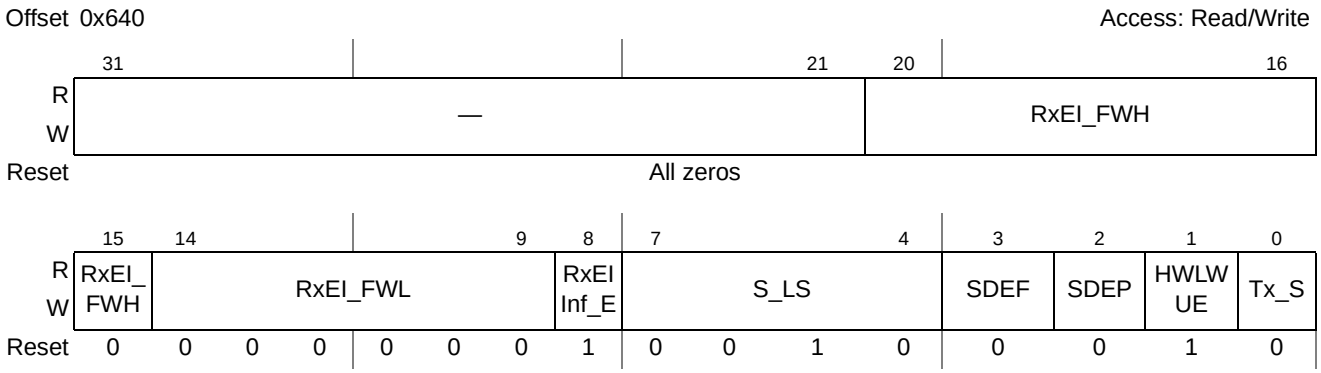


Figure 17-143. PEX Gen2 Control Register (PEX_GEN2_CONTROL)

The fields of the PEX Gen2 control register are described in **Table 17-142**.

Table 17-142. 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

18 Common Public Radio Interface (CPRI) Complex

The CPRI complex enables communication among radio devices (REC or RE) over a CPRI bus. The CPRI subsystems located at the baseband processing side are called Radio Equipment Control (REC) and the subsystems located at the radio module are called Radio Equipment (RE). The subsystems (RE and REC) are also called nodes. A radio basestation must have at least two nodes, with at least one of each type (RE and REC). RE nodes can be local or remote.The nodes are connected by one or more interface links, where each link connects two ports which have asymmetrical functions and roles: a master port and a slave port. At least one REC node in a radio basestation must have at least one master port with optional ports that can be slave or master. Typically, a basestation has one master port and one slave port.

The CPRI complex in the MSC8157E is designed to support the CPRI version v.4.1 specification and can be configured to support several air interface standards, including WiMAX, LTE, and WCDMA. Each complex supports up to six CPRI links with each link configurable as a master or slave port.

Each CPRI link supports three types of service access points (SAP):

- IQ samples for antenna transferred through the SAP IQ Interface
- CPRI frames synchronized by the SAP synchronization interface
- CPRI link control and management (C&M) data transferred between SAPs in both CPRI master and slave ports. The fast C&M channel follows the same Ethernet standards as a slow C&M channel (HDLC).

Figure 18-1 shows the CPRI complex block diagram and external interfaces. The CPRI Framer modules construct CPRI frame structures and manage transmission of CPRI frames to a serial link using the ten-bit interface (TBI) protocol. The CPRI Framer also extracts the data fields from the received CPRI frames, and the C&M data is transferred to the system memory via the Ethernet HDLC integrated engines. The receive and transmit IQ data is transferred by the CPRI DMA to/from antenna carrier buffers in the main memory. The VSS, HDLC, and fast Ethernet also transfer the data to buffers in the system memory using DMA. Auxiliary interfaces are required to support chain topologies.

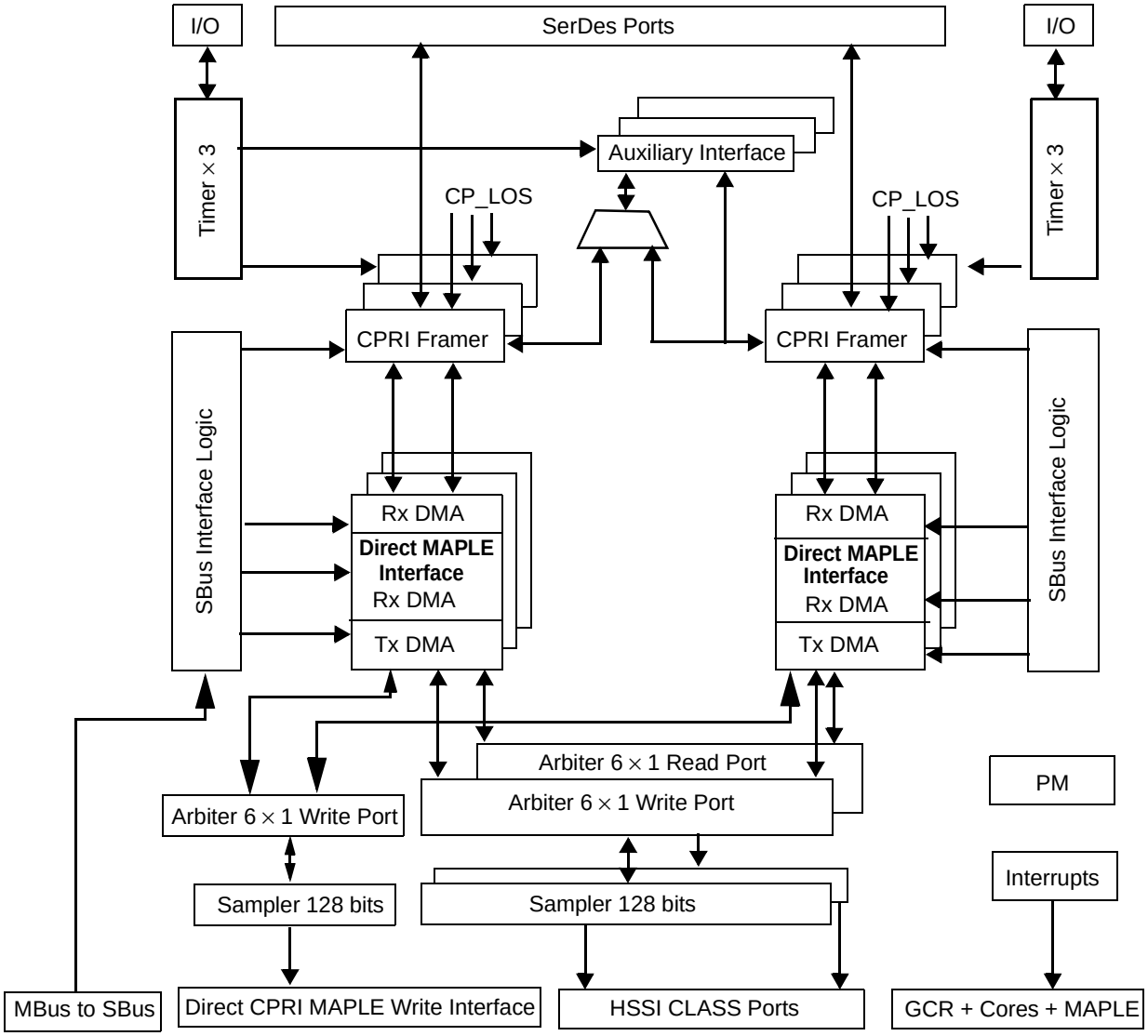


Figure 18-1. CPRI Complex Block Diagram

18.1 Features

The CPRI complex has the following features:

- Supports the following CPRI link rates:
  - 1228.8 Mbps
  - 2457.6 Mbps
  - 3072.0 Mbps
  - 4915.2 Maps
  - 6144.0 Mbps

Note: All the CPRI units must be configured to the same link rate

- Supports up to 6 lanes
- Supports the following topologies
  - Single point-to-point link between one REC and one RE (figure 3 in the CPRI Specification V4.1)
  - Multiple point-to-point link between one REC and one RE (Figure 4 in the CPRI Specification V4.1)
  - Multiple point-to-point links between one REC and several REs—Star topology (Figure 5 in the CPRI Specification V4.1)
  - Chain topology (figure 5A in the CPRI Specification V4.1)
  - Tree topology (figure 5B in the CPRI Specification V4.1)
  - Multiple point-to-point links between several RECs and one RE (figure 5D in the CPRI Specification V4.1)
  - Chain topology of multiple RECs (figure 5E in the CPRI specification V4.1)
- Supports the following mapping methods:
  - LTE mapping method 1 (IQ sample based) and method 3 (Backward compatible)
  - WiMAX mapping method 3 (Backward compatible).
  - WCDMA mapping method 3 (Backward compatible)
- Supports packed position and flexible position of each AxC container
- Each port supports up to 24 antenna carriers.
- All carriers of a CPRI unit should run with the same sampling rate. In order to support different radio interfaces running in different sampling rates on the same link, auxiliary interface must be used.
- Sample width supported: 8- (with double oversampling), 15- and 16-bit sample width.
- All the following LTE sampling rates:
  - 1.92 MHz (channel BW 1.25 MHz).
  - 3.84 MHz (channel BW 2.5 MHz).
  - 7.68 MHz (channel BW 5 MHz).
  - 15.36 MHz (channel BW 10 MHz).
  - 23.04 MHz (channel BW 15 MHz).
  - 30.72 MHz (channel BW 20 MHz).

Note: All the following LTE sampling rates: 1.92 MHz, 3.84 MHz, …, 1.92 * k MHz, …, 30.72 MHz (that is, any multiple of 1.92 MHz from 1.92 MHz through 30.72 MHz (inclusive) is supported

- Different WiMAX sampling rates like:
  - 4 MHz.
  - 5.6 MHz.
  - 8 MHz.
  - 10 MHz.
  - 11.2 MHz.
- Transparent mode (all the CPRI frames transfer to a single buffer in the main memory)
- Fast Control and Management MAC (Ethernet):
  - Three different MAC address filters:

  - Unicast filtering.
  - Multicast filtering (up to 32 different addresses).
  - Broadcast filtering.
- — 4b5b decoding and encoding
- — CRC.

- Slow Control and Management MAC (HDLC)
- Vendor Specific Data (VSS)
- L1 InBand.
- Link start-up and speed negotiation by software.
- Accurate delay measurement and calibration integrated.
- 32-bit Bus interface for register and memory configuration.
- Interrupts. Edge and level.
- Scrambling (optional) for non-daisy-chain topologies.

18.2 Modes of Operation

- Normal operation mode
- Transparent mode

18.3 Functional Description

18.3.1 CPRI Framer

The CPRI complex contains six identical CPRI Framers. The CPRI Framers are built of two main parts: receive and transmit. The transmit part constructs the CPRI frame structure and handles transmission of CPRI frames to the serial link. The receive part of the CPRI Framer extracts data fields from the received CPRI frames and transfers it to the CPRI DMA module. The CPRI Framer supports packed position and flexible position of the AxC container in the CPRI basic frame as defined in the CPRI standard. The CPRI Framer provides synchronization service access points (SAPs), using hyper frame number (HFN) and radio frame number (BFN) fields to provide timing information.

The CPRI framer handles the transmission and reception of all IQ data, C&M (Ethernet, HDLC) and VSS data.

In the transmit direction the framer constructs the CPRI frame structure, executes a 8b/10b encoding and parallel to serial conversion.

In the receive direction the framer executes serial to parallel conversion, comma/byte alignment, 8b/10b decoding, phase alignment and extracts the data from the frame.

Each instantiation of the CPRI framer is designed to provide a complete set of features required to support current and future standard evolutions. The principle followed by the development

process has been to keep it simple, optimized, and complete. This design supports the following basestation standards:

- UTRA-FDD Support (UMTS/WCDMA).
- E-UTRA Support (3GPP LTE)
- WiMAX Support (**IEEE** 802.16 standard)

Each framer contains the following functional blocks:

- CPRI Transmitter
- CPRI Receiver
- CPRI MAP Transmitter
- CPRI MAP Receiver
- Fast C&M interface (Ethernet MAC 10/100 MHz)
- Slow C&M interface (HDLC MAC)
- C&M Channels
  - VSS
  - L1 Inband
  - Alarms
- Register interface
- Delay measurement and calibration

The CPRI standard is based on a frame structure built of 10 ms radio frames. Each radio frame is built of 150 hyperframes and each hyperframe is built of 256 basic frames. There are 16 words in each basic frame; the first is a control word and the others are IQ data. The size of the words depends on the link rate as defined in the standard (at 1.228 Gbps link rate each word is 16 bits and at 6.144 Gbps link rate each word is 80 bits).

The following sections describe the framer components that process the frames:

- **Section 18.3.1.1** provides the basic description of the Framer Transmitter.
- **Section 18.3.1.2** provides the basic description of the Framer Receiver.
- **Section 18.3.1.3** describes the auto negotiation process.
- **Section 18.3.1.4** describes the way the IQ data is mapped in the IQ data words.
- **Section 18.3.1.5** describes the transmission and reception of the control words.

18.3.1.1 CPRI Framer Transmitter

Figure 18-2 shows the functions of the CPRI TX module inside the CPRI Framer. After the software driver initializes the system and enables the CPRI, the TX framer starts to negotiate versus the RE or REC connected to it (autonegotiation flow is described in **Section 18.3.1.3**). At the end of the autonegotiation phase, the user can enable the DMA and the DMA begins to read the data into its Memory (IQ, Ethernet, HDLC, and VSS).

As long as the Framer is enabled, it reads the IQ samples, Ethernet data, HDLC data, and other control data from the DMA memory and multiplexes them together to form the CPRI frame structure. If the DMA does not contain any data to be transmitted, the Framer keeps sending “0” instead of IQ, Ethernet, HDLC, and VSS. All transmit data from the Framer goes through the 8b/10 encoder and parallel to the serial module towards the physical interface.

It is possible to scramble the CPRI frames (as specified in the CPRI v4.1 specification) for rates greater than 3.072 Mbps for non-daisy-chain topologies. The scrambling is enabled using the CPRIn_TX_PROT_VER register with the scrambler seed configured in the CPRIn_TX_SCR_SEED register.

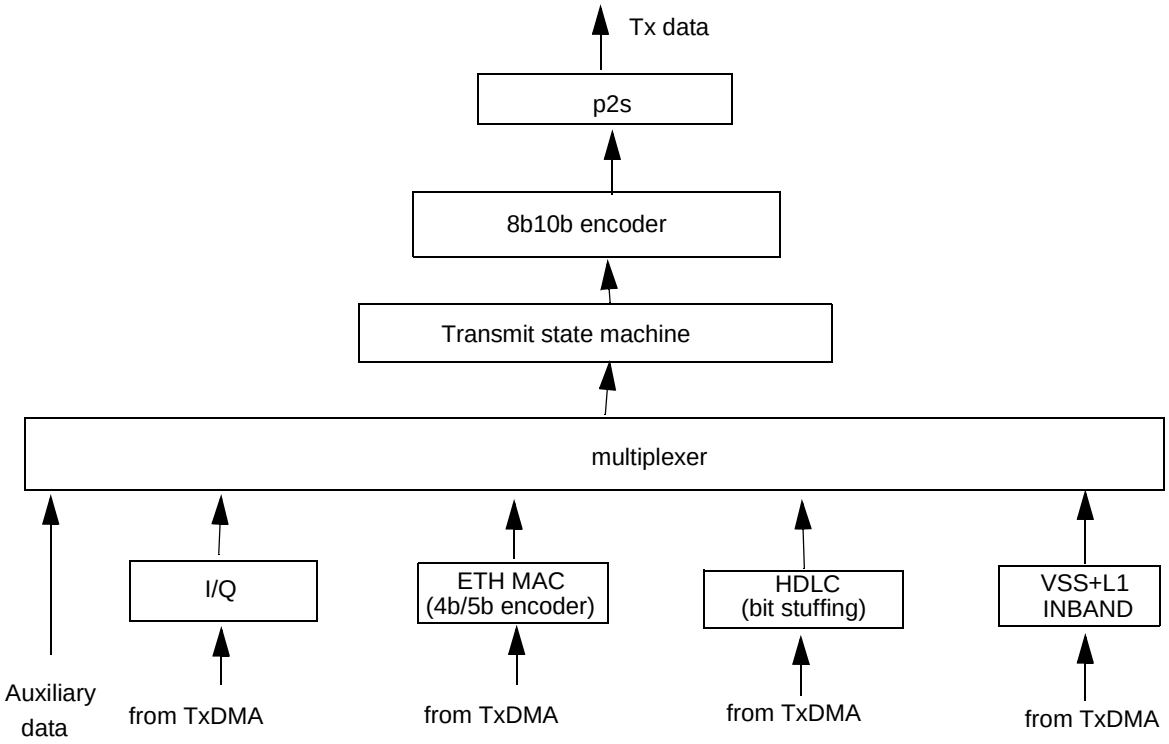


Figure 18-2. CPRI Framer TX Path Architecture

18.3.1.2 CPRI Framer Receiver

Figure 18-3 shows the functions of the CPRI framer RX module.

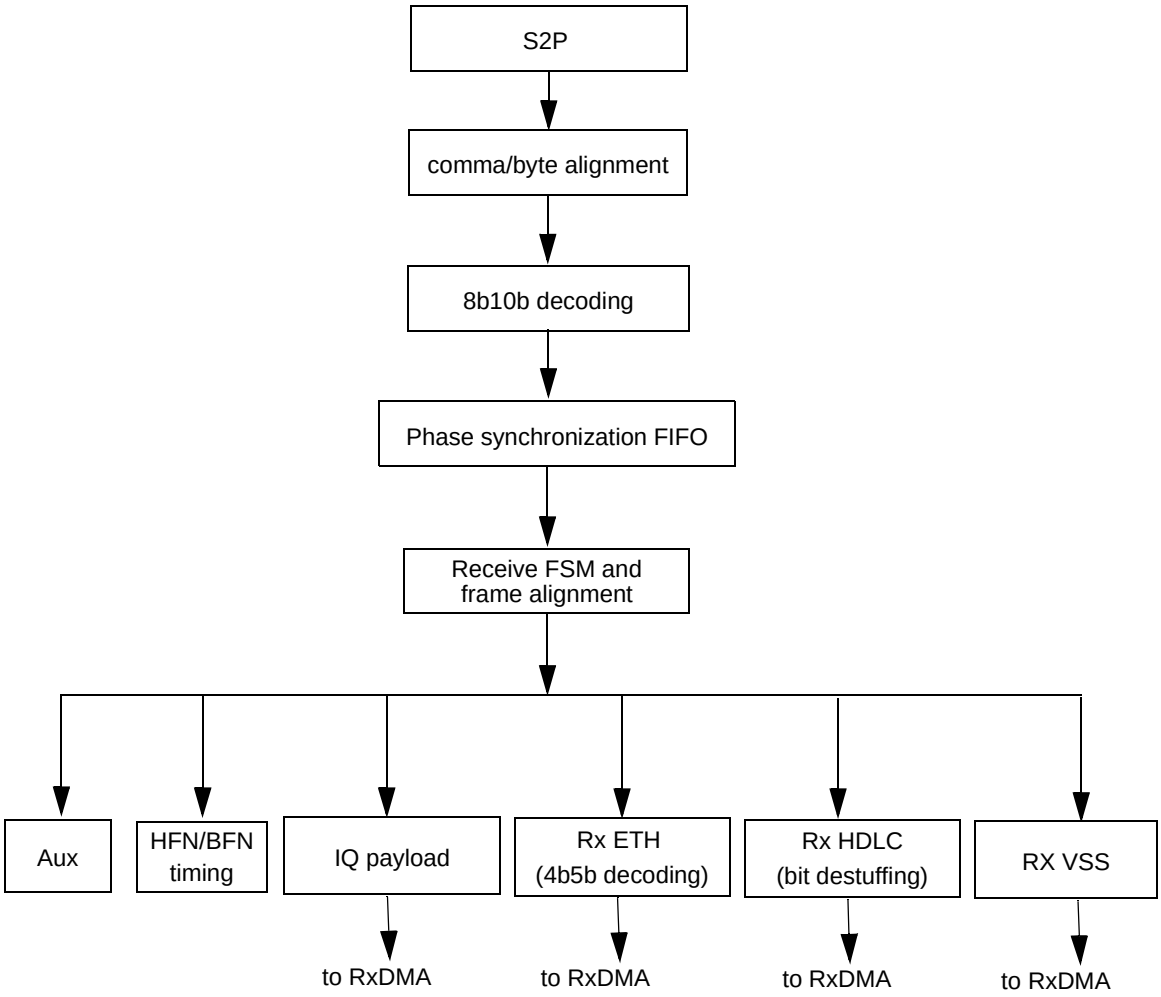


Figure 18-3. CPRI RX Module Path Architecture

In the receive path, the role of the SerDes is to recover the clock from the received data and both clock and data are then moved towards the Framer. The data is first converted from serial to parallel followed by comma alignment and 8b10b decoding. Then, the framer extracts the IQ, Ethernet, HDLC, and VSS from the received frames as follows:

- IQ data are forwarded to the CPRIn_RX_MAP module for extraction of IQ data streams. This data is written through the RX IQ DMA to the configured system memories.
- Ethernet data is extracted according to the pointer p (Z.194.0). Pointer p value is recovered (new pointer accepted when 4 consecutive equal values are received) and used to identify control words carrying Ethernet data. These control words are 4b5b decoded and forwarded to the Ethernet termination module. The Ethernet packets are written through the RX Ethernet DMA to the relevant system memory location.

- The HDLC bit rate information is recovered (new rate accepted when 4 consecutive equal values are received) and used to identify control words carrying HDLC data. These control words are bit-destuffed symbols and are forwarded to the HDLC termination module. The HDLC packets are written through the RX HDLC DMA to the relevant system memory location.
- VSS data extraction uses the recovered pointer p and it is used to identify control words carrying VSS. The VSS data is written through the RX VSS DMA to the relevant system memory location.

The Recovered Receive Clock (CDR clock) that is used to sample the receive data from the SerDes must be frequency synchronous with the internal Framer Clock (TBI clock divided by 4). If there is a difference in the frequency (due to system problem like uplink generated by using different clock frequency), the synchronization buffer can overflow or underflow and resynchronization executes automatically. The RX_FREQ_ALARM_HOLD field in the CPRIn_STATUS register can be used to monitor whether the CDR Clock is synchronous to the Framer Clock. If a frequency difference of more than ±4 clock cycles is detected, the alarm bit is flagged. The alarm bit is cleared when reading the CPRIn_STATUS register. If the alarm is reset, it requires a new drift of 4 cycles to set the alarm again. After clock synchronization, the CPRI frame detection is performed.

As specified in CPRI v4.1, auto detection of scrambled data is performed for rates above 3.072 Gbps based upon the received number in the protocol version field (Z.2.0). The scrambler seed is automatically extracted and is made available through the register CPRIn_RX_SCR_SEED. In case data is scrambled, it automatically descrambles before further processing.

A state machine performs CPRI hyper frame alignment in the receive direction expecting the K28.5 synchronization control word always in the first basic frame of each hyperframe. The HFN and BFN alignments are recovered when 4 matching sequences are received, respectively Alignment is considered lost after 4 consecutive mismatches. RX_HFN_STATE and RX_BFN_STATE status can be read in the CPRIn_STATUS register. The recovered CPRI frame timing is available to the system via the timing output interface. The RX_STATE bit in the CPRIn_STATUS register is set after completion of the full synchronization flow as described previously.

It can as many as 6 hyperframes to achieve rx sync. 6 hyperframes = 6 * 256 (basic frames in a hyperframe) * 40 (clock cycles in a basic frame @ 6.144 Gbps line rate) Framer Clock cycles. The count starts when the first K28.5 is seen on the TBI/4 Framer Clock side of the Rx Synchronization Buffer.It takes two K28.5 (spaced correctly) to exit LOS, and it then takes another 4 K28.5s with correct spacing to enter rx sync state.

18.3.1.3 Auto Negotiation (Setup)

The Autonegotiation flow has three stages:

- Sets up the link rate between the devices.
- Sets up a common protocol version
- Sets up a common C&M channel rate if needed

The auto negotiation flow is done by software according to the flow described below. The flow is for a single CPRI link.

All the CPRI lanes must be configured to the same link rate even if they are not enabled. it means that fields RRAT_SEL and TRAT_SEL in registers LEGCR0– LJGCR0 should be configured with the same value.

The auto negotiation reset is done by writing to bit SDRST in register SRDSB2RSTCTL located outside the CPRI complex. It resets the PLL inside the SerDes which it shared between all the CPRI modules.

The auto negotiation reset has the following effects:

- Resets all the CPRI Framer registers
- Resets all the CPRI DMA operation registers (pointers, counters, and so on)
- All the state machines return to their idle state.
- All the registers are reset.
- The CPRI memories are not affected.

18.3.1.3.1 Autonegotiation (Setup) Flow

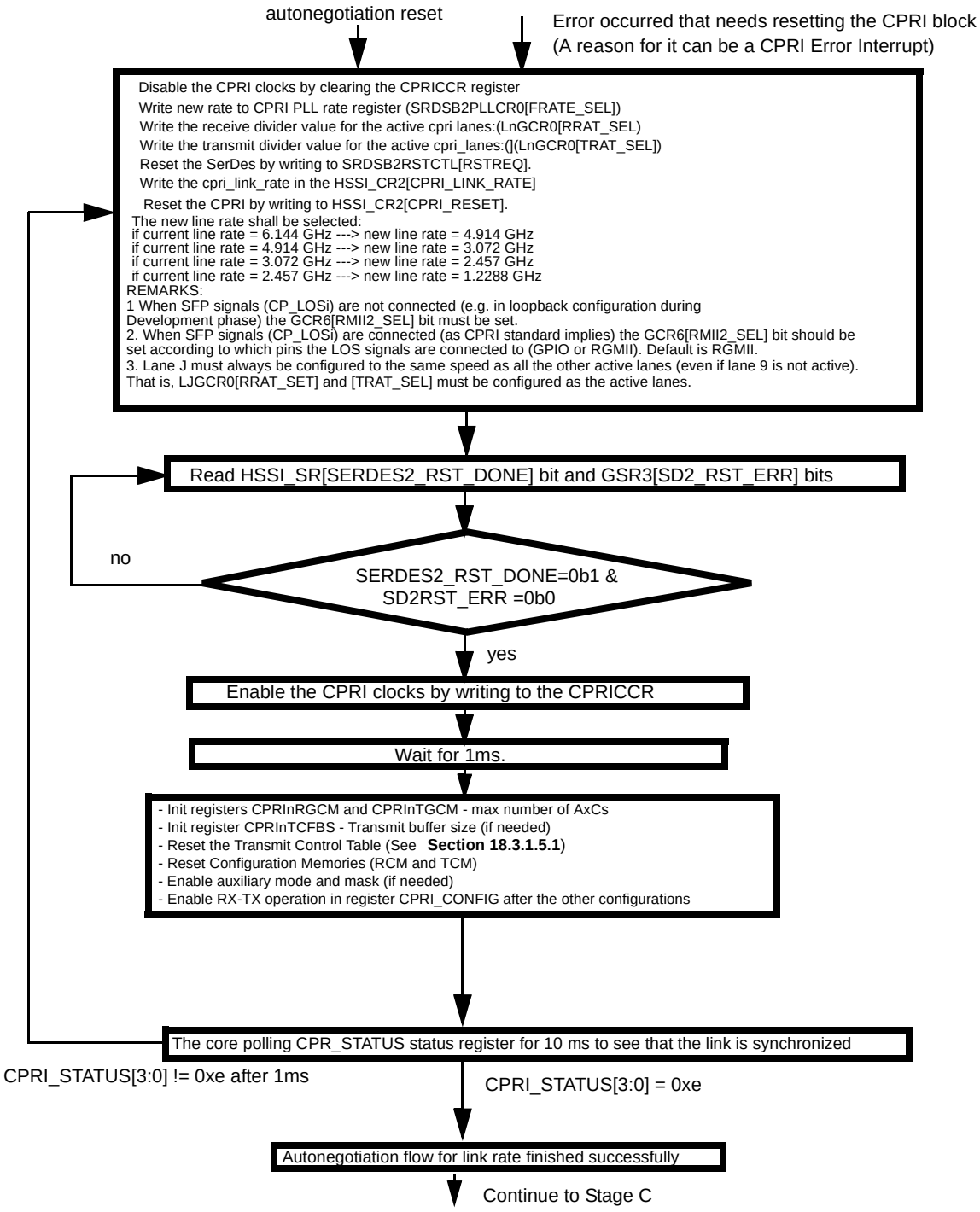


Figure 18-4. Autonegotiation Setup Flow

18.3.1.3.2 Protocol Setup (State C)

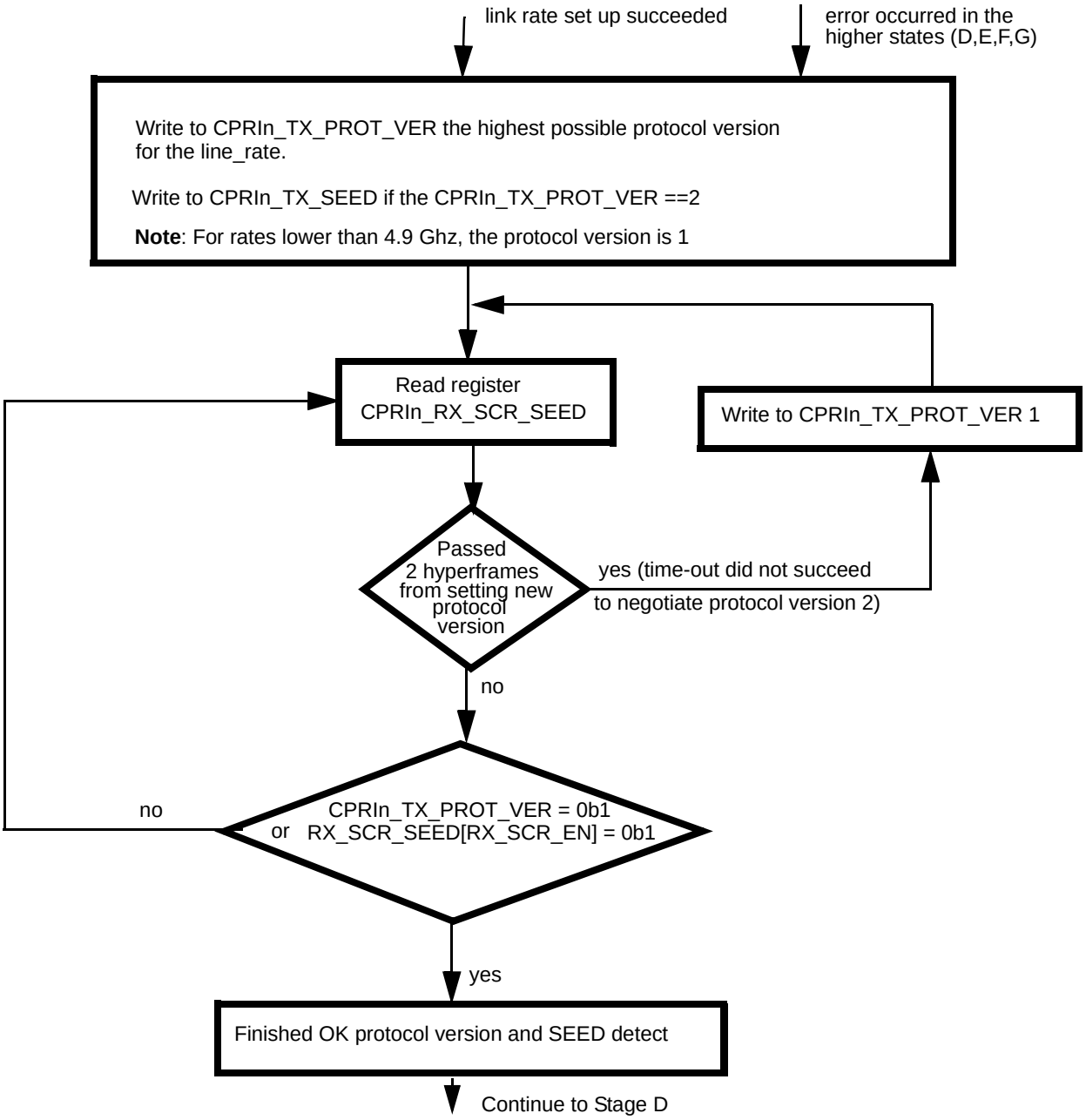


Figure 18-5. Protocol Setup

18.3.1.3.3 C&M Channel Rate Setup (State D)

During this state a common C&M channel bit rate is determined (at least one control rate)

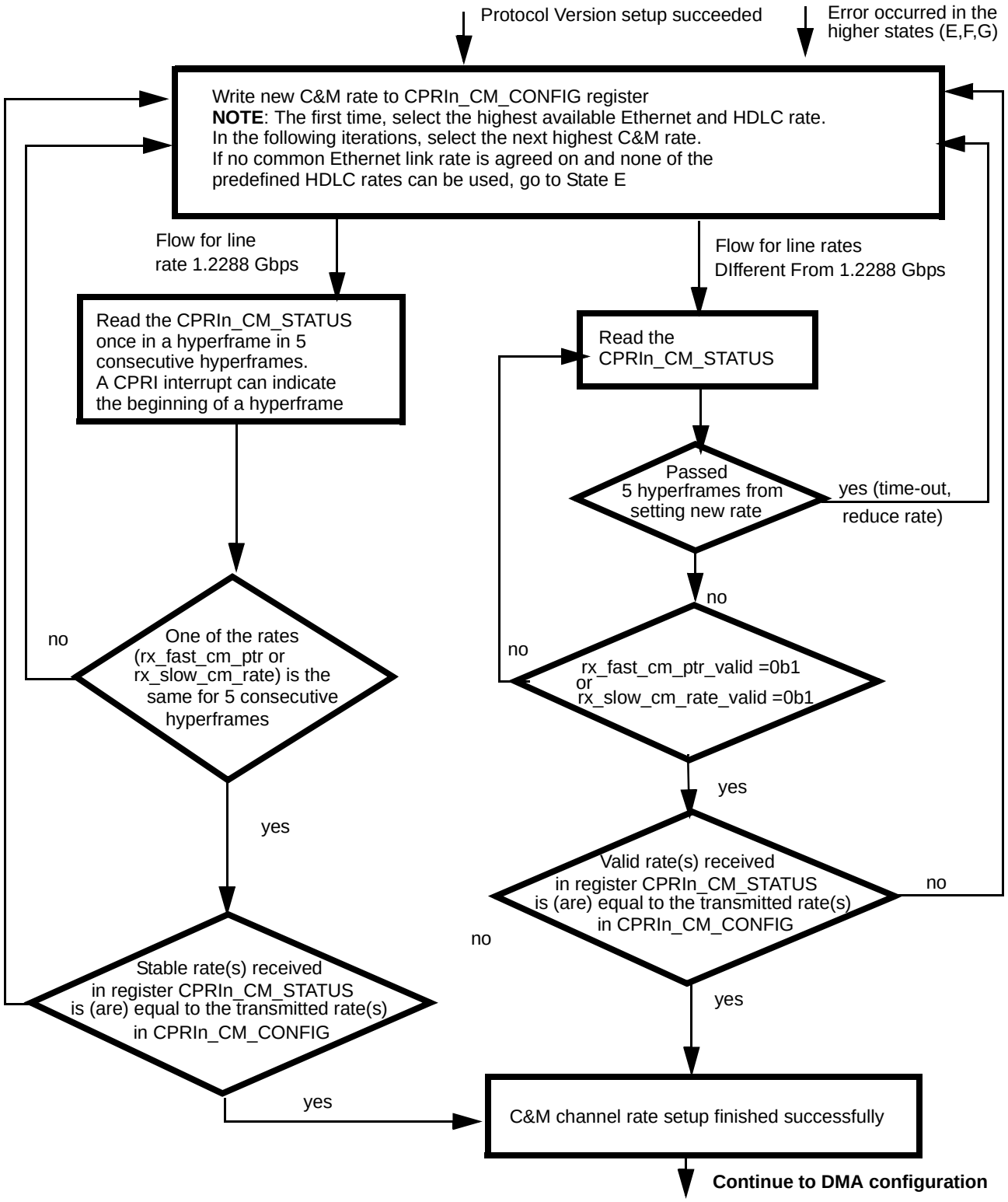


Figure 18-6. C&M Channel Rate Setup

18.3.1.3.4 DMA Configuration

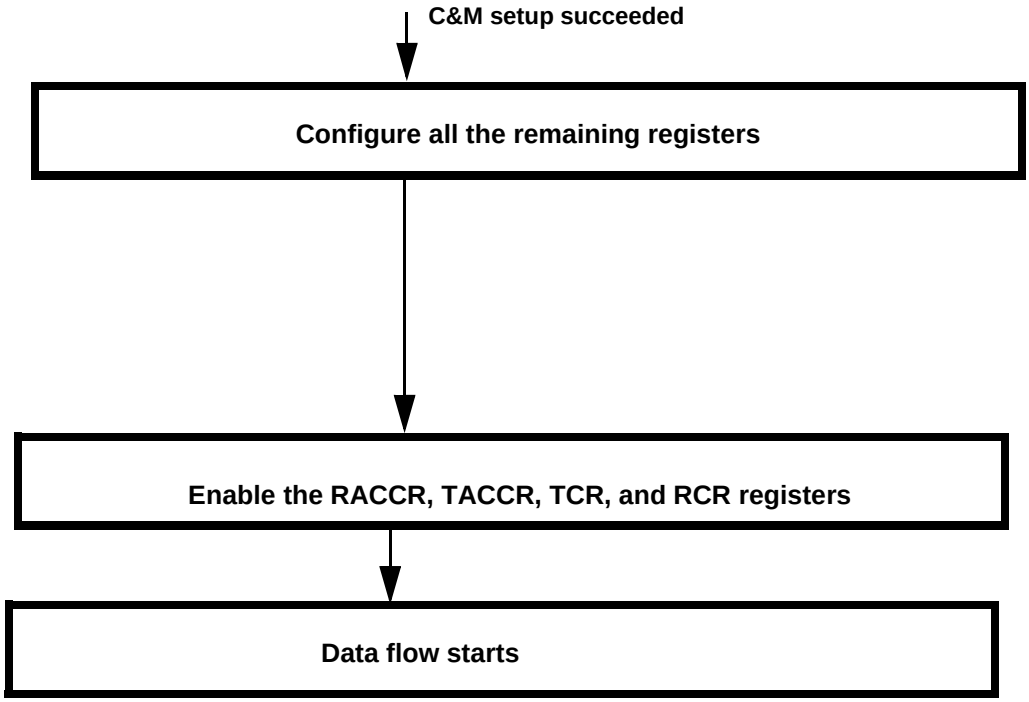


Figure 18-7. Configuring and Enabling the DMA Logic

18.3.1.3.5 State E—Interface and Vendor Specific Negotiation

During the E state, the application for the master and slave ports negotiates the CPRI usage. If a common bit rate for the Ethernet link is agreed to in state D, it is used. Otherwise, the HDLC link is used. In this state, negotiation to a HDLC bit rate that is not one of the pre-defined bit rates may occur. After the negotiation, the core sets the TX_SLOW_CM_RATE field in register CPRIn_CM_CONFIG to 0x007 to indicate to the slave port that a new HDLC bit rate is used. The characteristics of the negotiated HDLC channel is vendor-specific and is outside of the specifications defined for the CPRI.

18.3.1.4 CPRI AxC Mapping

The IQ mapping modules perform mapping of I/Q samples of the different AxCs into the CPRI frame structure. The MAP_MODE field of the CPRIn_MAP_CONFIG register selects between different mapping schemes available.

18.3.1.4.1 Basic AxC Mapping Mode

The basic mapping scheme is selected by setting MAP_MODE to 00 and applies to UMTS/LTE standards in which all the AxCs use the same sample rate. This mapping scheme follows the description given in CPRI Sections 4.2.7.2.2 and 4.2.7.2.3 Option 1 (Packed Position) in Figure #13 of the CPRI specifications.

IQ samples are mapped into Basic Frames as shown on **Figure 18-8**. The mapping is controlled by MAP_AC and MAP_N_AC fields in register CPRI Mapping Counter Configuration (CPRIn_MAP_CNT_CONFIG). For each AxC stream, IQ samples are mapped into AxC Containers, which are consecutively mapped into the IQ Data Block of the Basic Frames. The maximum number of AxC containers (NMAP) is 24. Each AxC Container contains samples only from one AxC mapping stream.

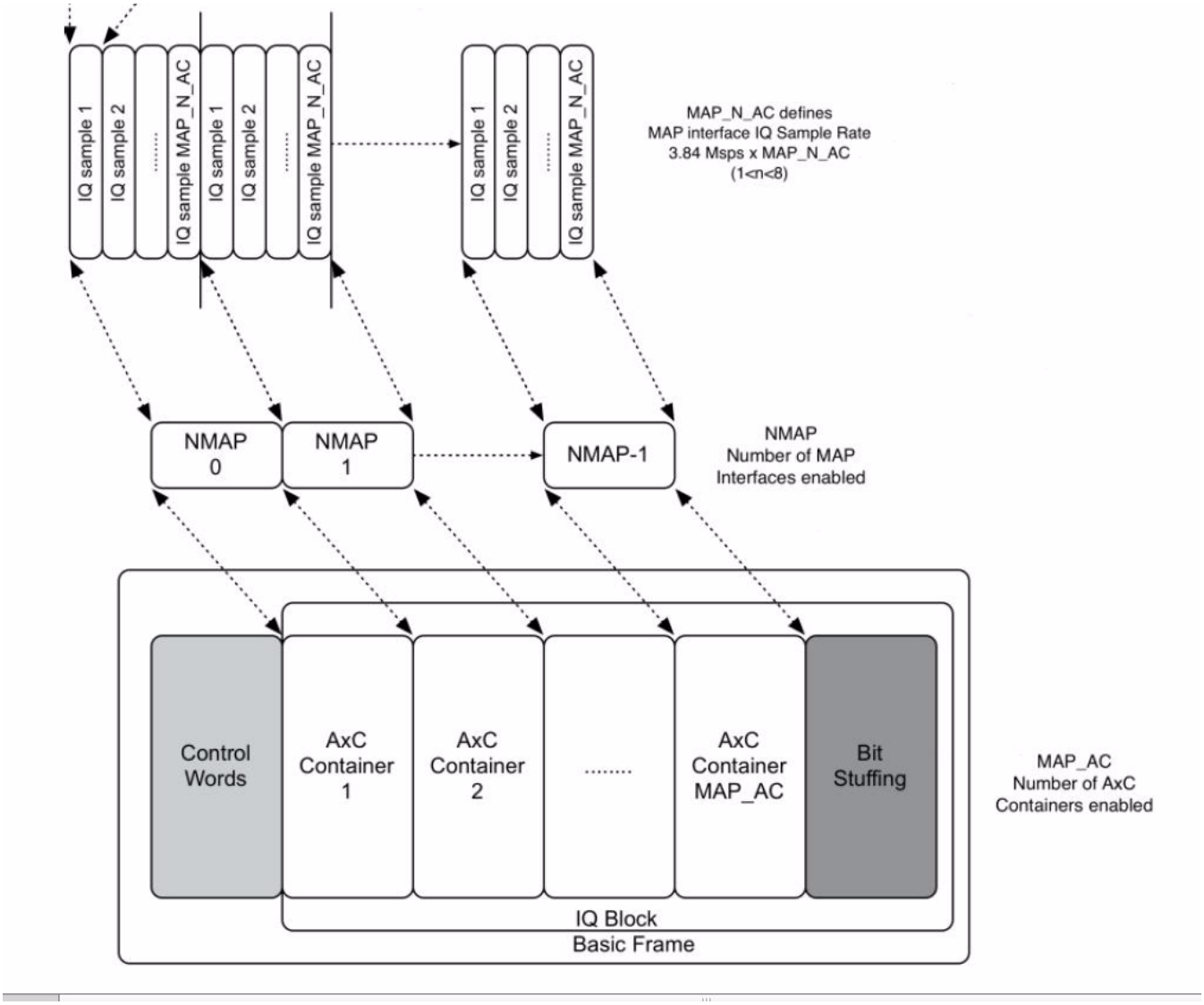


Figure 18-8. Example of Mapping of 16bits IQ samples

The number of IQ samples mapped into each AxC Container is given by the oversampling ratio MAP_N_AC. The sampling rate per AxC interface is computed in the following way:

$$\text{Sampling rate} = \text{MAP\_N\_AC} \times 3.84\ \text{MSPS}$$

The Oversampling Ratio factor is an integer value up to 8 so the maximal sampling rate, thus maximum 8 × 3.84 Msps = 30.72 Msps. **Table 18-1** shows the relation between UMTS/LTE sample rates and channel bandwidth.

Table 18-1. Relation between UMTS/LTE sample rates and RF channel BW

Sampling Rate (Msps)	Channel Bandwidth (MHz)
3.84	2.5 (LTE) & 5 (UMTS)
7.68	5 (LTE)
15.36	10 (LTE)
23.04	15 (LTE)
30.72	20 (LTE)

The number of AxCs and oversampling ratio are configured in the CPRI Mapping Counter Configuration (CPRIn_MAP_CNT_CONFIG) register. The bits in the Basic Frame that are not used for IQ samples are in the end of the Basic Frame and are padded with zeros. Locations of inactive AxCs are also padded with zeros. The active/inactive AxCs are determined by Receive AxC Control Register (CPRInRACCR) and by Transmit AxC Control Register (CPRInTACCR).

Example 18-1. AxC0 and AxC3 with No Oversampling.

2 AxCs (AxC0 and AxC3) used with no oversampling the configuration is:

CPRIn_MAP_CNT_CONFIG = 0x00000401 (MAP_AC=4, MAP_N_AC=1)

CPRInRGCM[MAP_AC] = 4

CPRIn_RACCR= 0x00000009

In case of sample width 16 bits and 2.457 Gbps link rate the Basic frame will be:

Control word, AxC0, {0x0}, {0x0}, AxC3, {0x0}, {0x0}, {0x0}, {0x0}, {0x0}, {0x0}, {0x0}, {0x0}, {0x0}, {0x0}, {0x0}

Example 18-2. AxC0 and AxC3 with Oversampling Ratio 3

2 AxCs (AxC0 and AxC3) used with oversampling ratio 3 the configuration is:

CPRIn_MAP_CNT_CONFIG = 0x00000301 (MAP_AC=4, MAP_N_AC=3)

CPRInRGCM[MAP_AC] = 4

CPRIn_RACCR= 0x00000009

In case of sample width 16 bits and 3.072 Gbps link rate the Basic frame will be:

Control word, AxC0, AxC0, AxC0, {0x0}, {0x0}, {0x0}, {0x0}, {0x0}, {0x0}, AxC3, AxC3, AxC3, {0x0}, {0x0}, {0x0}, {0x0}, {0x0}, {0x0}, {0x0}

Sample width inside the framer is programmable to 15-bit or 16-bit for each I and Q. The register CPRI Mapping Configuration (CPRIn_MAP_CONFIG) has the field MAP_15BIT_MODE that can select whether 15-bit or 16-bit samples are mapped. Note that 8-bit sample width and double oversampling (typically used in UL) can be also supported by transferring two samples at a time on the IQ mapping interface. 8-bit sample width and double oversampling can be configured in the CPRInRGCM register.

The maximum number of AxCs is limit to 24 AxCs but it is also limited by the sample width and the link rate **Table 18-2** describes the number of IQ bits in basic frame as function of link rate.

Table 18-2. Bits in IQ Data Blocks

"Rate" Configuration	Transmission Bit-rate	Number of Bits in IQ Data Block of BF
"0010"	1228.8 Mbps	240
"0100"	2457.6 Mbps	480
"0101"	3072.0 Mbps	600
"1000"	4915.2 Mbps	960
"1010"	6144 Mbps	1200

The maximum number of AXCs is determined by the formula below:

```
2 × <sample width > × MAP_N_AC × MAP_AC <= Bits in IQ Data Block
```

Table 18-3. AxC Link Capacity for 16-bit IQ sample

Channel BW LTE (Sample Rate)	2.5 MHz (3.84 Msps)	5 MHz (7.68 Msps)	10 MHz (15.36 Msps)	15 MHz (23.04 Msps)	20 MHz (30.72 Msps)
1228.8 Mbps	7	3	1	1	—
2457.6 Mbps	15	7	3	2	1
3072 Mbps	18	9	4	3	2
4915.2 Mbps	24 (limited by maximum number of supported AxCs)	15	7	5	3
6144 Mbps	24 (limited by maximum number of supported AxCs)	18	9	6	4

Table 18-3 shows AxC link capacity for sample width 16. As an example for a CPRI link at 3072.0 Mbps, there are 600 bits for IQ in each basic frame; therefore, in the case of LTE 5MHz (that is, oversampling ratio = 2), there is bandwidth to transfer nine LTE 5 MHz carriers in each Basic Frame. The IQ data takes only 2*16*2*9= 576 bits so there are always 24 bits of padding in the end of the basic frame that cannot be used in this mapping mode. This bandwidth can be also used to transfer up to 18 AxC channels at 3.84 Msps UMTS rates or up to 2 AxC at 30.72 Mbps LTE rates or anything in between, as long the available bandwidth relation is not violated. For the 15-bit sample size, the total number of required bits must be less than or equal to the following number:

```
2 × 15 × MAP_N_AC × MAP_AC <= Bits in IQ Data Block
```

Table 18-4. AxC Link Capacity for 15-bit IQ Sample

Channel BW LTE (Sample Rate)	2.5 MHz (3.84 Msps)	5 MHz (7.68 Msps)	10 MHz (15.36 Msps)	15 MHz (23.04 Msps)	20 MHz (30.72 Msps)
1228.8 Mbps	8	4	2	1	1
2457.6 Mbps	16	8	4	2	2
3072 Mbps	20	10	5	3	2
4915.2 Mbps	32	16	8	5	4

Table 18-4. AxC Link Capacity for 15-bit IQ Sample

Channel BW LTE (Sample Rate)	2.5 MHz (3.84 Msps)	5 MHz (7.68 Msps)	10 MHz (15.36 Msps)	15 MHz (23.04 Msps)	20 MHz (30.72 Msps)
6144 Mbps	40	20	10	6	5

Note: Configurations requiring more bit bandwidth than available transfer the first AxC channels correctly until all AxC segments (timeslots) in the Basic Frame are used; the last AxC channels are not transferred correctly.

18.3.1.4.2 Advanced AxC Mapping Modes

The advanced AxC mapping modes enable mapping for IQ samples of different rates and different radio standards independently on the interface using auxiliary mode. The sampling rate of all the AxCs in a CPRI unit must be the same.

These modes are enabled by using configuration tables to define mapping of IQ samples to the CPRI IQ block data areas of the basic frame. These tables are static and are configured before link start up. They should not change until the next link start up.

The advanced table based mapping is selected by setting MAP_MODE field in the CPRIn_MAP_CONFIG register, as follows:

- 01 = CPRI v4.1 Standard Section 4.2.7.2.5 Method 1: IQ sample based.
- 10 = CPRI v4.1 Standard 4.2.7.2.7 Method 3: Backward Compatible

Support of the following is available:

- Support for mapping method 3 E-UTRA [CPRI v4.1] is available. Note that in order to achieve it in the case of a 15-bit sample width, use the following:
  - For flexible position, use mapping method 1.
  - For packed position, use mapping method 3.
- Support for mapping method 1 E-UTRA [CPRI v4.1] is available. Note that in order to achieve it in the case of a 15-bit sample width, use the following:
  - For flexible position, use mapping method 1.
  - For packed position, use mapping method 3.
- Support for mapping method 3 WiMAX [CPRI v4.1] is available. Note that in order to achieve it in the case of a 15-bit sample width, use the following:
  - For flexible position, use mapping method 1.
  - For packed position, use mapping method 3.

Supports 15 and 16-bit (including 8b with double oversampling) sample width as in the basic mapping mode.

In addition to the Basic Mapping mode the Advanced Mapping mode supports:

- WiMAX
- Mapping tables to map the IQ samples to the frames
- Flexible position. It is possible to configure the number of stuffing bits between samples or in the beginning of a Basic Frame or AxC Container Block
- Individual sample rates per AxC interface for multiple standards or rates via auxiliary mode. **Note that each unit supports only one sample rate**.
- Supporting K>1
- It is possible to configure an IQ sample rate different from n $\times$ 3.84 Msps. For example, the WiMAX sample rates and E-UTRA sample rate at 1.92 Msps can be configured easily.

K value:

IQ samples are mapped into AxC Container Blocks with a duration of K basic frames, where K is defined in the register CPRI Mapping Table Configuration (CPRIn_MAP_TBL_CONFIG). The K value is common for all AxC interfaces, and the number S of samples allocated to an AxC interface must be also the same. K is typically set to one of the values defined in CPRI v4.1 which are 1, 2, 12, 24 or 48.

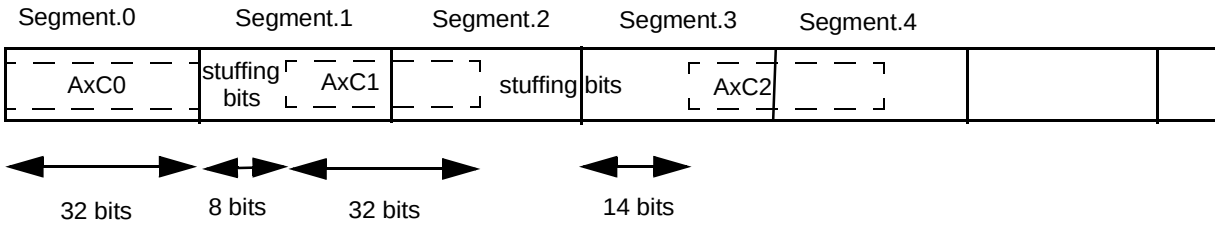
Mapping Table:

Separate mapping tables for RX side and TX side are used to define position of samples in the AxC Container Block. The tables have an entity for each 2*16 bits or 2*15 bits segments of the AxC Container Block depending on the mapping method and if 15-bit sample mode is selected. In case of sample width 15 bits and mapping method 3 the segments are of 30 bits otherwise 32 bits.

Each a table entity contains the following attributes:

- Enable bit, which must be asserted to map an IQ sample into the segment
- AxC number to select the A&C interface
- Bit position to select bit-offset relative to first bit position of the segment
- Sample width to define number of sample bits mapped into the CPRI frame.

Figure 18-9 shows an example for mapping method 3, 16-bit sample width, and how the mapping table should be configured.



Mapping table configuration:

Segment 0: Enable bit =0x1, AxC number=0x0, Width=0x10, Position =0x0
Segment 1: Enable bit =0x1, AxC number=0x1, Width=0x10, Position =0x4
Segment 2: Enable bit =0x0, AxC number=0x0, Width=0x0, Position =0x0
Segment 3: Enable bit =0x1, AxC number=0x2, Width=0x10, Position =0x7

Mapping method 3, Sample width 16 bits

Figure 18-9. Mapping Table Configuration with Sample Width 16 bits

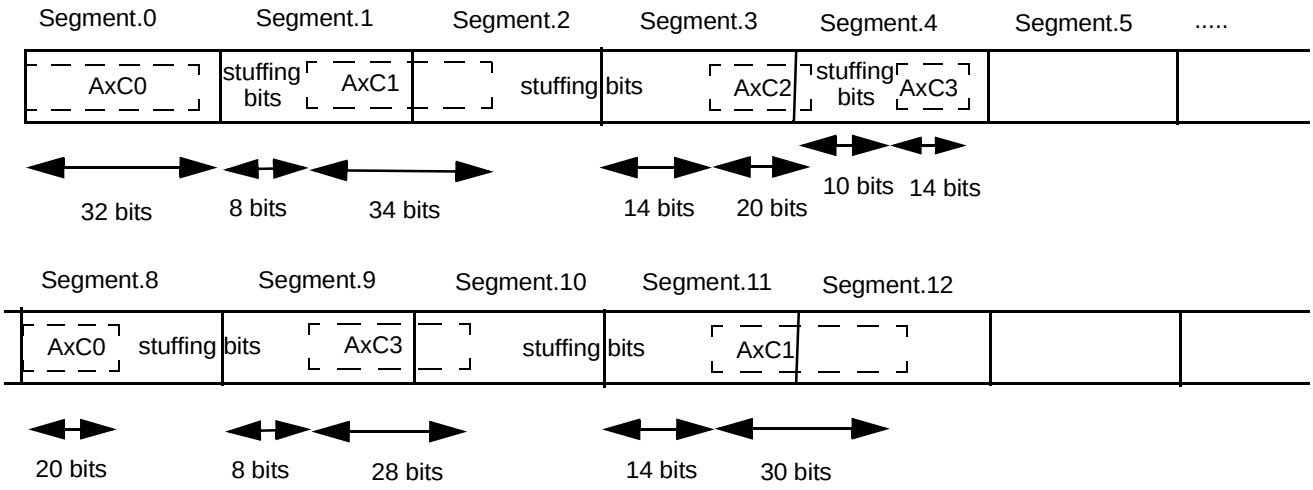
The receive memory fields are described by Receive Configuration Memory (CPRInRCM_<i>) and transmit memory fields are described by Transmit Configuration Memory (CPRInTCM_<i>).

The bit position field can be used for 16-bit sample width in both mapping methods (map_modes “01” and “10”). For 15-bit sample width, the bit position field can be used only in mapping method 1 (map_mode “01”). This makes it possible to define the exact bit offset position relative to the samples default position, such that a sample can be placed on any even bit position of the AxC Container Block.

In Mapping Mode 3, 15-bit sample width, exactly n IQ samples can be mapped into each Basic Frame, and all n*K entries of the table can be used. (n = 4*RATE)

In Mapping Method 1, the width field can be used as well to determine the number of bits used for each segment in the mapping table. This means that a segment may not contain exactly one sample. It could contain less than one sample (in this case that remainder of the sample is transmitted in the next segment enabling the AxC in question), or it can contain more (in this case some bits from the following sample are transmitted).

Figure 18-10 shows an example for mapping method 1, 15 bit sample width, and how the mapping table should be configured.



Mapping table configuration:

Segment 0: Enable bit =0x1, AxC number=0x0, Width=0xf, Position =0x0
Segment 1: Enable bit =0x1, AxC number=0x1, Width=0x11, Position =0x4
Segment 2: Enable bit =0x0, AxC number=0x0, Width=0x0, Position =0x0
Segment 3: Enable bit =0x1, AxC number=0x2, Width=0xa, Position =0x7
Segment 4: Enable bit =0x1, AxC number=0x3, Width=0x7, Position =0x5
Segment 5: Enable bit =0x0, AxC number=0x0, Width=0x0f, Position =0x0
Segment 6: Enable bit =0x0, AxC number=0x0, Width=0x0, Position =0x0
Segment 7: Enable bit =0x0, AxC number=0x0, Width=0x0, Position =0x0
Segment 8: Enable bit =0x1, AxC number=0x0, Width=0xa, Position =0x0
Segment 9: Enable bit =0x1, AxC number=0x3, Width=0xef, Position =0x4
Segment 10: Enable bit =0x0, AxC number=0x0, Width=0x0, Position =0x0
Segment 11 Enable bit =0x1, AxC number=0x1, Width=0x10, Position =0x7

Mapping method 1, Sample width 15 bits

Figure 18-10. Mapping Table Configuration for Sample Width 15 Bits

In 16-bit sample mode, there is no integer number of samples per AxC interface for most of the line rates (see **Table 18-5**). Stuffing bytes are placed by default at the end of each basic frame for mapping methods 1 and 3. The bit position field makes it is possible to place stuffing bits at the beginning of the basic frame or AxC Container Block or distribute them between samples.

Table 18-5. Capacity for 16-bit Mapping Mode

RATE	Line rate	n Number of Segments (timeslots)	Basic frame payload size	Samples	Spare bytes
2	1228Mbps	8	30 bytes	7	2
4	2457Mbps	16	60 bytes	15	0
5	3072Mbps	20	75 bytes	18	3
8	4915Mbps	32	120 bytes	30	0
10	6144Mbps	40	150 bytes	37	2

Segment numbering:

Depending on the mapping mode selected (map_mode and MAP_15BIT_MODE), there are entries of the mapping table that cannot be used. For mapping methods 1 and 3, unused table entries are always placed at the end of each group of n entries, where n is number of 32-bit words per Basic Frame shown in **Table 18-5** (n is also equal to 4*RATE).

For mapping methods 1 and 3, n segments are available for a basic frame, regardless of whether using 15 or 16 bit sample widths. For 15 bit samples in mapping mode 3, all the segments can be used but for 16 bit samples, there are a few segments at the end of each basic frame that cannot be used.

For example, for a 16 bit sample width and 3.072 Gbps line rate (rate=5) with K set to 2, there are 18 AxC Containers per Basic frame. In this case, n = 20 segments are available per basic frame so a second basic frame runs from segment number 20 to segment 37. See **Figure 18-8** for explanation. The total tables required is 40, but two groups of 18 are used.

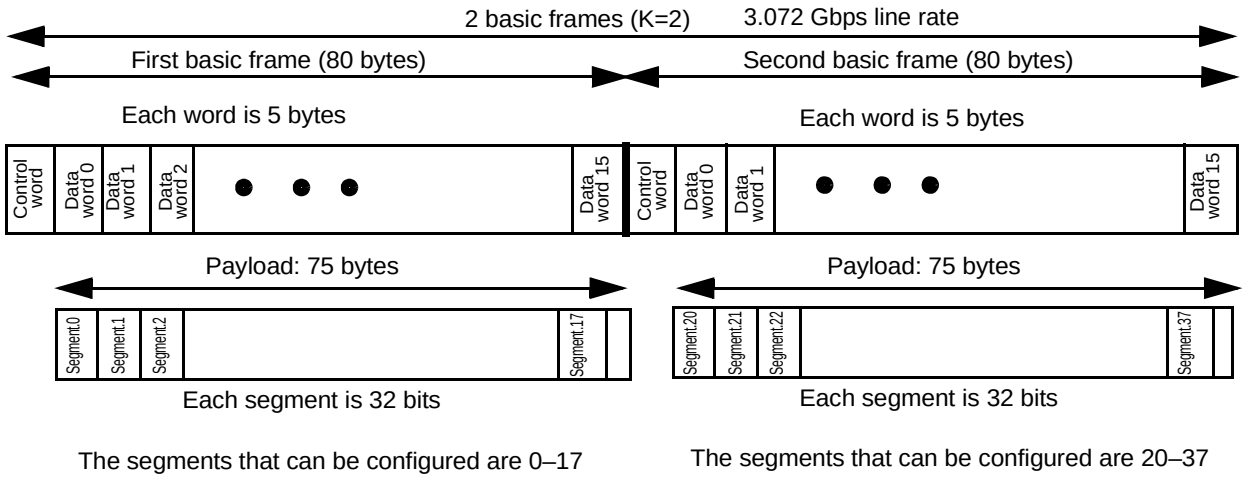


Figure 18-11. Use of Segments

18.3.1.4.3 CPRI IQ MAP Interface Synchronization

The AxC Container Block is synchronized to the start of the 10 ms CPRI frame with a frame offset defined in the CPRIn_MAP_OFFSET_RX and CPRIn_MAP_OFFSET_TX registers. The CPRI RX AxC Container Mapping Block Offset (CPRIn_MAP_OFFSET_RX) register defines where the first receive IQ sample is mapped relative to the start of the radio frame or CPRI Hyper Frame. This enables extraction of the CPRI frame sample position for the different AxCs.

The CPRI TX AxC Container Mapping Block Offset (CPRIn_MAP_OFFSET_TX) register defines the location of the first IQ sample in the transmit CPRI frame. The offset is defined relative to the start of the radio frame or CPRI hyper frame. This enables controlling the start of AxC container block when using advance mapping modes. This position is typically aligned with the WiMAX frame as defined in the CPRI Standard section 4.2.8.2, such that the offset register points at the first basic frame of the first AxC Container block in the WiMAX frame.

The values in the CPRIn_MAP_OFFSET_RX and CPRIn_MAP_OFFSET_TX registers also select the buffer synchronization done for every 10 ms CPRI Radio Frame or for every 67 µs CPRI Hyper Frame.

Typically, the CPRIn_START_OFFSET_TX register is set to an offset that is the number of basic frames before the offset of the CPRIn_MAP_OFFSET_TX register in which the number of basic frames is equal to the Transmit Buffer Size (as defined in the CPRInTCFBS register). For example, if CPRIn_MAP_OFFSET_TX is configured to 0x0 it means that the first IQ sample should be placed at the beginning of Radio frame (10 ms frame) in Hyperframe No 0, Basic frame No 0.

If the transmit buffer size is 16, then CPRIn_START_OFFSET_TX register should be configured as:

CPRIn_START_OFFSET_TX.START_TX_OFFSET_X = 0xF0 (Buffer size (16) before basic frame No 0)

CPRIn_START_OFFSET_TX.START_TX_OFFSET_Z = 0x95 (the1st hyperframe before hyperframe No 0).

See **Figure 18-12** describing the example

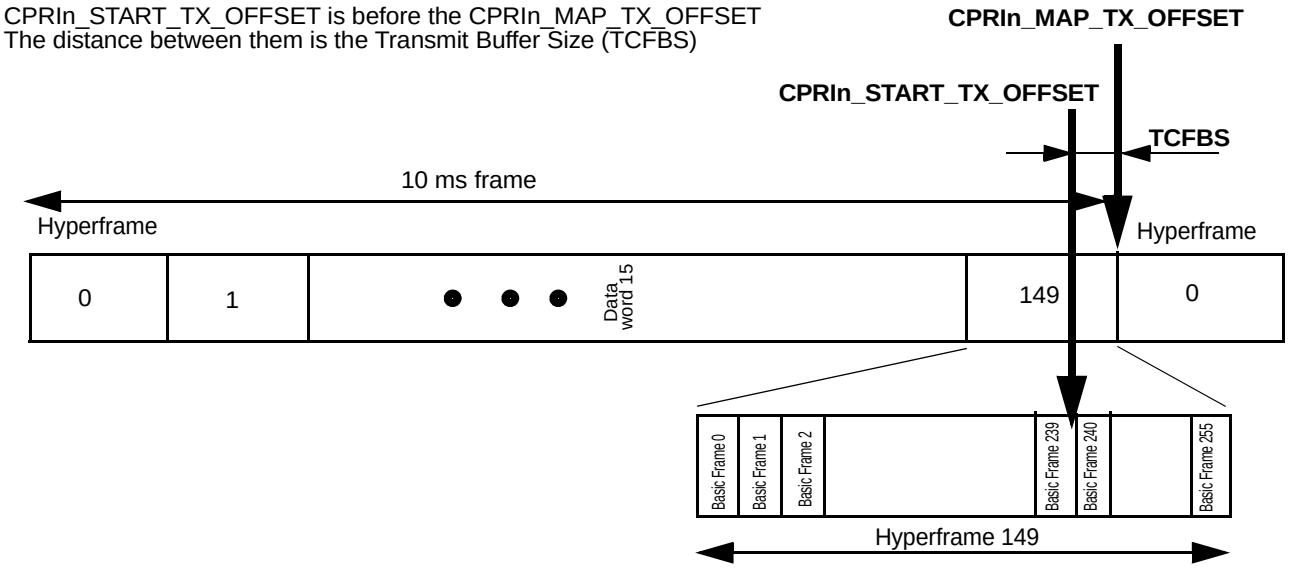


Figure 18-12. Use of Offset registers

18.3.1.5 Control Words

The following sections describe how the control words are processed.

18.3.1.5.1 Control Words Transmission

The following CPRI transmit control data is inserted by the CPRI module:

- Synchronization control byte (K28.5) and filling bytes (D16.2) in the synchronization control word
- HFN
- BFN
- HDLC bit rate
- Fast C&M pointer
- 4b5b encoded Fast C&M Ethernet frames
- Bit-stuffed Slow C&M HDLC frames

- VSS
- L1 Inband
- Alarms

Other control bytes (#Z.X.0.Y) can be inserted via the control transmit table. Programing the control transmit table is done by using the following registers:

- TX_CTRL_INSERT_EN CPRIn_CONFIG
- Transmit Control Data Register 0 (CPRInTCD0)
- Transmit Control Data register 1(CPRInTCD1)
- Transmit Control Data Register 2 (CPRInTCD2)
- Transmit Control Attribute Register (CPRInTCA)
- Transmit Control Table Insert Enable 1(TCTIE1)
- Transmit Control Table Insert Enable 2(TCTIE2)

The transmit table has an entry for each of the 256 control words, each up to 10 bytes values.

The VSS, Ethernet and HDLC data is managed by the DMA, but the other control words (words 0–15, 64–79,192–207,128–143) can be managed through registers mentioned above.

For example, to insert control word number 130, use the following steps:

1. Set bit 2 of TCTIE2 register.
2. Write the required data to TCD0,TCD1 and TCD2 registers.
3. Write the following to the TCA register: {0x0000000000000, 0x1, 0x00000082, 0x0000000000}

To read control word <x>, use the following steps:

1. Write the following to the TCA register: {0x0000000000000, 0x0, <x>, 0x0000000000}
2. Read the TCD0, TCD1, and TCD2 registers.

As the Transmit Control Table is a memory, it must be reset before use during the autonegotiation flow. The way to reset the Transmit Control Table is:

1. Write 0 to Transmit Control Data Register 0 (CPRInTCD0)
2. Write 0 to Transmit Control Data register 1 (CPRInTCD1)
3. Write 0 to Transmit Control Data Register 2 (CPRInTCD2)
4. Write to the Transmit Control Attribute Register (CPRInTCA) the following values in a loop ($0 \le i \le 255$) {0b0000000000000, 0b1, M, 0b1111111111} where M is an 8-bit number equal to i. (The first value of CPRInTCA is 0x00040000, the last is 0x0007FC00)

18.3.1.5.2 Control Words Reception

All received control bytes (#Z.X.Y) can be inspected via the control receive table, which is accessed using the following registers:

- Receive Control Attribute Register (CPRInRCA)
- Receive Control Data register 0 (CPRInRCD0)
- Receive Control Data Register 1 (CPRInRCD1)
- Receive Control Data Register 2 (CPRInRCD2)

The receive table has an entry for each of the 256 control words of a hyperframe containing 256 X × (up to) 10 Y (bytes) values. The table is continuously updated with received control bytes. To read a control byte first write to Receive Control Attribute Register (CPRInRCA) registers and than read the data from Receive Control Data register 0 (CPRInRCD0), Receive Control Data Register 1 (CPRInRCD1) and Receive Control Data Register 2 (CPRInRCD2).

The L1 inband data and the receive VSS can be read from the receive control table, but the received VSS data can be also transferred to internal memory by the receive DMA.

Note: The receive table is only updated when the receive state machine performing CPRI hyper frame alignment in the receive direction reflects alignment. Using the control word access tables, the user can also read the Ethernet and HDLC locations if using the wrong index.

18.3.1.5.3 Fast C&M Channel (Ethernet)

The Ethernet MAC is connected to the Ethernet interface of the CPRIn_RX and CPRIn_TX modules with a synchronous MII. The number of CPRI control words used for Ethernet frame transfer is controlled by the Fast C&M pointer, which is mapped into control byte Z.194.0. This pointer can be set to values from 0x14 to 0x3F. When the Ethernet channel is not used, set the pointer to 0x3F, which means that 4 control words are dedicated for Ethernet and 188 for VSS. To block reception of Ethernet packets, write all zeros to register CPRIn_ETH_CONFIG_1. To prevent transmission of Ethernet packets, do not prepare any BDs. The size of each control word depends on the link rate. Ethernet transfer uses all the bytes of the relevant control words so the achieved Ethernet bit rate depends on both the link rate and the Fast C&M pointer.

The Fast C&M pointer is set up during State D of the autonegotiation flow (See **Section 18.3.1.3.3**) The CPRIn_ETH_CONFIG_1, CPRIn_ETH_CONFIG_2 and CPRIn_ETH_CONFIG_3 registers are used for general Ethernet configurations.

When an Ethernet packet is received, the Ethernet MAC destination address is checked, and, if it does not pass the address filter, the BD reflects that the packet is aborted. The MAC address check can be disabled using the MAC_CHECK field of the CPRIn_ETH_CONFIG_1 register.

MAC address filtering assumes that MAC destination address is the first 6 bytes of the CPRI encapsulated frame. For other frame formats, all MAC address filtering must be disabled. In that case, no particular packet format is expected, and the packet format can be controlled from software. MAC address filtering is configured in the CPRIn_ETH_CONFIG_1 register. If the MAC_CHECK field is cleared (0), all received packets pass (Accept All Mode).

Three different Ethernet MAC address filtering are supported:

- Unicast (single) filtering: The received DMAC (Destination MAC) address is compared with MAC address specified in the CPRIn_ETH_ADDR_LSB and CPRIn_ETH_ADDR_MSB registers.
- Multicast filtering: An Ethernet multicast address is an address with LSB of the first MAC address byte set to one. A very simple HASH table is used to filter multicast mac addresses. It is important that this multicast address filtering is used in combination with full address validation in software. This function is enabled using the MULTICAST_FLT_EN_FIELD of the CPRIn_ETH_CONFIG_1 register.
- Broadcast filtering: An Ethernet broadcast address is an address with all bits set to one. Enable/disable of broadcast mode is via the BROADCAST_EN field of the CPRIn_ETH_CONFIG_1 register

The current packet received can be aborted by setting RX_DISCARD field of the CPRIn_ETH_RX_CONTROL register. After a packet is discarded the next read from the Ethernet RX buffer will be start of a new packet.

The data transfer between the framer and the DMA is based on triggers. Bits 18–17 and 14–10 of register CPRIn_ETH_CONFIG_1 managing the triggering must be set.

The size of the Ethernet buffers is indicated by the CPRInRETHBS register. If the size of an arrived Ethernet packet is larger than the defined Ethernet buffer size, the current received packet is aborted and the ABORT bit is set in the buffer descriptor. If the size of an arrived Ethernet packet is equal to the defined Ethernet buffer size, the current received packet is accepted, but the ABORT bit is set in the buffer descriptor and the packet is aborted.

If the start of packet has been already read by the DMA from the Framer, abort is indicated the next time the CPRIn_ETH_RX_STATUS register is read; otherwise, the previously received part of the packet is just removed without further notice. The CPRIn_ETH_RX_STATUS register indicates if a packet was aborted and the reason of it (for example, CRC error, buffer overflow, and so forth).

Note: Always make sure that the Ethernet buffer size definition in the CPRInRETHBS register is larger than the expected Ethernet packet size.

18.3.1.5.4 Slow C&M Channel (HDLC)

The number of CPRI control bytes used for HDLC frame transfer is controlled by the HDLC rate, which is mapped into control byte Z.66.0. Table 18-6 shows the possible rate configurations; see also CPRI standard Section 4.2.7.6. Setting HDLC bit-rate Minimum CPRI rate.

Table 18-6. HDLC Rate Setting

Setting	HDLC bit-rate in Kbps	Minimum CPRI rate
0	No HDLC	Any rate
1	240	Any rate
2	480	Any rate
3	960	>= 1228.8 Mbps
4	1920	>= 2457.6 Mbps
5	2400	>= 3072.0 Mbps
6	HIghest possible. For link rate 4915 Mbps: 3840. For link rate 6144 Mbps: 4800	>= 4915.2 Mbps

The slow C&M channel functions as follows:

- On the transmit side, the HDLC rate is configured in the CPRIn_CM_CONFIG register.
- On the receive side, the accepted HDLC rate can be read in the CPRIn_CM_STATUS register.
- The HDLC rate setting is negotiated as described in **Section 18.3.1.3** The CPU control interface for HDLC packet transfer is equal to the interface for Ethernet transfer.
- The data transfer between the framer and the DMA is based on triggers. Bits 18–17 and 14–10 of the CPRIn_HDLC_CONFIG_1 register managing the triggering must be set.

18.3.2 The CPRI Clocks

Table 18-7 describes the CPRI complex input clocks.

Table 18-7. CPRI Complex Input Clocks

Clock Name	Clock Frequency	Clock Description
Recovered Receive Clock (CDR Clock)	line_rate/10	The CDR clock is used to receive the incoming CPRI frame. There are six different CDR clocks supplied by the SerDes to CPRI Framer modules. The CDR Clock is asynchronous to the other clocks.
TBI Clock (Ten Bit Interface Clock)	line rate/10	The TBI clock drives the 10 bit transmit data toward the SerDes. This clock is also used by the CPRI Framer module for delay measurement. The TBI clock is common for all the CPRI Framer modules and is synchronous to the Framer Clock
Framer Clock	line rate/40	Framer Clock is the main processing clock of the CPRI Framer module. The Framer Clock is common for all the CPRI Framer modules and it is synchronous to the TBI clock
System Clock	HSSI clock	The System Clock is the system clock of the chip. It is used by the CPRI DMA to transfer data between the system memories and the CPRI Framer modules, to configure the memories and registers of the CPRI complex. The System Clock is asynchronous to other clocks.

18.3.3 CPRI DMA Controller

The CPRI DMA is composed of 6 identical DMA modules where each DMA module is connected to a single CPRI Framer. Each CPRI DMA controller contains a receive DMA module that transfers the received VSS, Ethernet, HDLC, and IQ data from the CPRI Framer to buffers in the system memory via the write MBus bus. The CPRI DMA block also contains transmit DMA module that transfers the VSS, Ethernet, HDLC, and IQ data from the buffers in the system memory to the CPRI Framer via the read MBus bus. The CPRI DMA block also contains an additional receive DMA module that transfers the IQ data from the CPRI Framer to buffers in the MAPLE in multicast mode through a direct bus.

All the six receive DMA modules are connected to an arbiter that controls write accesses between the different receive DMA modules to the MBus. The arbiter simultaneously performs a Pseudo Round Robin arbitration algorithm for each of the priority levels and chooses the highest level request. The priority level of a write accesses depends on how full its DMA buffer is. All six transmit DMA modules also connect to an arbiter that controls read accesses to the MBus bus by the different transmit DMA modules. As for write access, the arbiter simultaneously performs a Pseudo Round Robin arbitration algorithm for each of the priority levels and chooses the highest level request. The priority level of a read accesses depends on how empty its DMA buffer is.

Each receive CPRI DMA module and transmit CPRI DMA module contains four sub modules:

- rec_vss/trn_vss: handles the receive/transmit VSS data
- rec_eth/trn_eth: handles the receive/transmit Ethernet data
- rec_hdlc/trn_hdlc: handles the receive/transmit HDLC data
- rec_IQ/trn_IQ: handles the receive/transmit IQ data.

Figure 18-4 describes the CPRI DMA block diagram.

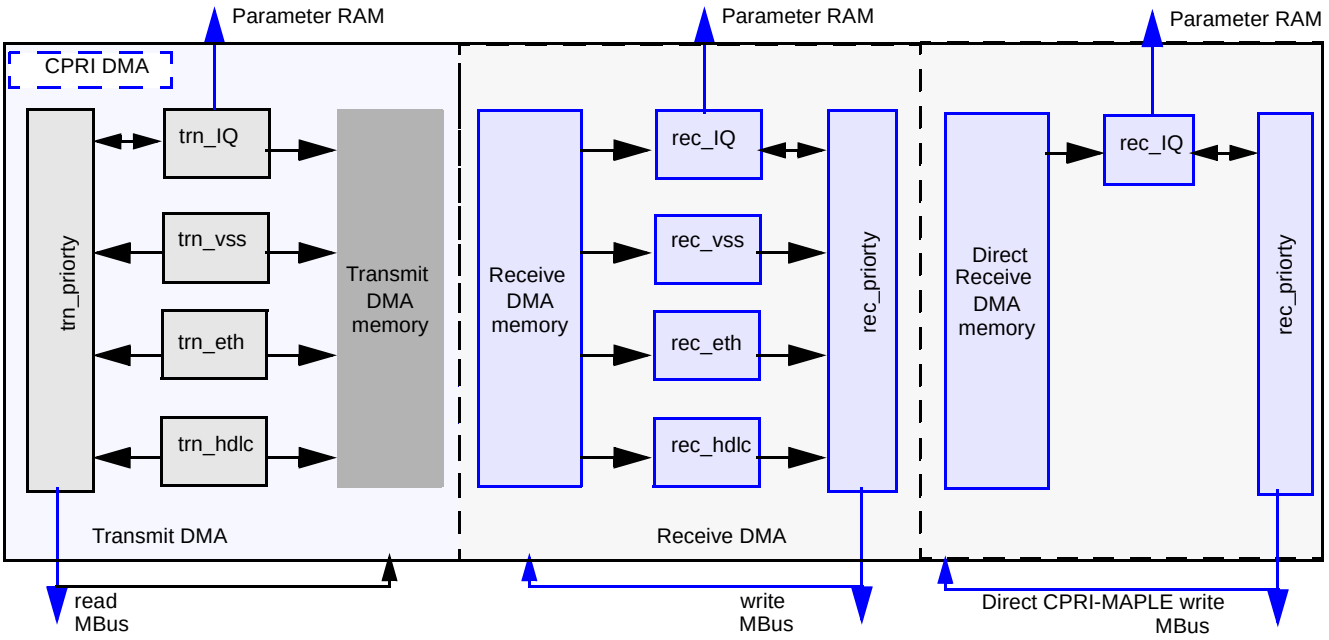


Figure 18-13. CPRI DMA Block Diagram

18.3.3.1 Receive DMA and Transmit DMA Memories

The receive DMA module contains receive DMA memory that temporarily stores the received IQ, VSS, HDLC, and Ethernet data. Similarly, the transmit DMA modules also contains transmit DMA memory that temporarily stores the transmit IQ, VSS, HDLC, and Ethernet data. The receive DMA memory and transmit DMA memory use the same partitioning, which is described in **Table 18-8 on page 18-29**.

Table 18-8. Receive DMA/Transmit DMA Memory Partition

Memory Section	Number of Bytes Allocated for each section
IQ data	256 bytes × 24(maximum number of AxC)
VSS data	256 bytes
ETH data	256 bytes
HDLC data	256 bytes

The IQ data section in the Receive DMA memory is divided into 96 buffers of 64 bytes each. The buffers for each antenna carrier are consecutive in the Receive DMA memory. The number of buffers allocated for each antenna carrier depends on the total number of antenna carriers. For example, if the number of antenna carriers is 3, then each AxC has 2048 bytes (64 bytes × 32(number of buffers)). The number of antenna carriers is described by CPRInRGCM[MAP_AC] field.

Table 18-9 on page 18-30 describes the memory configuration as a function of the number of antenna carriers.

Table 18-9. Memory Configuration As Function of Number of Antenna Carriers

Number of Antenna Carriers	Number of Buffers (of 64 bytes) for Each Antenna Carrier	Valid MBus Transaction Size	Notes
1	96	2048 bytes, 1024 bytes, 512 bytes, 256 bytes, 128 bytes or 64 bytes	The MBus transaction size is configurable as 64 bytes, 128 bytes, 256 bytes, 512 bytes, 1024 bytes or 2048 bytes. If the transaction size is 64 bytes, then MBus request is generated only if at least 1 64 bytes buffers for each Antenna Carriers are full If the transaction size is 128 bytes, then MBus request is generated only if at least 2 64 bytes buffers for each Antenna Carriers are full If the transaction size is 256 bytes, then MBus request is generated only if at least 4 64 bytes buffers for each Antenna Carriers are full If the transaction size is 512 bytes, then MBus request is generated only if at least 8 64 bytes buffers for each Antenna Carriers are full If the transaction size is 1024 bytes, then MBus request is generated only if at least 16 64 bytes buffers for each Antenna Carriers are full If the transaction size is 2048 bytes, then MBus request is generated only if at least 32 64 bytes buffers for each Antenna Carriers are full
2	48	1024 bytes, 512 bytes, 256 bytes, 128 bytes or 64 bytes	
3	32		
4	24	512 bytes, 256 bytes, 128 bytes or 64 bytes	
5	18		
6	16		
7	13	256 bytes, 128 bytes or 64 bytes	
8	12		
9	10		
10	9		
11	8		
12			
13	7	128 bytes or 64 bytes 128 bytes or 64 bytes	
14	6		
15			
16			
17	5		
18			
19	4		
20			
21			
22			
23			
24			

18.3.3.2 Receive IQ Data Flow

The receive CPRI Framer module extracts the antenna carrier data from the CPRI frame and transfers it to the receive DMA. The received antenna carrier data is temporarily stored in the receive DMA memory until it is transferred to the received antenna carrier buffers mapped in the system memory. A single transactions from the receive DMA memory to system memory is done via the MBus interface with minimal access size of at least 64 bytes. The receive MBus transaction size is programmable by the CPRInRIQMTS register (for details, see **Section 18.4.2.1**, *Receive IQ MBus Transaction Size (CPRInRIQMTS)*, on page 18-97). Small

MBus transaction size reduces the system latency but it may impact on the overall DSP device performance.

Note: The maximum valid transaction size is determined by CPRInRGCM[RMAP_AC] and CPRInRIQMTS[RSM] fields.

Each receive antenna carrier is stored in a different buffer mapped in the system memory. This buffer can be located in any location in the system memories. The antenna carrier n buffer base address is determined by Receive Antenna Carrier Parameter Register n (CPRInRACPRy) register. The antenna carrier buffer base address should be aligned to 16 bytes.

The buffers can be located in the DDR, M3, or any of the M2 memories. Also part of the buffers can be in one memory and other buffers in another memory, the only constraint is that if M2 memory is used all the buffers of a given CPRI unit must be in the same M2 memory.

The antenna carriers buffer size is identical for all the receive antenna carrier belonging to a CPRI link and is defined by Receive IQ Buffer Size (CPRInRIQBS) register. The antenna carrier buffer size should be aligned to MBus transaction size.

The antenna carriers have the same sample rate therefore the IQ buffers write pointer is identical for all the receives antenna carriers and it's described by the status register Receive IQ Buffer Displacement Register (CPRInRIQBDR). Adding this register to the Receive Antenna Carrier Base Address (CPRInRACPRy[RACBA]) field indicates the location to which CPRI writes next.

Figure 18-14 shows how the samples are stored in the system antenna carrier (IQ) buffer. Note that when the sample width is 15 bits, then Q0 and I0 are clear.

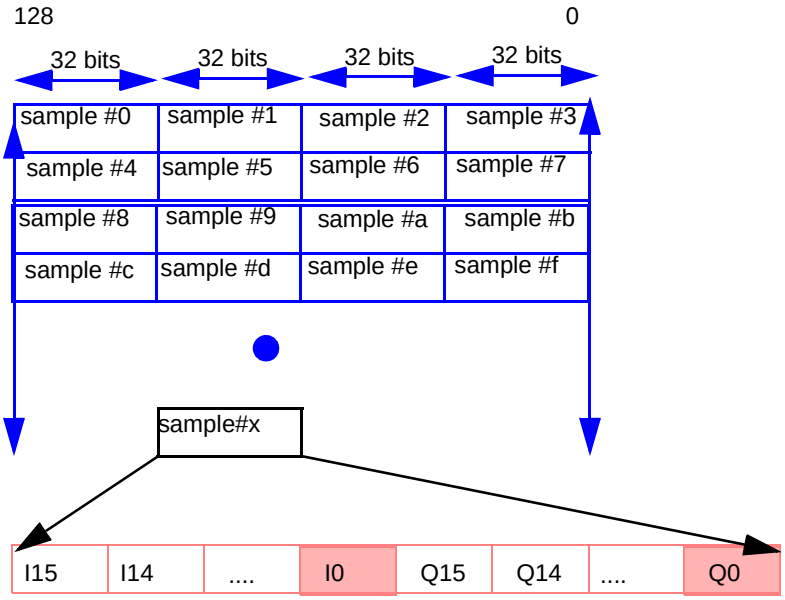


Figure 18-14. Antenna Carrier (IQ) Buffer in the Main Memory

The receive IQ buffers in the system memory share two threshold levels. The CPRI notifies the cores each time it fills the receive buffers up to a threshold level. The first threshold level is determined by Receive IQ First Threshold (CPRInRIQFT) and the second receive threshold level is determined by Receive IQ Second Threshold (CPRInRIQST).

When the CPRI DMA receiver fills the receive buffer through the MBus interface up to an offset defined by the first threshold defined by the CPRInRIQFT[RIQFT] field, the CPRInRER[RIQFTE] bit is set. If the bit CPRIICRy[RIQFTE] is also set and CPRInRER register is selected, a first threshold interrupt is generated. When the interrupt is asserted in the EPIC, then the SC3850 core can read all the receive buffers from their beginning up to the first threshold (RIQFT) point. Meanwhile, the CPRI DMA keeps writing new data to the second part of the buffer

When the CPRI DMA receiver fills the receive buffer through the MBus interface up to an offset defined by the second threshold defined by the CPRInRIQST[RIQST] field, the CPRInRER[RIQSTE] bit is set. If the bit CPRIICRy[RIQSTE] is also set and CPRInRER register is selected, a second threshold interrupt is generated.

The threshold interrupts can be generates as pulse or level, as determined by CPRIICRy[LEVEL] bit. If the interrupt is level, then the ISR should clear the appropriate bit in the CPRInRER register by writing a 1 to it. If the interrupt is pulse, there is no need to clear the status bit.

Figure 18-15 illustrates the pointers associated with receive AxC buffers that are mapped on the system memory.

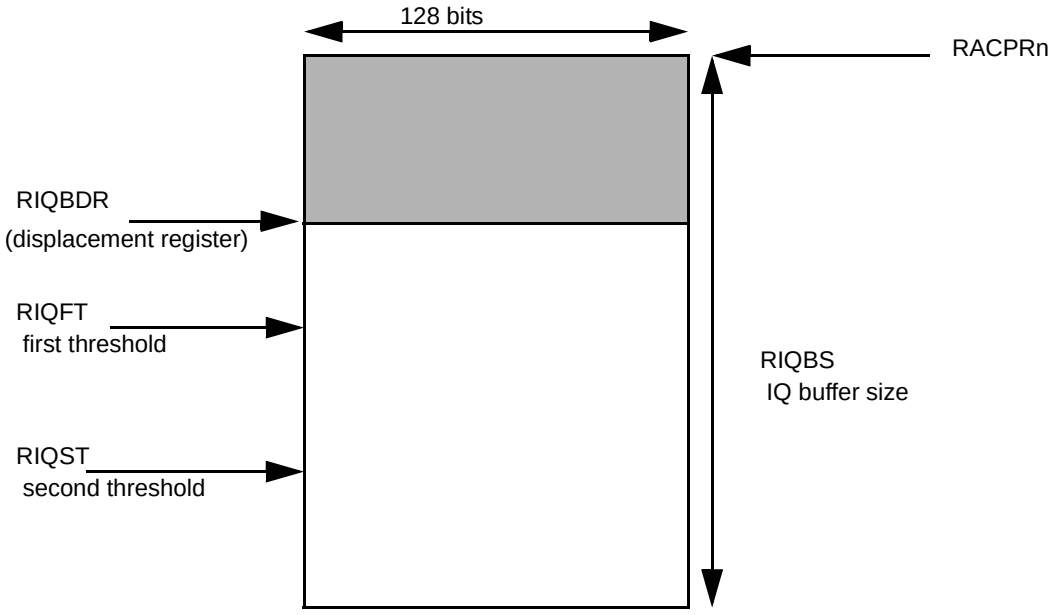


Figure 18-15. System Memory Receive IQ Buffers Threshold Pointers

It is also possible to generate an interrupt when a given number of bytes were written to the system memory as defined by register CPRInRIQT.

When the CPRI DMA cannot transfer data to AxC buffers in the system memory an overrun occurs and the CPRInEER[RIQOV] bit is set and if the CPRInEIER[RIQOVE] bit is also set, an error interrupt is generated. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the system interface, and, therefore, cannot write the data into the destination memory (the AxC buffers).

The receive DMA transfers received data to system memory only if the Receive Control Register (CPRInRCR) [RIQE] bit set. The Receive AxC Control Register (CPRInRACCR) determines which received antenna carrier is active.

18.3.3.3 Transmit IQ Data Flow

AxC data transmitted from the buffer mapped on system memory is temporarily stored in the CPRI transmit DMA memory until it is transmitted via the CPRI Framer.The data is transferred from the system memory to CPRI transmit DMA memory via the MBus. The transmit MBus transaction size is programmable by the CPRInTIQMTS register. For details, see **Section 18.4.2.3**, *Transmit IQ MBus Transaction Size (CPRInTIQMTS)*

Each transmit antenna carrier is stored in a different buffer mapped in the system memory. This buffer can be located in any location in the system memories. The antenna carrier n transmit buffer base address is determined by CPRInTACPRy register. The antenna carrier transmit buffer base address should be aligned to 16 bytes.

The antenna carrier buffer size is identical for all the transmit antenna carriers belonging to a CPRI and is defined by Transmit IQ Buffer Size (CPRInTIQBS) register. The transmit AxC buffer size should be aligned to the transmit MBus transaction size.

Figure 18-14 shows how the samples are stored in the system antenna carrier (IQ) buffer. Note that when the sample width is 15 bits, then Q0 and I0 should be clear.

The transmit AxC buffers share the same read pointer since the sample rate of all the AxC is the same. The read pointer is describes by the status register Transmit IQ Buffer Displacement Register (CPRInTIQBDR)). Adding this register to the transmit AxC buffer base address (CPRInTACPRy[TACBA]) field indicates the location to which the CPRI DMA reads next.This register can be used to show which data is already read from the buffer so that the buffer can be filled with new data.

The transmit AxC buffers also share two threshold levels. The CPRI notifies the SC3850 core each time it reads from the transmit buffer up to a threshold level. The first transmit threshold level is determined by Transmit IQ First Threshold (CPRInTIQFT) register and the second threshold level is determined by the Transmit IQ Second Threshold (CPRInTIQST)). If the transmit CPRI DMA reads the data until the first threshold, then the status bit CPRInTER[TIQFTE] bit is set and if it reads the data until the second threshold level, then the status bit CPRInTER[TIQSTE] is set. If the CPRIICRy[IQFTE] bit is set and CPRInTER is selected by the CPRIICRy[ERS] field, then a first threshold interrupt is generated. If the

CPRIICRy[IQFSE] bit is set and CPRInTER is selected by the CPRIICRy[ERS] field, then a second threshold interrupt is generated.The interrupt can configured as pulse or level, as determined by the CPRIICRy[LEVEL] bit. If the interrupt is level, then the ISR should clear the CPRInTER[TIQFTE/TIQSTE] bit by writing a 1 to it. If the interrupt is pulse, there is no need to clear the status bit.

Figure 18-16 illustrates the pointers associated with transmit AxC buffers that are mapped on the system memory.

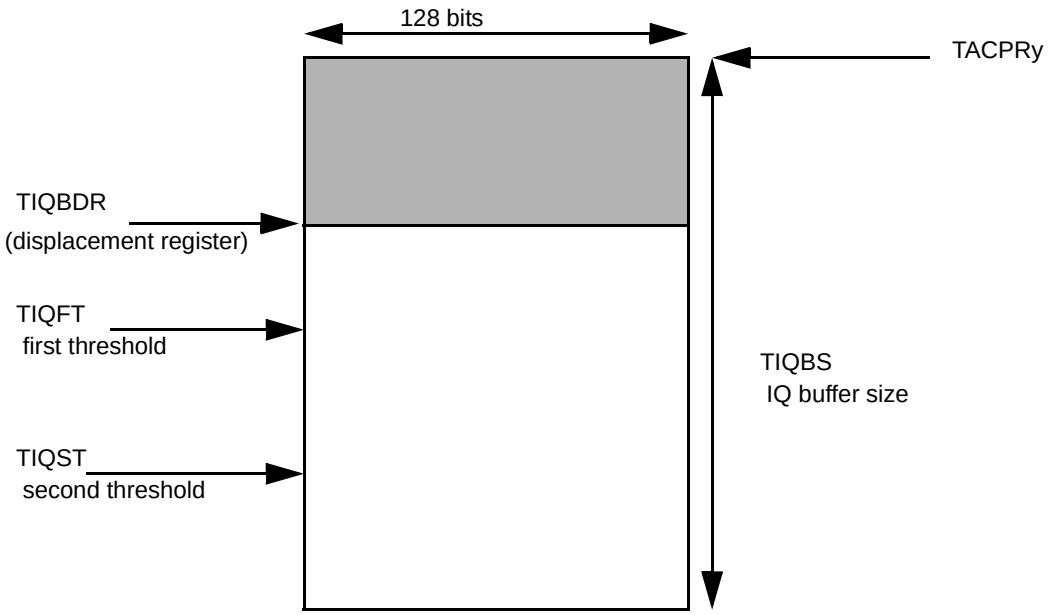


Figure 18-16. System Memory Transmit IQ Buffers Threshold Pointers

It is also possible to generate an interrupt when a given number of bytes are read from the system memory as defined by register CPRInTIQT.

When the CPRI DMA cannot transfer data from AxC buffers to CPRI DMA transmit memory, an underrun occurs. When the CPRI transmit DMA memory is empty, the CPRInEER[TIQUR] bit is set and if the CPRInEIER[TIQURE] bit is also set, an error interrupt is generated. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the system interface, and, therefore, cannot read the data from the system memory into CPRI transmit DMA memory.

The transmit DMA transfers transmit data from the system memory only if the Transmit Control Register (CPRInTCR)[TIQE] bit is set. The Transmit AxC Control Register (CPRInTACCR) determines which transmit antenna carrier is active.

18.3.3.4 Receive VSS (Vendor Specific Data) Data Flow

The receive CPRI Framer module extracts the VSS data from the CPRI frame and writes it to control memory in the CPRI Framer. The receive DMA reads the VSS data from the control

memory and stores the VSS data temporarily in the VSS section of the receive DMA memory until it is transferred to the VSS buffer in the system memory.

The received VSS data can be also read directly from the receive control table by the core. The Receive Control Register (CPRInRCR)[RVSSE] bit enables the writing of the received VSS data to VSS buffer in the system memory.

The VSS data is aligned to a hyperframe and the amount of VSS bytes during one hyper frame is determined by the Ethernet pointer. The minimum number of VSS control words during one hyperframe is 16. The maximum number of VSS control words during one hyperframe is 192.

The VSS data is transferred to VSS buffer via the MBus bus. The VSS MBus transaction size is configurable by register and it can be 16 bytes, 32 bytes, 64 bytes or128 bytes. The VSS data is stored in the VSS buffer in the system memory. This buffer can be located in any location in the system memories. The VSS buffer base address is determined by the Receive VSS Base Address (CPRInRVSSBA) and its size is also configurable. This size is determined by the Receive VSS Buffer Size (CPRInRVSSBS) register. The VSS buffer size should be aligned to the receive VSS MBus transaction size.

The VSS buffer write pointer is defined by the status register CPRInRVSSBDR. Adding this register to the VSS buffer base address (Receive VSS Base Address (CPRInRVSSBA)) field indicates the location to which the CPRI writes next.

The receive DMA notifies the cores each time that it fills the VSS buffer with N VSS data bytes. The number N is determined by CPRInRVSST register. CPRI_INTy interrupt is generated if the CPRIICRy[EVSSTE] bit is set and CPRInRER register is selected by CPRIICRy[ESR] field. This interrupt can be configured as pulse or level as determined by the CPRIICRy[LEVEL] bit. The control interrupt CPRIn_RX_control is also generated if the CPRInRCIER[RVSSEE] bit is set. The interrupt can be pulse or level and it is determined by CPRInRCIER[LEVEL] bit.

When the CPRI DMA cannot transfer the VSS data to VSS buffer in system memory, an overrun occurs and bit CPRInEER[RVSSOV] is set. If CPRInEIER[RVSSOVE] bit is also set, an error interrupt is generated. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the system interface, and, therefore, cannot write the data into the VSS buffer in the system memory.

18.3.3.5 Transmit VSS (Vendor Specific Data) Data Flow

The VSS data is transmitted from the VSS buffer mapped in system memory and is temporarily stored in the VSS section of the transmit DMA memory. The transmit DMA reads the VSS data from the transmit DMA memory and writes it to the transmit control table in the CPRI Framer. The CPRI Framer reads the VSS data from the control table and transfers it externally. The transmit VSS data can be also written directly to the transmit control table by the core via. The Transmit Control Register (CPRInTCR)[TVSSE] bit enables the reading of the transmit VSS data via the MBus bus.

The VSS MBus transaction size is configurable by Transmit VSS MBus Transaction Size (CPRInTVSSMTS) register and it can be 16 bytes, 32 bytes, 64 bytes or 128 bytes.

The transmit VSS data is stored in the VSS buffer in the system memory. This buffer can be located in any location in the system memories. The VSS buffer base address is determined by Transmit VSS Base Address (CPRInTVSSBA). Its size is also configurable and is determined by Transmit VSS Buffer Size (CPRInTVSSBS) register. The VSS buffer size should be aligned to the transmit VSS MBus transaction size.

The VSS buffer read pointer is defined by the status register CPRInTVSSBDR. Adding this register to the transmit VSS buffer base address field indicates the location that CPRI reads next.

The transmit DMA notifies the cores each time that it read N bytes of VSS data from the VSS buffer. The number N is determined by CPRInTVSST register. An interrupt is generated if the CPRIICRy[EVSSTE] bit is set and CPRInTER register is selected by CPRIICRy[ESR] field. This interrupt can be configured as pulse or level as determined by the CPRIICRy[LEVEL] bit. The control interrupt CPRIn_tx_control is also generated if the CPRInTCIER[TVSSEE] bit is set. The interrupt can be pulse or level as determined by CPRInTCIER[LEVEL] bit.

When the CPRI DMA cannot transfer the VSS data from the VSS buffer in the system memory, an underrun occurs and the CPRInEER[TVSSUN] bit is set. If the CPRInEIER[TVSSUNE] bit is also set, an error interrupt is generated. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the system interface, and, therefore, cannot read the data from the VSS buffer in the system memory.

18.3.3.6 Receive Ethernet Data Flow

The CPRI Framer extracts the Ethernet packets from the CPRI frame and transfers them to an integrated Ethernet MAC in the CPRI Framer. The MAC decodes the 4B5B Ethernet frame and checks the destination MAC address. If the destination MAC address passes the address filter, the data is transferred to system memory via the MBus. Otherwise, the Buffer Descriptor reflects an aborted packet.

The Ethernet MAC supports three different MAC address filters:

- Unicast filtering
- Multicast filtering
- Broadcast filtering

Each Ethernet packet is stored in a different buffer in the system memory where the Ethernet data buffer location is described by RDBP pointer field of CPRInRETHBD register. The size of the receive Ethernet buffers is indicated by the CPRInRETHBS register. If the size of an arrived Ethernet packet is equal to or greater than the Ethernet buffer size, the packet is aborted and the ABORT bit in the buffer descriptor is set.

For the case of overrun for Ethernet RX, the RX BD does not reflect that the data in the relevant packet is corrupted. The CPRI generates an error interrupt (if enabled) that indicates that an Ethernet RX overrun has occurred. Overrun and underrun cases should only occur during system development and not during normal operation.

The receive buffer descriptors are located in the system memory in consecutive order. The base address of the buffer descriptor ring is determined by the Receive Ethernet BD Ring Base Address (CPRInREBDRBA) register and the number of buffer descriptors in the ring is determined by the Receive Ethernet BD Ring Size (CPRInREBDRS) register. The minimum number of buffer descriptors that the ring supports is four and the maximum number of buffer descriptors in the ring is 256.

The receive DMA has a CPRInREWPR register that indicates the next written buffer descriptor number. This register is updated only by the core each time that a new receive buffer descriptor (or some buffer descriptor) is updated in the system memory. It is the core responsibility to ensure that a Buffer Descriptor underrun (lack of receive descriptors) does not occur. Each time that the core wraps to the beginning of the buffer descriptor ring, the core should update the wrap bit.

The receive DMA Ethernet controller also contains a status register CPRInRERPR that indicates the next buffer descriptor to be read by the receive DMA Ethernet controller. This register is updated only by the receive DMA.

Figure 18-17 shows the Ethernet buffer descriptors rings.

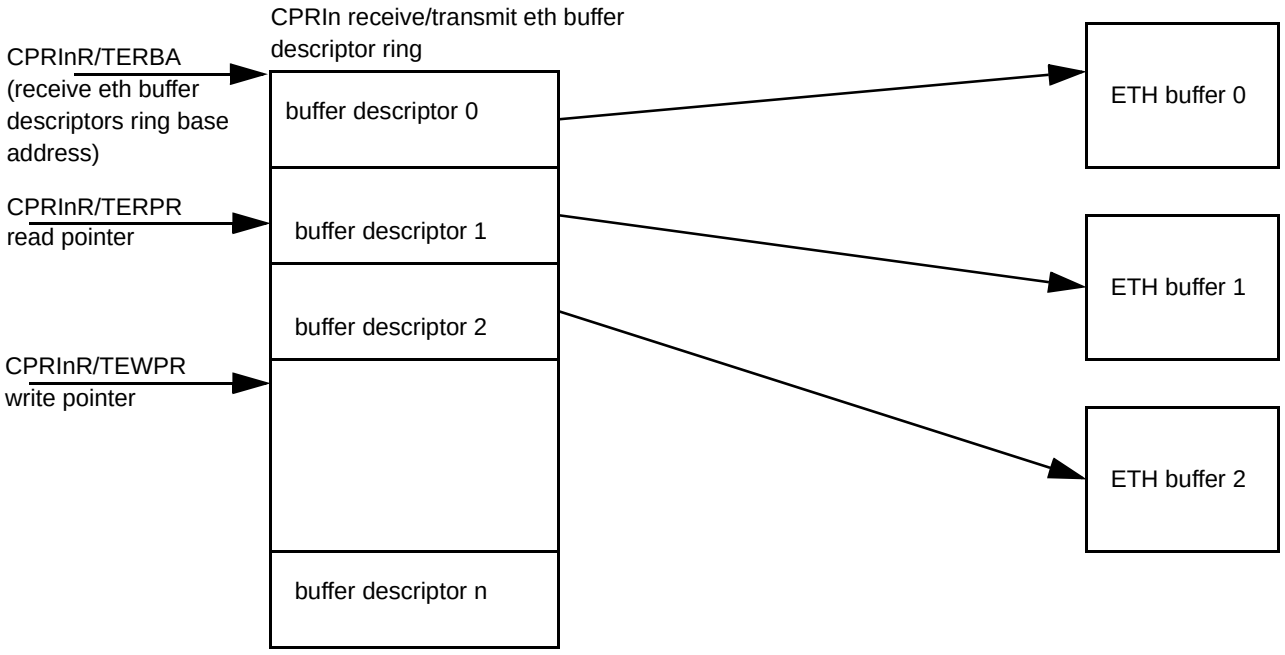


Figure 18-17. Ethernet Buffer Descriptor Ring

18.3.3.7 Receive Ethernet Interrupts

The receive DMA can notify the cores each time that a packet is transferred to the system memory or after a programmable number of packets are transferred to the system memory. The number of packets is determined by the CPRInRETHCT register. The number of packets transferred is indicated by the coalescing counter status register (CPRInRETHCS). In unicast and multicast MAC address filtering, an interrupt is generated not only when a programmed number of packets are transferred to system memory, but also when part or all of the packets do not fit the MAC filter. In this case, the BD of the packets reflects that the packet is aborted.

When the coalescing counter reaches its threshold value, the status bit RETHE in CPRInRER is set, an interrupt is generated if the CPRIICRy[EETHE] bit is set, and the CPRInRER event register is selected by the ESR field. This interrupt can be configured as pulse or level as determined by CPRIICRy[LEVEL] bit. The control interrupt CPRIn_Rx_control is also generated if the CPRInRCIER[RETHEE] bit is set. The interrupt can be pulse or level and it is determined by CPRInRCIER[LEVEL] bit.

When the CPRI DMA can no longer transfer the Ethernet packets to buffers in the system memory, an overrun occurs and the CPRInEER[RETHOV] bit is set. If the CPRInEIER[RETHOVE] bit is also set, an error interrupt is generated. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the system interface, and, therefore, cannot write the data into the Ethernet buffer in the system memory.

When a new packet arrives and the buffer descriptor ring does not contain a valid buffer descriptor, a buffer descriptor underrun occurs. In this case, the CPRInEER[REBDUN] bit is set and, if CPRInEIER[REBDUNE] bit is also set, an error interrupt is generated.

18.3.3.8 Transmit Ethernet Data Flow

The Ethernet packet is transmitted from the Ethernet buffer on the system memory and is temporarily stored in the Ethernet section of the transmit DMA data memory. The transmit DMA transfers the Ethernet packets to an integrated Ethernet MAC in the CPRI Framer module. The Ethernet MAC decodes 4b5b the Ethernet frame and transfers the packets externally. Each Ethernet packet is stored in different buffers in the system memory in the location described by the transmit Ethernet data buffer location in the TDBP pointer field of CPRInTETHBD register.

The transmit buffer descriptors are located in the main memory in a consecutive order. The base address of the buffer descriptor ring is determined by the Transmit Ethernet BD Ring Base Address (CPRInTEBDRBA) register and the number of buffer descriptors in the ring is determined by the Transmit Ethernet BD Ring Size (CPRInTEBDRS) register. The minimum number of supported buffer descriptors in the ring is four and the maximum number of buffer descriptor in the ring is 256.

The transmit DMA Ethernet controller contains a status register CPRInTEWPR that indicates the next written buffer descriptor number. This register is updated only by the core and is updated each time that the core writes a new buffer descriptor (or some buffer descriptors). When there are no valid buffer descriptors the CPRI doesn't transmit Ethernet packets. Each time that the core wraps to the beginning of the buffer descriptor ring, the core should update the wrap bit.

The transmit DMA Ethernet controller contains also a status register CPRInTERPR that indicates the next buffer descriptor that should be read by the transmit DMA Ethernet controller. This register is updated only by the transmit DMA Ethernet controller. **Figure 18-17 on page 18-37** shows the Ethernet buffer descriptors ring.

18.3.3.9 Ethernet Transmit Interrupts

The transmit DMA can notify the cores each time that a packet is transferred from the system memory or after that a programmable number of packets are transferred from the system memory. The number of packets transferred is indicated by the coalescing counter status register (CPRInTETHCS).

When the coalescing counter reaches its threshold value, the status bit TETHE in the CPRInTER is set and an interrupt is generated if the CPRIICRy[EETHE] bit is set and the CPRInTER event register is selected by the ESR field.This interrupt can be configured as pulse or level as determined by the CPRIICRy[LEVEL] bit.

The control interrupt CPRIn_Tx_control is also generated if the CPRInTCIER[TETHEE] bit is set. The interrupt can be pulse or level as determined by the CPRInRCIER[LEVEL] bit.

When the CPRI DMA can no longer transfer the Ethernet packets from buffers in the system memory, an overrun occurs and the CPRInEER[TETHUN] bit is set. If the CPRInEIER[TETHUNE] bit is also set, an error interrupt is generated. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the system interface, and, therefore, cannot read the data from the Ethernet buffer in the system memory.

18.3.3.10 Receive HDLC Data Flow

The CPRI Framer extracts the HDLC packets from the CPRI frame and transfers them to an integrated HDLC MAC in the CPRI Framer. Each HDLC packet is stored in a different buffer in the system memory in the HDLC data buffer location described by the RDBP pointer field of the CPRInRHDLCBD register. The size of the receive HDLC buffers is determined by the CPRInRHDLCBS register. If the size of an HDLC packet that arrives exceeds the HDLC buffer size, then the packet is aborted and the ABORT bit in the buffer descriptor is set.

For HDLC RX overruns, the RX BD does not reflect that the data of the relevant packet is corrupted. The CPRI generates an error interrupt (if enabled) to reflect that the RX overrun has

occurred. Overrun and underrun cases should occur only during system development and not during normal operation.\

The receive buffer descriptors are located in the system memory in consecutive order. The base address of the buffer descriptor ring is determined by the CPRInRHBDRBA register and the number of buffer descriptors in the ring is determined by Receive HDLC BD Ring Size (CPRInRHBDRS) register. The minimum number of buffer descriptors that the ring supports is four and the maximum number of buffer descriptors in the ring is 256.

The receive DMA contains a CPRInRHWPR register that indicates the next written buffer descriptor number. This register is updated only by the core each time that a new receive buffer descriptor (or some buffer descriptors) is updated in the system memory. It is the core responsibility to ensure that a buffer descriptor underrun does not occur. Each time that the core wraps to the beginning of the buffer descriptor ring, the core should update the wrap bit.

The receive DMA HDLC contains also a status register CPRInRHRPR that indicates the next buffer descriptor that should be read by the receive DMA. This register is updated only by the receive DMA. The structure of the HDLC buffer descriptor ring is similar to the Ethernet buffer descriptor ring described in **Figure 18-17 on page 18-37**

18.3.3.11 Receive HDLC Interrupts

Whenever an HDLC packet is transferred to system memory, then the RHDLCE bit in the CPRInRER register is set. A CPRI_INTy interrupt is generated if the CPRIICRy[EHDLCE] bit is set and the CPRInRER event register is selected by the ESR field. This interrupt can be pulse or level as determined by CPRIICRy[LEVEL] bit. The control interrupt CPRIn_RX_control is generated if the bit CPRInRCIER[RHDLCEE] bit is set. The interrupt can be pulse or level as determines by CPRInRCIER[LEVEL].

When the CPRI DMA can no longer transfer the HDLC packets to buffers in the system memory, an overrun occurs and the CPRInEER[RHDLCOV] bit is set. If the CPRInEIER[RHDLCOVE] bit is also set, an error interrupt is generated. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the system interface, and, therefore, cannot write the data into the HDLC buffer in the system memory.

If a new packet arrives and the buffer descriptors ring does not contain a valid buffer descriptor, a buffer descriptor underrun occurs. In this case, the CPRInEER[RHBDUN] bit is set and if CPRInEIER[RHBDUNE] bit is also set, an error interrupt is generated.

18.3.3.12 Transmit HDLC Data Flow

The HDLC packet is transmitted from the HDLC buffers on the system memory and it is temporarily stored in the HDLC section of the transmit DMA memory. The transmit DMA transfers the HDLC packets to an integrated HDLC MAC in the CPRI Framer module.

Each HDLC packet is stored in a different buffer in the system memory for which the transmit HDLC data buffer location is described by TDBP pointer field of CPRInTHDLCBD register.

The transmit buffer descriptors are located in the main memory in consecutive order. The base address of the buffer descriptor ring is determined by the CPRInTHBDRBA register and the number of buffer descriptors in the ring is determined by the Transmit HDLC BD Ring Size (CPRInTHBDRS) register. The minimum number of supported buffer descriptors in the ring is four and the maximum number of buffer descriptors in the ring is 256.

The transmit DMA contains a status register CPRInTHWPR that indicates the next written buffer descriptor number. This register is updated only by the core and it updates each time that the core writes a new buffer descriptor (or some buffer descriptors). When there are no valid buffer descriptors the CPRI doesn't transmit HDLC packets. Each time that the core wraps to the beginning of the buffer descriptor ring, the core should update the wrap bit.

The transmit DMA contains also has a status register CPRInTHRPR that indicates the next buffer descriptor for the transmit DMA to read. This register is updated only by the transmit DMA. The structure of the HDLC buffer descriptor ring is similar to the Ethernet buffer descriptor ring described in **Figure 18-17 on page 18-37.**

18.3.3.13 HDLC Transmit Interrupts

The transmit DMA notifies the core each time that an HDLC packet is transferred from the system memory. The THDLCE bit in the CPRInTER register is set and a CPRI_INTy interrupt is generated if the CPRIICRy[EHDLCE] bit is set and the CPRInTER event registers is selected by the ESR field. A control interrupt (CPRIn_tx_control) can be also generated if the bit CPRInTCIER[THDLCEE] bit is set. The interrupt can be pulse or level as determined by the CPRInTCIER[CLEVEL] bit.

18.3.4 Transparent Mode

The CPRI supports transparent mode in both directions (receive and transmit). Each can be enabled independently. When the CPRIn_MAP_CONFIG[CPRIn_MAP_RX_TR_MODE] bit and CPRInRGCM[TM] bit are set, the received CPRI frames are directed to one shared IQ buffer in the system memory through the MBus interface. The buffer location is described by CPRInRACPR0 register and the buffer size is described by CPRInRIQBS register.

When the CPRIn_MAP_CONFIG[CPRIn_MAP_TX_TR_MODE] bit and CPRInTGCM[TM] bit are set, all the CPRI frames (including the control words) are transmitted directly from a single buffer in the system memory. The buffer location is described by CPRInTACPR0 register and the buffer size is described by the CPRInTIQBS register

Note: In transparent mode the buffer size should be aligned to CPRI basic frame size. For example, if the link rate is 3.072 GHz then each word in the basic frame is 5 bytes and

the buffer size must be 80 × X, where X is an integer number. In transparent mode, the HDLC and Ethernet MACs are disabled.

Figure 18-18 on page 18-42 shows how the CPRI frames are stored in the system memory.

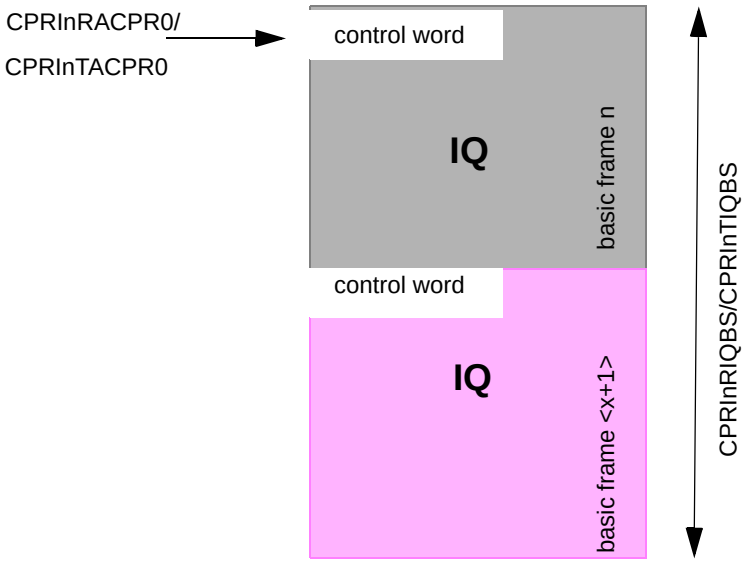


Figure 18-18. CPRI Buffer in the System Memory in Transparent Mode

18.3.5 Chip Rate Mode

Chip rate processing can be performed on uplink and downlink data.

18.3.5.1 Uplink Functional Description in Chip Rate Mode

The uplink data in chip rate mode use an 8-bit sample width with double oversampling. The oversampling data that belongs to the same AxC must be located back-to-back in the same CPRI frame. The CPRI module can separate the oversampling data to two different buffers. Bit ROVS in the CPRInRGCM register determines whether the received data that belongs to the same AxC is separated into two different buffers. In this case, the sample width should be set to 8 (the RSW field in the CPRInRGCM register should be 0x2).

The oversampling data is transferred to system memory and also to MAPLE-B memory. The transactions from the receive DMA memory to system memory and to MAPLE-B memory is done in parallel via different Mbus ports. The transactions toward the MAPLE-B memory are generated when there are 64 bytes of data for each AxC (for 16-byte transactions) whereas the transaction size toward the system memory is programmable by the Receive IQ MBus Transaction Size (CPRInRIQMTS) register. The small MBus transaction size reduces the system latency but it may impact device performance. The RMCM bit in the CPRInRGCM register determines whether the received data is sent in parallel to MAPLE memory and to system memory.

If synchronization between some CPRI links is required, then the CPRInRCR[RIQS] bit should be set for the relevant units, but the data transfer between the Framer and the receive DMA only starts when the GRIQS bit in the CPRIGRSR register is also set. The CPRIGRSR register is common for all the CPRI links, and it should be set only after that all the relevant CPRI lanes are synchronized.

18.3.5.2 AxC Data Buffers

Each receive oversampling antenna carrier is stored in a different buffer mapped on the system memory. The AxC<n> oversampling0 buffer can be located in any location in the system memories and the AxC<n> oversampling buffers are locate in consecutive locations. The antenna carrier <n> buffer base address is determined by the Receive Antenna Carrier Parameter Register n (CPRInRACPRy). The antenna carrier buffer base address should be aligned to 16 bytes.

The oversampling antenna carrier buffer size is identical for all the receive antenna carrier belonging to a CPRI link and is defined by Receive IQ Buffer Size (CPRInRIQBS) register. The antenna carriers buffer size should be aligned to MBus transaction size (CPRInRIQMTS[RIQMTS]).

The antenna carriers have the same sample rate; therefore, the IQ buffers write pointer is identical for all the receive oversampling antenna carriers and is described by the status register Receive IQ Buffer Displacement Register (CPRInRIQBDR). Adding this register to the receive AxC oversampling base address indicates the location to which CPRI writes next in the system memory. The location of AxCn OVS0 is described by Receive Antenna Carrier Base Address (CPRInRACPRy[RACBA]) field and the location of AxCn OVS1 is equal to CPRInRACPRy[RACBA] + CPRInRIQBS[RIQBS] +1.

In addition to a displacement status register, each CPRI module contains a status register that indicates the number of 2560 chips per OVS buffer that are written to the OVS buffers in the device. The status register Receive Chips Counter Register (CPRInRCCR) indicates the number of chips in a resolution of 2560 chips.

In Chip Rate mode, the CPRI complex supports up to 24 AxCs. The number of the AxCs handled by all the CPRI units in chip rate (multicast) mode can be up to 24.

When only one pair is used for chip-rate mode:

- If the pair operates in shared mode (when only one CPRI unit operates in a pair), this unit can handle all 24 AxCs.
- If the pair does not operate in shared mode (both CPRI units can operate in a pair), each unit can handle a maximum of 12 AxCs.

Figure 18-19 describes the AxC data buffer locations in the system memory.

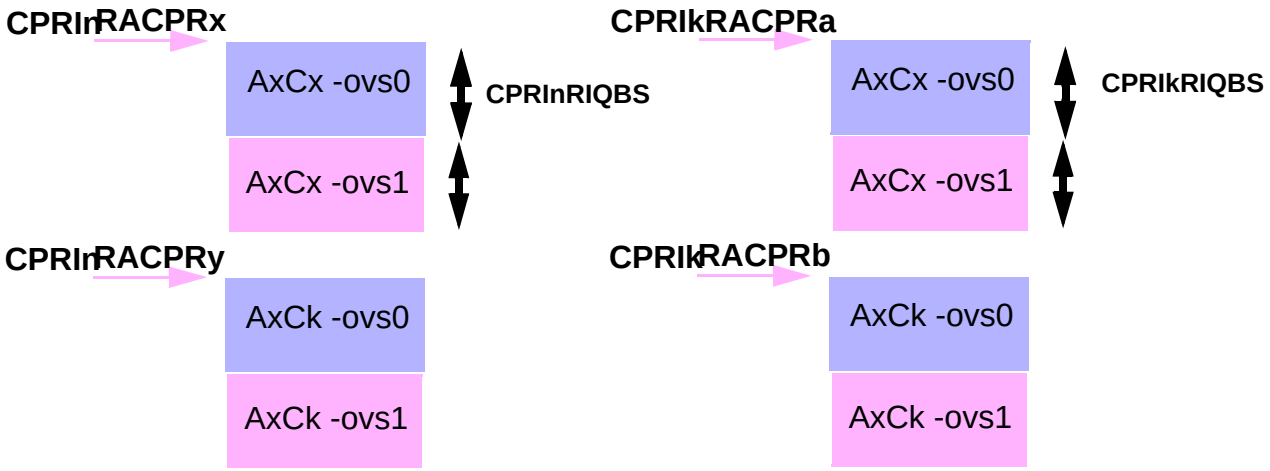


Figure 18-19. AxC Data Buffers Location in the System Memory

In the case in which the RMCM bit of Receive General CPRI Mode (CPRInRGCM) is set, then the received oversampling antenna carriers are also stored in a different buffer mapped in the MAPLE memory. The AxC buffers that belong to the same CPRI link are located in consecutive order in the MAPLE memory where the first active AxC location is determined by the Receive IQ Second Destination Base Address register (CPRInRIQSDBA). This base address in MAPLE must be aligned to 16 bytes.

The OVS antenna carrier buffer size is identical for all the receive antenna carrier belonging to a CPRI link and is indicated by the Receive IQ Second Destination Buffer Size register (CPRInRIQSDBS).The AxCn ovs buffer size must be aligned to 64 bytes.

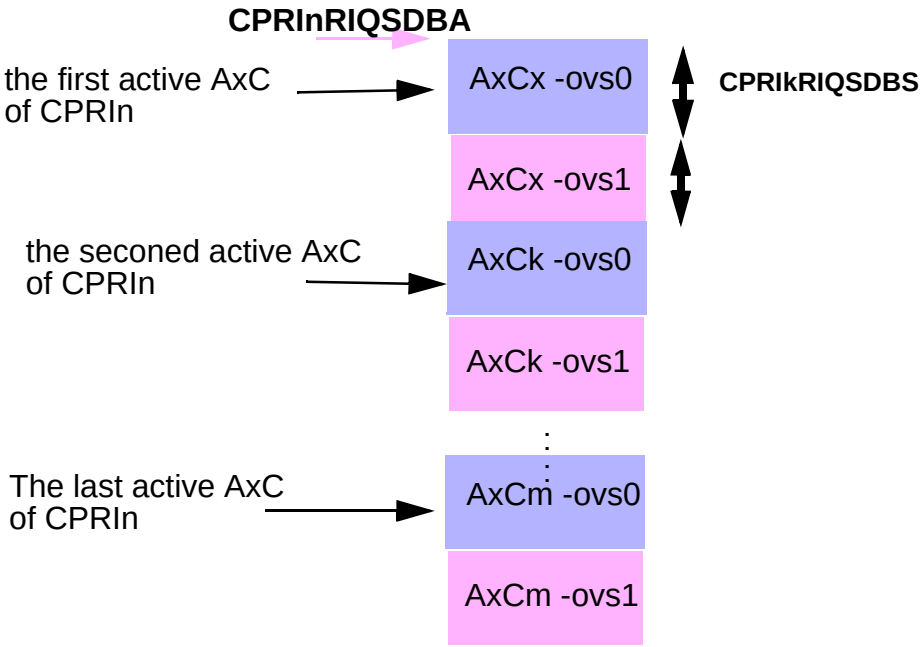


Figure 18-20. AxC Data Buffer Locations in the MAPLE Memory

Figure 18-20 shows how the oversampling data is stored in the MAPLE buffers in chip rate mode. Note the sample width is 8 bits. **Figure 18-21** shows the AxC oversampling buffer in the MAPLE-B.

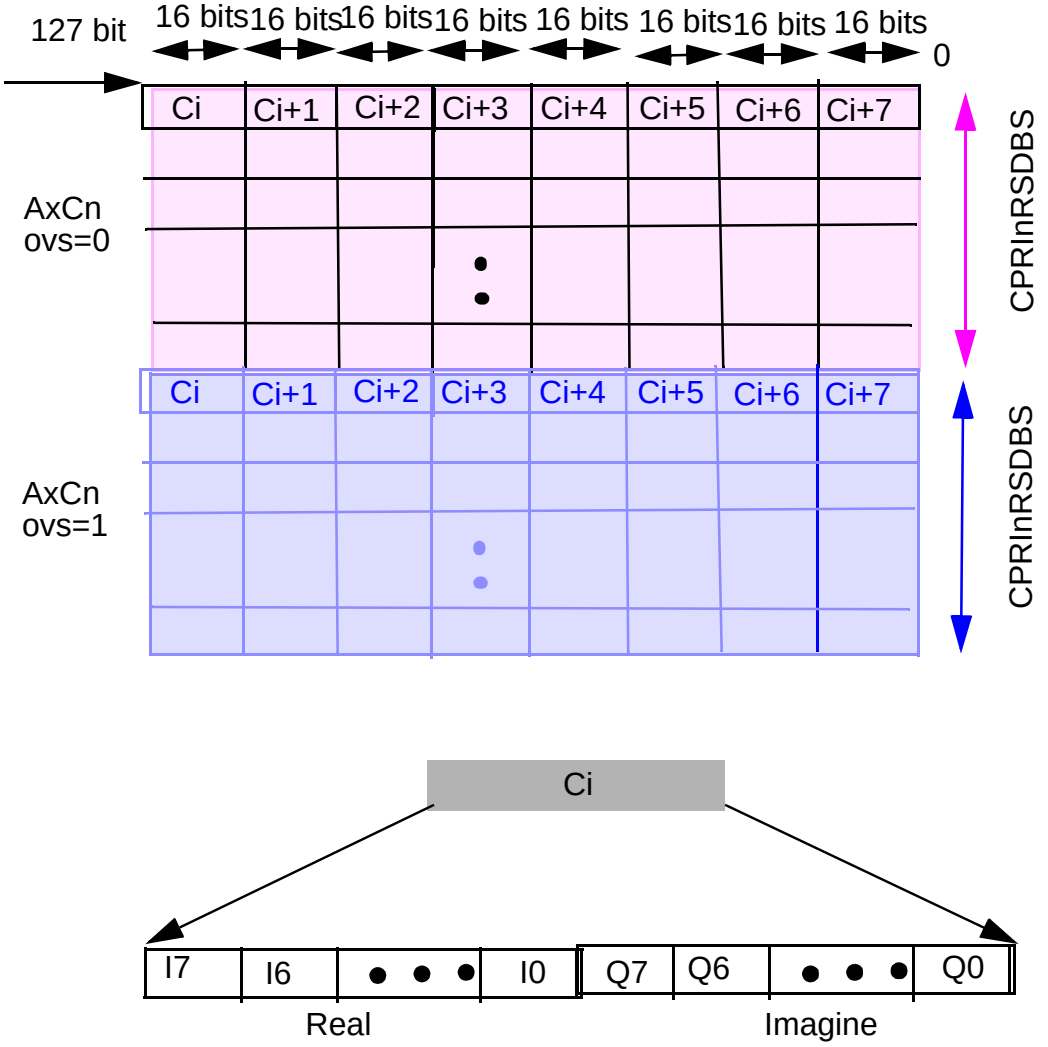


Figure 18-21. Antenna Carrier Oversampling Buffer

The CPRI can notify the cores each time that it transfers <x> bytes to the buffers or each time it fill the buffers up to threshold levels. The number of bytes is determined by the Receive IQ Threshold register (CPRInRIQT) and the threshold levels defined by CPRInRIQFT and CPRInRIQST registers.

18.3.5.3 Uplink Delay in Chip Rate Mode

The uplink delay in chip rate mode is determined based on the delay of the receive buffers located in the CPRI module.

- Receive IQ shared memory (buffer inside the CPRI framer). The delay of the IQ shared memory is 4 chips
- Receive DMA memory. The latency of this memory depends on the transactions size.
  - Latency toward the MAPLE buffers. The transaction size toward the MAPLE is 64 bytes, therefore the latency is 32 chips
  - Latency to ward the SOC buffers. The transaction toward the DSP system is configurable by the Receive IQ MBus Transaction Size (CPRInRIQMTS) register therefore the delay is RIQMTS*32
- The maximum uplink latency toward the MAPLE in chip rate mode: 4 + 32 = 36 chips
- The maximum uplink latency toward the DSP system in chip rate mode is: 4 + RIQMTS*32

18.3.5.4 Downlink Functional Description in Chip Rate Mode

The transmit DMA data flow in chip rate mode is identical to the flow in normal mode, but some configurations and the order of enabling is more limited. Downlink can be done using one or more CPRI lanes. When multiple lanes are used, each CPRI controller transmits part of the AxCs, but they read the data from the MAPLE-B2 at the same time. For downlink in chip-rate mode, there is only one IQ sample for each AxC in a basic frame.

The functional description is summarized as follows:

- If more than one CPRI unit is required, all the units should use should use the same Node B frame (10 ms) sync signal. It can be done using the shared mode of the timer. In this case, the pulse is generated by the CPRI1 timer. The CPRI1 sync signal is connected to the other relevant CPRI modules if the CPRInTMRC[SSM] bit is set. The CPRI1 sync signal determines the start of the transmit Node B frame.
- All relevant units should have the same values in the CPRIn_MAP_TX_OFFSET and CPRI_START_TX_OFFSET registers. The value in CPRIn_MAP_TX_OFFSET should be 16 basic frames bigger than the value in the CPRIn_START_TX_OFFSET register.
- The CPRInTIQMTS of all the relevant lanes should be configured to 1.
- All relevant CPRI units should be configured except for the CPRInTCRs. After the MAPLE CPRE prepares the data to be transmitted and fills up its buffers, the TCR of all relevant lanes should be set together. Both the CPRInTCR[TIQE] and CPRInTCR[TIQSYNC] bits should be set.
- Set the relevant bit in the CPRIGTSR one HyperFrame after all CPRInTCRs are enabled. It should occur at least one HyperFrame before reaching the offset value configured in the CPRIn_START_TX_OFFSET register. This can be done by reading the offset value in the

PCRInTXHEN register and checking whether the TX_OFFSET value is within 1 HyperFrame from the offset value configured in CPRIn_START_TX_OFFSET register. If the distance is not at least 1 HyperFrame, the relevant bit in CPRIGTSR should be set later when there is a distance of at least 1 HyperFrame.

- The interrupt to the MAPLE-CRPE indicting that the CPRI started to fetch new data from the MAPLE-CRPE can be generated by any of the relevant CPRI units. The interrupt can be enabled when the CPRIICRy[EIQE] bit is set and the TYER is selected in the ERS field.
- The maximum number of AxCs that can be handled by all the CPRI units is 16.

18.3.5.5 Down Link Latency.

The transmit latency is determined by the latency of the transmit DMA buffers and the transmit buffers inside the CPRI framer module.

- Transmit IQ shared memory (buffer inside the CPRI framer). This delay is configurable and it is determined by the Transmit CPRI Framer Buffer Size (CPRInTCFBS) register. The minimum delay in chip rate mode is 16 chips.
- Transmit DMA memory. The latency imposed by this memory depends on the transmit transactions size; in chip rate mode, the transaction size is 64 bytes and, therefore, the delay is 16 chips.

Latency from the MAPLE buffers to the CPRI frame in chip rate mode = 16 + 16 = 32 chips.

18.3.6 External Implementation

The CPRI complex implements the external interface with 6-high speed serialized links. Each link is connected to a single SerDes lane and they all share the same PLL. These links support 1228.8 MHz, 2457.6 MHz, 3072.0 MHz, 4912.2 MHz, and 6144.0 MHz line rates. The CPRI complex supports the following topologies:

- Single point-to-point link between one REC and one RE (Figure 3 in the CPRI Specification V4.1)
- Single point-to-point link between one REC and one RE (Figure 4 in the CPRI Specification V4.1)
- Multiple point-to-point links between one REC and several REs—Star topology (Figure 5in the CPRI Specification V4.1)
- Chain topology (Figure 5A in the CPRI Specification V4.1)
- Tree topology (Figure 5B in the CPRI Specification V4.1)
- Multiple point-to-point links between several RECs and one RE (Figure 5D in the CPRI Specification V4.1)
- Chain topology of multiple RECs (Figure 5E in the CPRI specification V4.1)

Figure 18-22 describes the CPRI complex system when minimum frames skew is required between the REC links. In order to promise full synchronization between the CPRI links, the connect the Timer1 output to all the CPRI Framer modules.

18.3.6.1 Star Topology

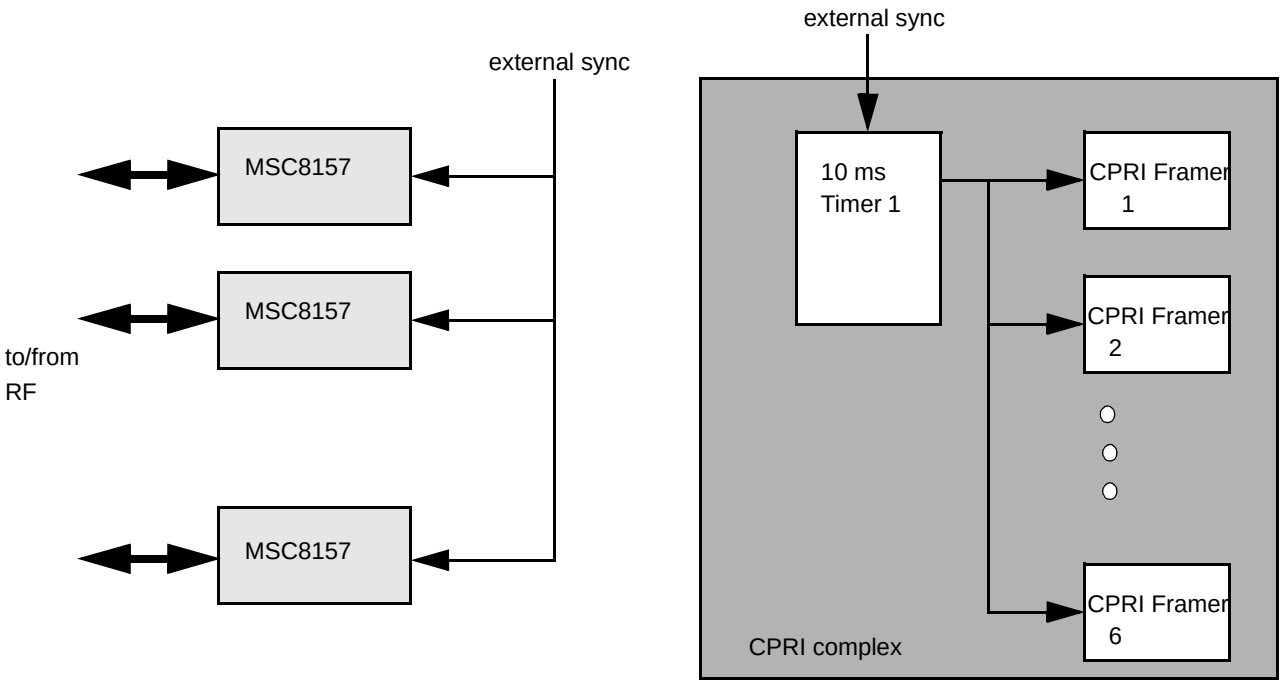


Figure 18-22. Star Topology When Minimum Frame Skew Is Required

18.3.6.2 Chain Topology

The daisy chain topology is supported by the auxiliary ports of the CPRI Framer modules and by the auxiliary modules. The CPRI complex contains three pairs of CPRI units; each pair can be used in auxiliary mode. **Figure 18-23** describes the down link data flow. The “master” device (the last in the chain and the most far from the RE) starts to transfer the CPRI downlink frame where each MSC8157E device in the chain can insert AxC data in any segment (timeslot) in the CPRI frame. The control words are always controlled by the master device. The master device generates the 10 ms pulse of the chain, and both CPRI units of the REC slave in the chain generate their 10 ms pulse using the received 10 ms sync signal of the other CPRI unit in the pair. The REC master can reset the RE using the Reset bit in the #Z.130.0 according to the standard. The reset request from the REC master for the slave REC either can be sent externally (not using the CPRI standard) or can be resolved in software; the software of the REC slave device should poll the Reset bit in #Z.130.0 once in a hyperframe to determine whether the REC master has issued a reset.

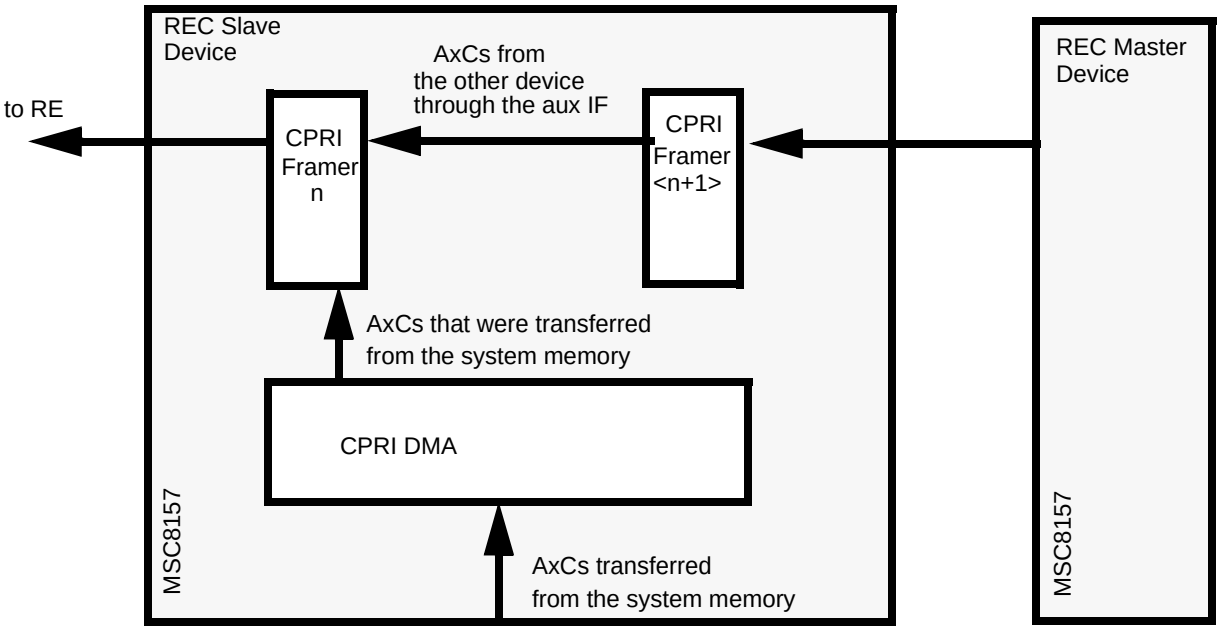


Figure 18-23. Down Link Data Flow in Daisy Chain Topology

The mask registers define for each bit in the frame whether the transmit data should be driven by the auxiliary port or by the CPRI Framer itself. Figure 18-24 describes how the CPRI Framer uses the mask registers.

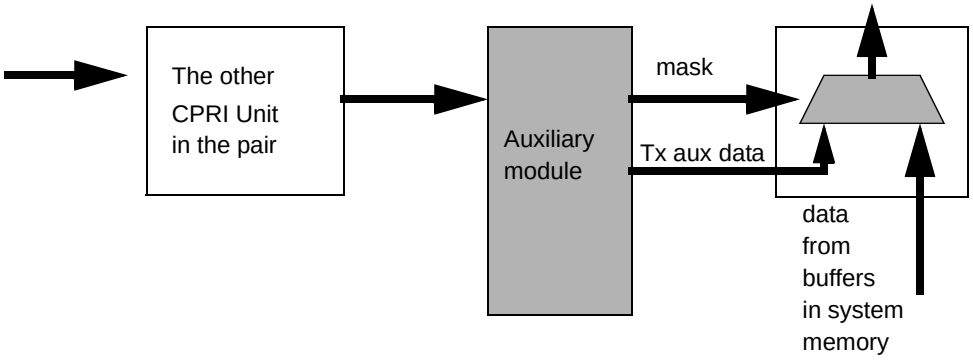


Figure 18-24. Mask Registers Inside the CPRI Framer

Figure 18-25 describes the uplink data flow at daisy chain configuration. The received CPRI frame arrives via the CPRI Framer and the selected antenna carriers and the control data are transferred to the system memory via the MBus bus. In parallel the received CPRI frame is transferred via the receive auxiliary port to the other CPRI Framer in the pair and is transmitted externally to another device in the chain.

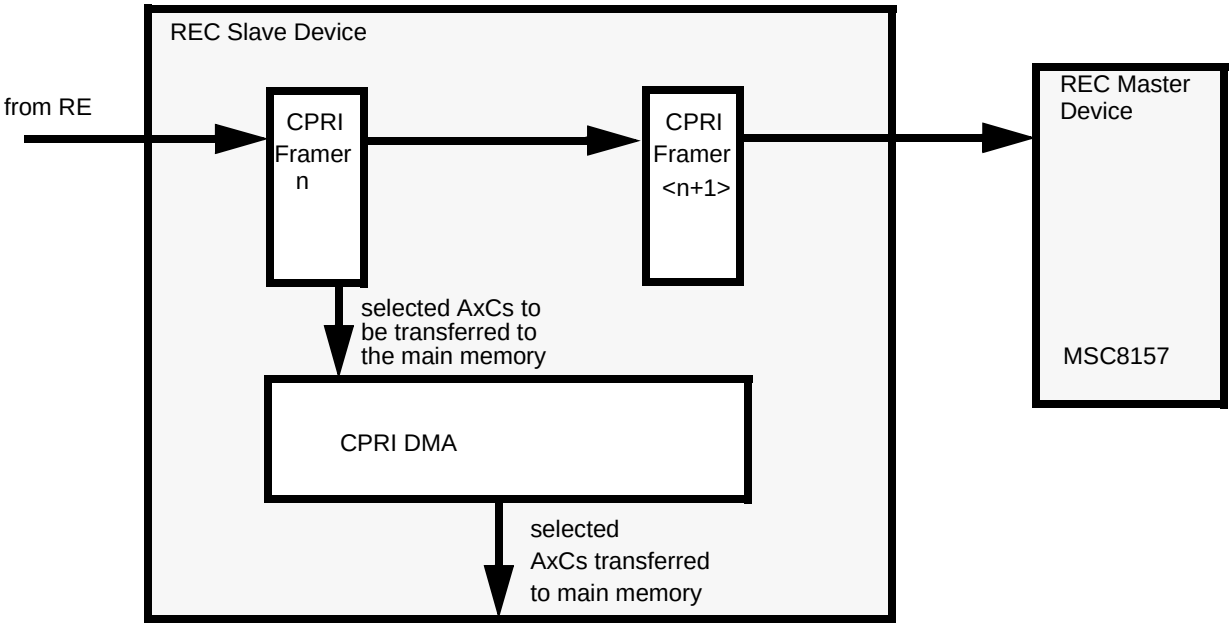


Figure 18-25. Uplink Data Flow in Daisy Chain Configuration

Note: For lanes that operate as pairs in auxiliary mode, when they generate an Error interrupt due to Local Loss of Frame error (bit CPRInEER[LLOF] is set, see **Section 18.4.4.17**), software should disable the CPRInAUXCR[AUX_EN] bit for both lanes of the pair and then immediately reenable both lanes.

18.3.7 CPRI Double Sampling Rate

A single CPRI module supports only one single sampling rate. To support a double sampling rate, three CPRI modules are required. The first CPRI is connected to the RF (CPRI 0 in the figure) and it transfers the AxCs with sampling rate <x> to the system memory. The second CPRI (CPRI 1 in the figure) module receives the RF data from the first CPRI module via the auxiliary interface and it outputs the data directly via the SerDes to the third CPRI module.The third CPRI module transfers the AxCs with sampling rate <y> to the system memory.

In the transmit direction, the third CPRI module fetches the AxCs with sampling rate <y> from the system memory and it transfers the data to the second CPRI module via the SerDes interface. The second CPRI module transfers the transmit data via the auxiliary interface to the first CPRI module which outputs the data toward the RF.

Each CPRI module contains a receive configuration table that determines which AxCs to transfer to the system memory and the AxCs location in the basic frame. The receive configuration table is programable.

Each CPRI module contains a transmit configuration table that determines which AxCs to fetch from system memory and the AxCs location in the basic frame. The transmit configuration table is programable.

Each CPRI module contains receive CPRI mask registers with the masking bits of the auxiliary interface. These registers should be configured for CPRI 0 in the example. If a masking bit is set then the auxiliary data (coming from CPRI 1) is sent out to the RF and if this bit is cleared then the data read by CPRI 0 from the system memory is sent out.

The synchronization between the different CPRI modules is done via internal sync signals (General Transmit Synchronization Register). **Figure 18-26** describes the system connectivity in double sampling rate.

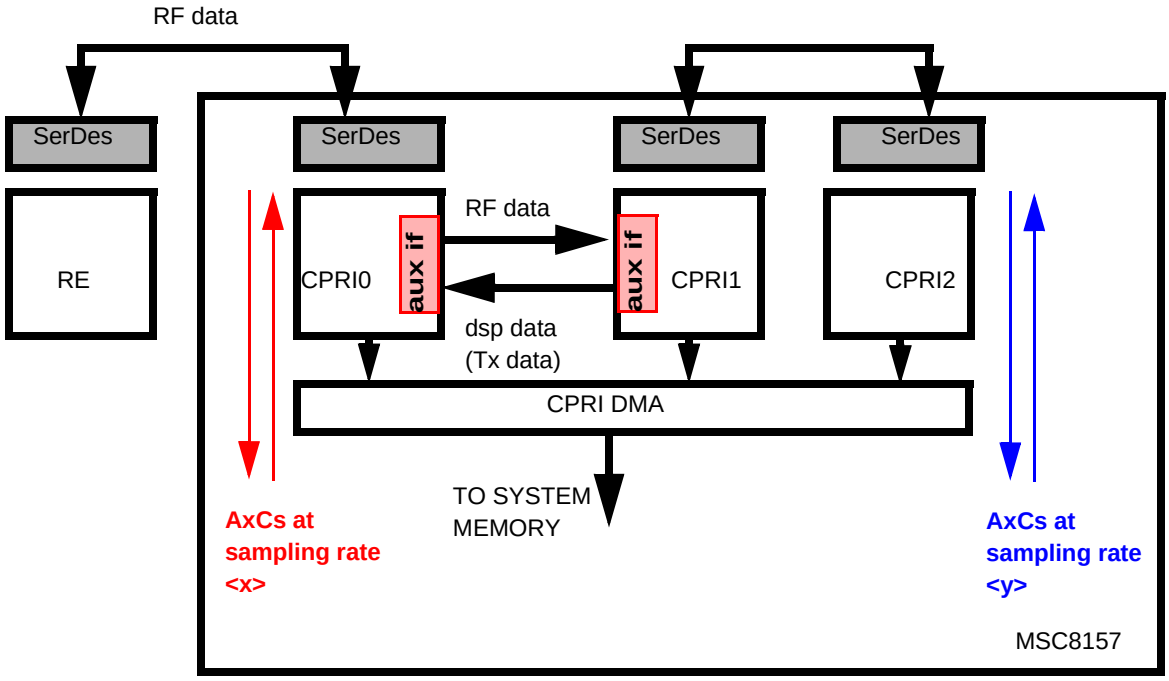


Figure 18-26. CPRI Connectivity in Double Sampling Rate

18.3.8 Timers

Each CPRI Framer has a local timer which generates 10 ms sync signal used by the CPRI Framer to transmit the CPRI frame. The CPRInTMRC register determines the CPRIn timer operation mode. When a minimum skew is required between the CPRI links, then the tsync signal of timer 1 is connected to CPRI Framers 2–6. Field SSM determines whether the CPRI complex works in shared mode or in independent mode.

The timer can generate the 10 ms sync signal in one of the following three modes, depending on field ESA:

- Sync signal generated locally by the timer

- Sync signal is aligned to 10 ms external sync signal and synchronized to Framer Clock. In this case, the maximum skew between the external sync signal and the generated sync is one Framer Clock cycle.
- Sync signal is aligned to received 10 ms sync signal of the CPRI Framer Unit in the pair.

The input sync signal is expected to be a minimum of one Framer Clock period wide, and asserted high. Asynchronous sync signals are double re synchronized to the Framer Clock domain. There is a rising edge detect circuit and an offset generator that can offset the sync signal by up to +16 Framer Clock cycles, or it can offset the sync by –255 Framer Clock cycles. The offset value is defined by SDLY field and its direction (+ or –) by the ESA field. The timer sync_out signal can be driven out externally (outside the device) when the TSO bit is set.

Figure 18-27 describes the frame sync signal when the delay is two cycles. **Figure 18-28** describes the frame sync signal when the delay is negative and is two cycles.

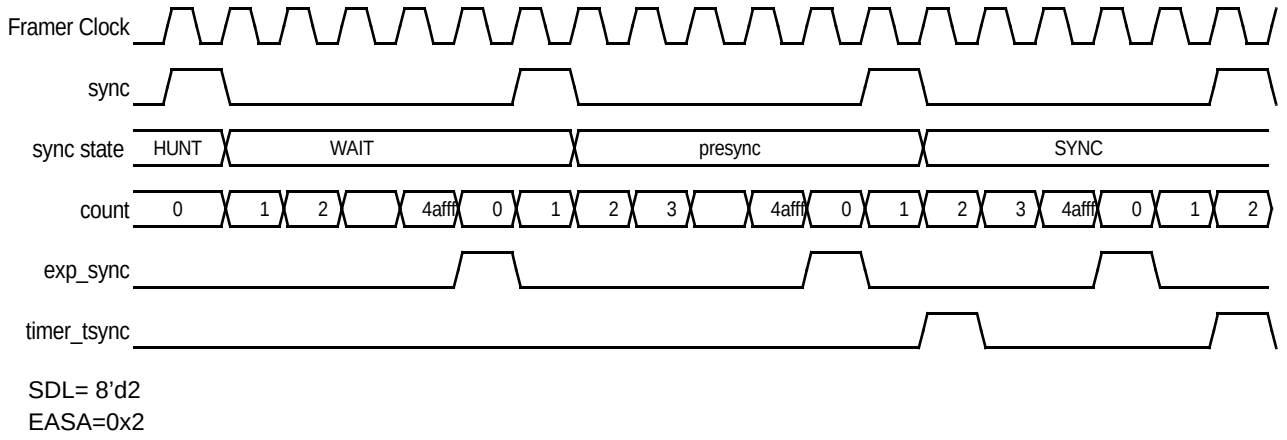


Figure 18-27. Sync Signal Offset is 2 Cycles

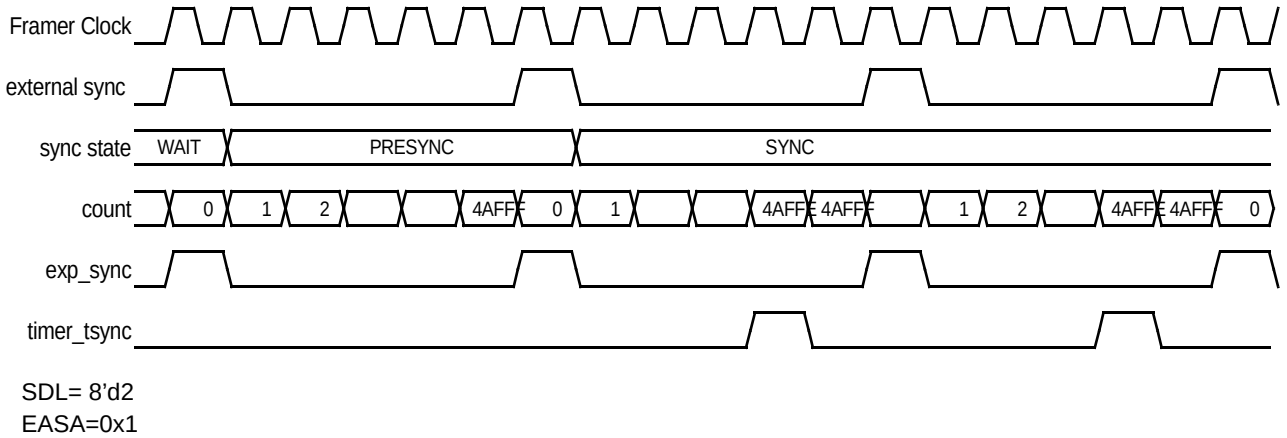


Figure 18-28. Sync Signal Offset Is Negative 2 Cycles

The local timer is clocked by Framer Clock and the number of clock cycles during 10 ms is depends on the link rate as describes in **Table 18-10**.

Table 18-10. The Counter Values as a Function of the Link Rate

Link Rate	Number of Cycles During 10 ms
1228.8 MHz	307200
2457.6 MHz	614400
3072.0 MHz	768000
4915.2 MHz	1228800
6144.0 MHz	1536000

18.3.8.1 Timer Sync State Machine

The timer state machine is required when the sync signal is not generated locally. The state machine is enabled only if the SSME bit in register CPRInTMRC register is set. The timer state machine state is described by the CPRInTMRSR status register. When sync synchronization is lost, then the EXSSLOS bit in the CPRInEER is set and an error interrupt is generated if the CPRInEIER[EXSSLOSE] bit is set.

Following is a list of the states of the sync state machine:

- HUNT (00). A sync event is constantly sought. As soon a sync event is detected, the state machine changes to a WAIT state. During the Hunt state, 10 ms sync signal is not generated.
- WAIT (01). At least one sync has been detected. The next sync event should be accepted after 10 ms.If the sync appears in the correct position after 10 ms the state changes to PRESYNC state. If the sync doesn’t appear the state returns to HUNT. Note that when the input sync is an asynchronous signal (when the ESA field = 01 or 10), then a mistake of up to 2 cycles is accepted. During the WAIT state sync signal is not generated.
- PRESYNC (11). Two sync events have been detected and the distance between the sync signals is 10 ms. If the sync event is recognized not after 10 ms the state returns to the WAIT state. Otherwise, the machine transfers to the SYNC state. During this state sync signal is not generated.
- SYNC (10). At least 3 sync events 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 earlier NUMERR times, the state changes to the HUNT state (00). During the SYNC state, sync signal is generated according to SDLY field of CPRInTMRC.

Figure 18-29 shows the timer state machine

Note: When the input sync is an asynchronous signal, then a mistake of up to two cycles (for each direction) is accepted.

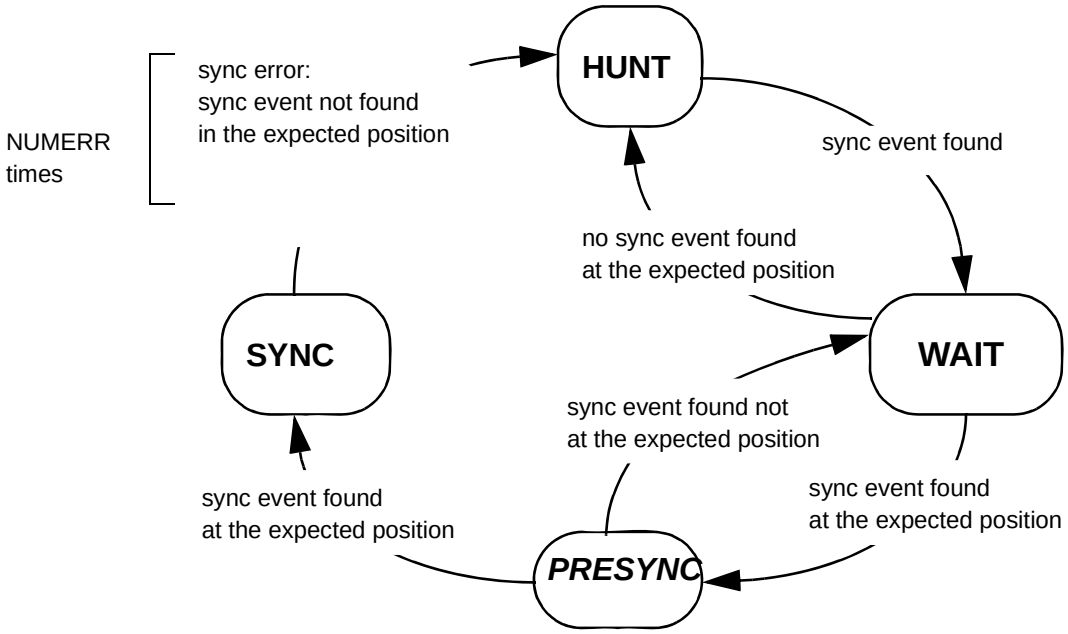


Figure 18-29. Sync State Machine

18.3.9 Interrupts

Table 18-11 shows the different kind of interrupts generated by the CPRI.

Table 18-11. CPRI Interrupts

Interrupt Name		Connectivity in MSC8157E	Description
IPI_INT_CPRIn_INTy	1–12	DSP	These CPRI interrupts are generated during normal operation as a result of any event described in the CPRInRER or CPRInTER registers. The CPRIICRy[ERS] field selects the receive or transmit event register and bits [8–0] of the register enable the interrupts. The interrupt can be generated as pulse or level as determined by the CPRIICRy[LEVEL] bit. **Figure 18-30** describes the CPRI interrupts scheme.
IPI_INT_CPRIn_RX_CONTROL	1–6	GCR	The receive control interrupt can be generated as result of the following events: • Receive Ethernet event (CPRInRER[RETHE] bit is set) • Receive HDLC event (CPRInRER[RHDLCE] bit is set) • Receive VSS event (CPRInRER[RVSSE] bit is set) The interrupt is enabled if at least one of the bits of CPRInRCIER[2:0] register is set. The interrupt can be generated as pulse or level and it is determined by the CPRInRCIER[CLEVEL] bit.

Table 18-11. CPRI Interrupts (Continued)

Interrupt Name		Connectivity in MSC8157E	Description
IPI_INT_CPRIn_TX_CONTROL	1–6	GCR	The transmit control interrupt can be generated as result of the following events: • Transmit Ethernet event (CPRInTER[TETHE] bit is set) • Transmit HDLC event (CPRInTER[THDLCE] bit is set) • Transmit VSS event (CPRInTER[TVSSE] bit is set) The interrupt is enabled if at least one of the bits of CPRInTCIER[2:0] register is set. The interrupt can be generated as pulse or level and it is determined by the CPRInTCIER[CLEVEL] bit.
IPI_INT_CPRIn_RX_FRAME_TIMING	1–6	GCR	Receive BFN/ hyper frame timing interrupt. These interrupt is asserted each time that new hyperframe or BFN frame arrives. These events are indicated by CPRInRER[RBFNE] bit and by CPRInRER[RHFNE]. The interrupt is enabled if CPRInRCIER[RBFE] bit or CPRInRCIER[RHFE] bit is set. The timing interrupt can be pulse or level and it is determined by the CPRInRCIER[RLEVEL] bit
IPI_INT_CPRIn_TX_FRAME_TIMING	1–6	GCR	Transmit BFN/ hyper frame timing interrupt. These interrupt is asserted each time that new hyperframe or BFN frame arrives. These events are indicated by CPRInTER[TBFNE] bit and by CPRInTER[THFNE]. The interrupt is enabled if CPRInTCIER[TBFE] bit or CPRInTCIER[THFE] bit is set. The timing interrupt can be pulse or level and it is determined by the CPRInTCIER[TLEVEL] bit
IPI_INT_CPRI_RX_CPU	1	VSG	CPRI receive CPU control interrupt The receive CPU control interrupt can be generated as result of the following events of any of the CPRI units depending on register CPRIRCCIER. • Receive Ethernet event (CPRInRER[RETHE] bit is set) • Receive HDLC event (CPRInRER[RHDLCE] bit is set) • Receive VSS event (CPRInRER[RVSSE] bit is set) The interrupt can be generated as pulse or level and it is determined by the CPRIRCCIER[RCPULEVEL] bit.
IPI_INT_CPRI_TX_CPU	1	VSG	CPRI transmit CPU control interrupt The transmit CPU control interrupt can be generated as result of the following events of any of the CPRI units depending on register CPRITCCIER. Transmit Ethernet event (CPRInTER[TETHE] bit is set) Transmit HDLC event (CPRInTER[THDLCE] bit is set) Transmit VSS event (CPRInTER[TVSSE] bit is set) The interrupt can be generated as pulse or level and it is determined by the CPRITCCIER[TCPULEVEL] bit.
IPI_INT_CPRI_ERR	1	GCR	CPRI error interrupt This interrupt is asserted if an error occurred in any CPRI module and if its corresponding bit is set in the Mask Register CPRInEIER. Error Status Register CPRIESR indicates which CPRI the error comes from The error events of CPRIn are indicated by the status registers CPRInEER. This interrupt is level.

Table 18-28 describes the CPRI_INTy generation.

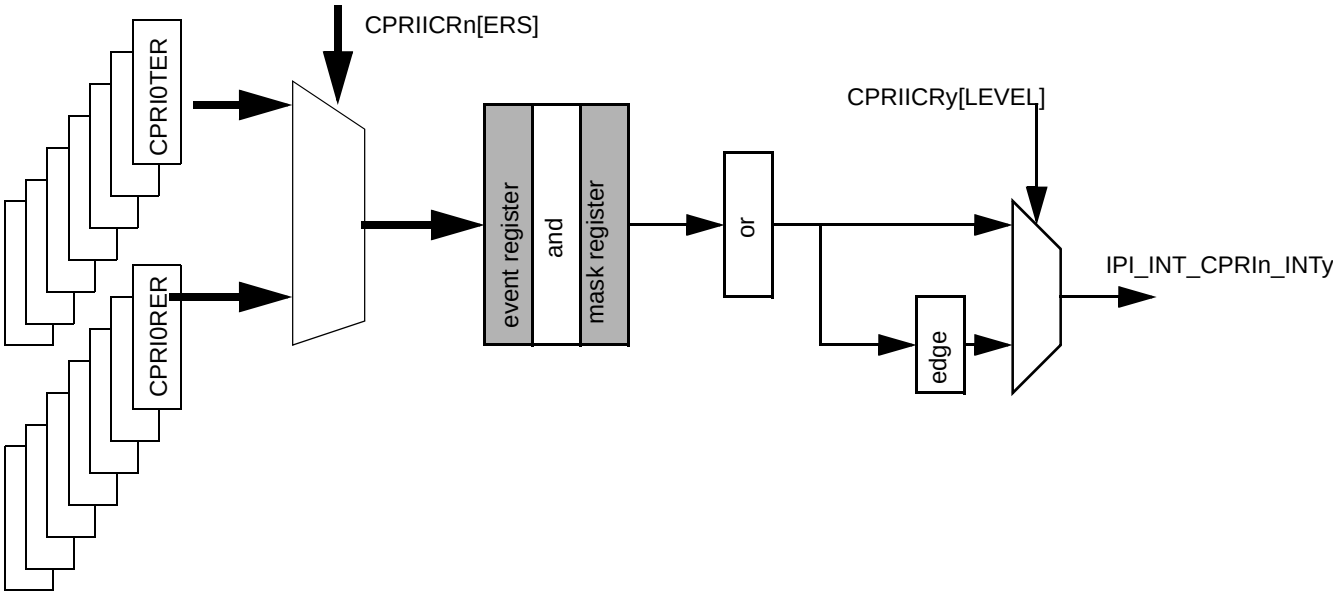


Figure 18-30. IPI_INT_CPRIn_INTy Interrupt Generation

18.3.10 L1 Inband Protocol Errors

Bits of control word Z.130.0 are used for basic Layer 1 functions. The following bits when set in the Receive Control word Z.130.0 can generate an error interrupt depending on the configuration in the CPRInEIER register:

- Bit “1” reflects RAI (Remote Alarm Indication)
- Bit “2” reflects SDI (Remote SAP Defect Indication)
- Bit “3” reflects LOS (Remote Loss of Signal Indication)
- Bit “4” reflects LOF (Remote Loss of Frame Indication)

Bit “0” of control word Z.130.0 reflects Remote Reset Request. It can be set through CPRIn_HW_RESET register.

When the REC sends a reset request to another REC, the second REC can poll this bit in the Control Table from time to time (for example, once in a hyperframe) and initiate an autonegotiation reset.

The REC master can send a reset request to the REC slave or RE through the CPRIn_HW_RESET register. The REC slave cn send a reset acknowledge to the master, but this configuration has the following restraints:

- If the REC slave is an end-point, it can generate the reset acknowledge.

- If the REC slave is not an end-point (but between the REC master and the RE), it must be configured as a master (to support the 10 ms chaining) and the reset request/ack must be managed outside the CPRI protocol.

18.3.11 CP_LOS

There are six CPRI loss-of-signal (LOS) input signals coming from externally-connected smallform-factor pluggable (SFP) optical transceivers that the SFP device uses to indicate that it has lost the CPRI input signal. Receipt of the CPRI LOS signal causes the framer to lose synchronization and can generate a CPRI error interrupt if enabled by the CPRIEIER[LLOSE] bit.

18.3.12 Delay Measurement and Calibration

Measurement and control of the signal delay is required for appropriate recovery of the synchronization references and system configuration. The MSC8157E CPRI provides internal support for accurate delay measurement and calibration. The RX delay measurement and the TX delay calibration values are accessible via the register interface. The CPRI standard contains a number of delay accuracy requirements (see **Sections 3.5** and **3.6** of the CPRI specification), most of which relate to the physical layer functions. The MSC8157E CPRI includes built-in PCS sub-layer functions to ensure meeting the tough requirements. A simple RE networking configuration can use the AUX interface to configure the frame timing to match the CPRI specification requirements.

A basic CPRI configuration has a single REC node and a single RE node with a single link connecting the REC master port to the RE slave port. This is called a single-hop point-to-point connection. As noted in **Section 18.3.6**, *External Implementation*, however, the modules support a number of connection topologies. While the star topology is similar to the basic configuration in that all links are between two nodes, the difference is that it has multiple master ports in the REC node. All the other topologies are considered multi-hop configurations. A multihop configuration is one which has continuously connected hops from the REC that end with a specific RE, but includes all the nodes in between. Delay budgets must be configured for each hop (both for upload and download).

The CPRI standard defines reference points for frame timing and delayed measurements, as shown in **Figure 18-31** (adapted from the CPRI specification. shows the frame timing sequence of these values for a single-hop interface.

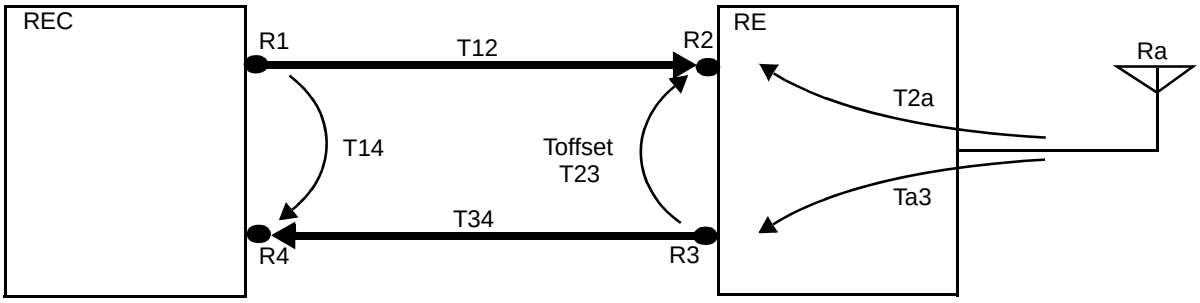


Figure 18-31. Single-Hop Frame Timing Reference Points

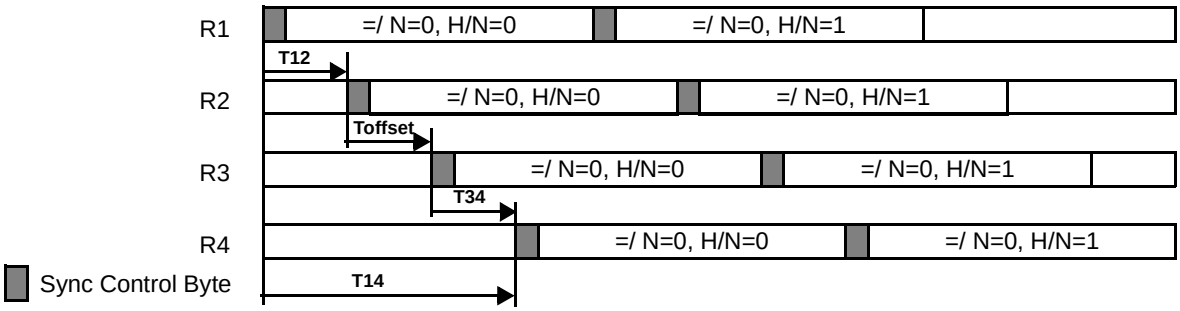


Figure 18-32. Single-Hop CPRI Frame Timing Diagram

The T14, Toffset, T12, and T34 delays can be calculated using the following equations:

- T14 = T_TX = CPRI_ROUND_DELAY * TCLK – (T_T4 + T_EXT + T_R1)
- Toffset(RE) = T23 = T_T4 + T_EXT + T_R1
- T12 = T34 = (T14 – Toffset) / 2

  where,

  — CPRI_ROUND_DELAY is the total round delay from start of internal transmit radio frame to start of internal receive radio frame in the REC node. The value for this variable is measured by the CPRIn_ROUND_DELAY registers (see **Section 18.4.1.10**, *CPRI Round Trip Delay (CPRIn_ROUND_DELAY)*).

  — TCLK is the period of the clock at line_rate/40

  — T_T4 is the transmit delay from the 10 ms synchronization radio frame input signal to the TX AUX counter outputs. See **Section 18.3.12.1**, *CPRI Transmission Delay*.

  — T_EXT = T_EXT_RX + T_EXT_TX—Total external delay is the sum of the SerDes receive delay and the SerDes transmit delay. Optical delays are outside the MSC8157E, and are not compensated. They should be regarded as an extra inaccuracy in the calculation. If these delays (SFP_RX_{REC}, SFP_TX_{REC}, SFP_RX_{RE}, SFP_TX_{RE}) are known, add them to T_EXT to reduce the inaccuracy of the total calculation. These values are stored in the PCVTRCPRInCR0 for T_EXT_RX and PCVTRCPRInCR1

for T_EXT_TX. See **Section 15.10.50**, *CPRIn PCVTR Control Register 0(PCVTRCPRInCR0)* and **Section 15.10.51**, *CPRIn PCVTR Control Register 1(PCVTRCPRInCR1)* for register details.

— T_R1 is the receive delay from the SerDes receive interface until the data is presented to the RX AUX interface. See **Section 18.3.12.2**, *CPRI Reception Delay*.

For these equations, you must consider different values for REC and RE. For example, Toffset (T23) is the total delay in the RE from the SerDes receive to SerDes transmit. This delay is computed using the following formula:

$$T\_R1_{TE} + R\_T4_{RE} + T\_EXT\_RX_{RE} + T\_EXT\_TX_{RE}$$

where, $T\_EXT\_(RX,TX)_{RE}$ are the SerDes delays.

Different REs may use different value for Toffset. The REC node should know the value of Toffset for each RE in advance (for example, by using a predefined value or by having the RE inform the REC by higher layer message).

The total REC SerDes delay is as follows:

$$T\_EXT_{REC} = T\_EXT\_RX_{REC} + T\_EXT\_TX_{REC}.$$

where, $T\_EXT\_(RX, TX)_{REC}$ are the SerDes delays.

Because optical delays are external to the MSC8157E, they do not have compensation and should be regarded as an extra inaccuracy in the calculation. If the delays (SFT_RX_{REC}, SFP_TX_{REC}, SFP_RX_{RE}, and SFP_TX_{RE}) are known, they can be added to T_EXT to reduce the inaccuracy of the total calculation.

Use the following equation according the CPRI specification to calculate T34 and T12:

$$T34 = T12 = (T14 – Toffset) / 2$$

The delay accuracy is discussed in **Section 18.3.12.3**, *Delay Accuracy*,

18.3.12.1 CPRI Transmission Delay

The Transmission delay consists of the innate existing internal delay plus the delay the user can add. The user delay is added mainly to compensate for line delay change (for example, due to temperature changes).

From the start of the transmit 10 ms RF pulse (CPRIn_tx_rfp) there is a delay of T_TX to the start of radio frame (K28_5 symbol) on the output to the SerDes.

- The existing internal delay is equal to 6.75 *T_CLK where T_CLK is a period of clock at line_rate/40

- The user defined delay is configured in the TX_EX_DELAY field of the CPRIn_EX_DELAY_CONFIG register (see **Section 18.4.1.11**, *CPRI Extended Delay Measurement Configuration (CPRIn_EX_DELAY_CONFIG)*). It enables a bit level delay calibration. The delay calibration field can be set between 0 and 39 bit steps. Default calibration on reset is set to zero.

The total delay is: T_TX = 6.75 *T_CLK + TX_EX_DELAY

Figure 18-33 shows the CPRI transmit delay path.

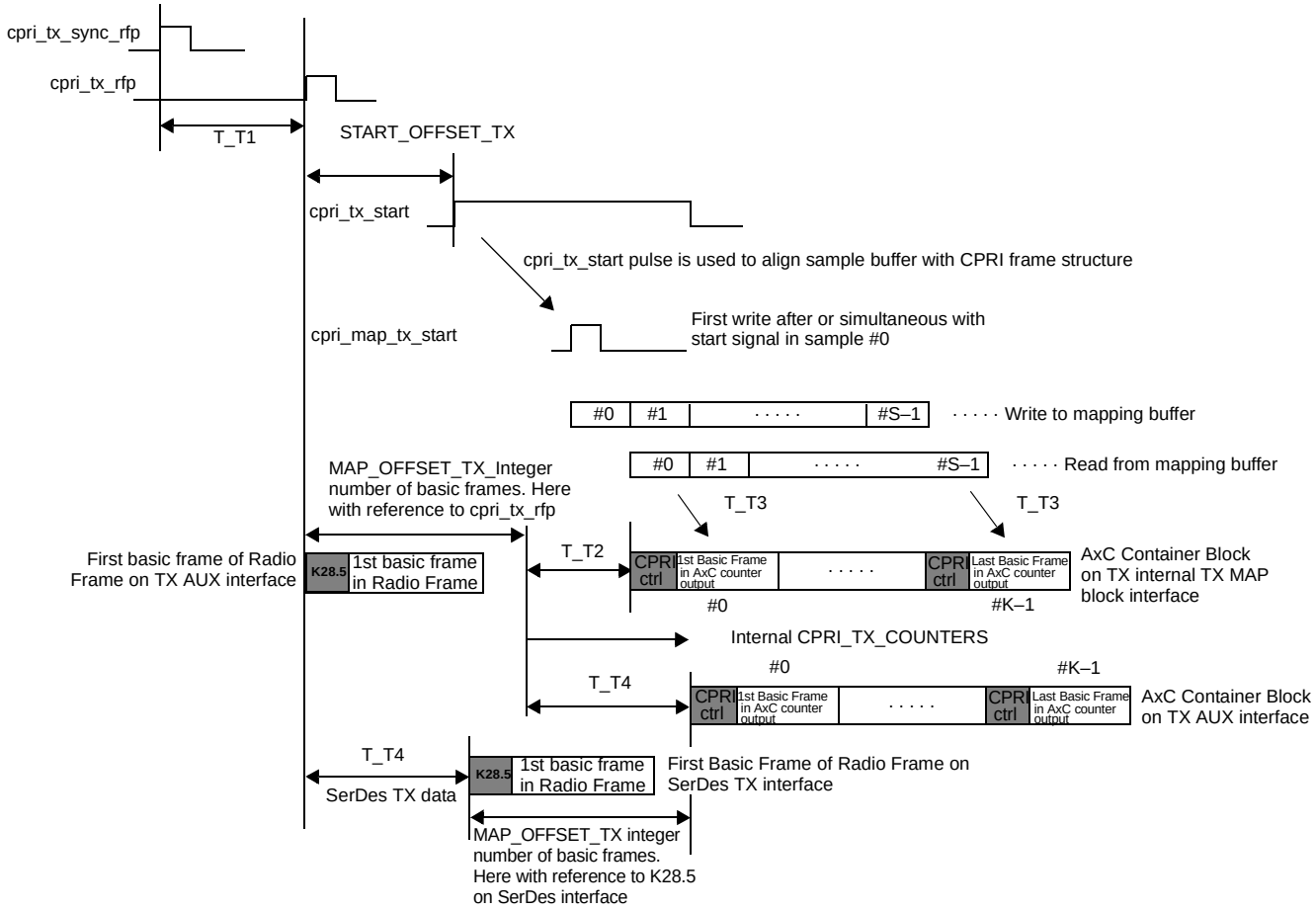


Figure 18-33. CPRI Transmit Delay Path

18.3.12.2 CPRI Reception Delay

The delay from the first serial bit at the receive interface is given by the parameter T_RX. The following equations are used to calculate the T_RX value:

T_RX = (T_RX_BIT_ALIGN + T_8B10B_DEC + T_RX_BUF + T_RX_PHASE + T_RX_BYTE_ALIGN + T_RX_PIPELINE) where

— T_RX_BIT_ALIGN = 2 * T_CLK + CPRIn_RX_DELAY[RX_ALIGN_DELAY] * T_CLK / 40

— T_8B10B_DEC = T_CLK
— T_RX_BUF + T_RX_PHASE is measured in register CPRIn_EX_DELAY_STATUS and computed by: (RX_EX_BUF_DELAY * T_CLK) / N
— T_RX_BYTE_ALIGN = CPRIn_RX_DELAY[RX_BYTE_DELAY] / 4 *T_CLK
— T_RX_PIPELINE = 6 * TCLK
— T_CLK is period of clock at line_rate/40
— N is the value configured in CPRIn_EX_DELAY_CONFIG [RX_EX_DELAY_PERIOD].

The RX_EX_DELAY_PERIOD should be configured depending on the link rate:

- For link rates 6.144 Gbps and 4.914Gbps, RX_EX_DELAY_PERIOD = 40
- For link rates 3.072 Gbps and 2.457Gbps, RX_EX_DELAY_PERIOD = 80
- For link rates 1.228 Gbps, RX_EX_DELAY_PERIOD = 160

Note: See **Section 18.4.1.11**, *CPRI Extended Delay Measurement Configuration (CPRIn_EX_DELAY_CONFIG)*, **Section 18.4.1.12**, *CPRI Extended Delay Measurement Status (CPRIn_EX_DELAY_STATUS)*

Figure 18-34 shows the CPRI receive delay path.

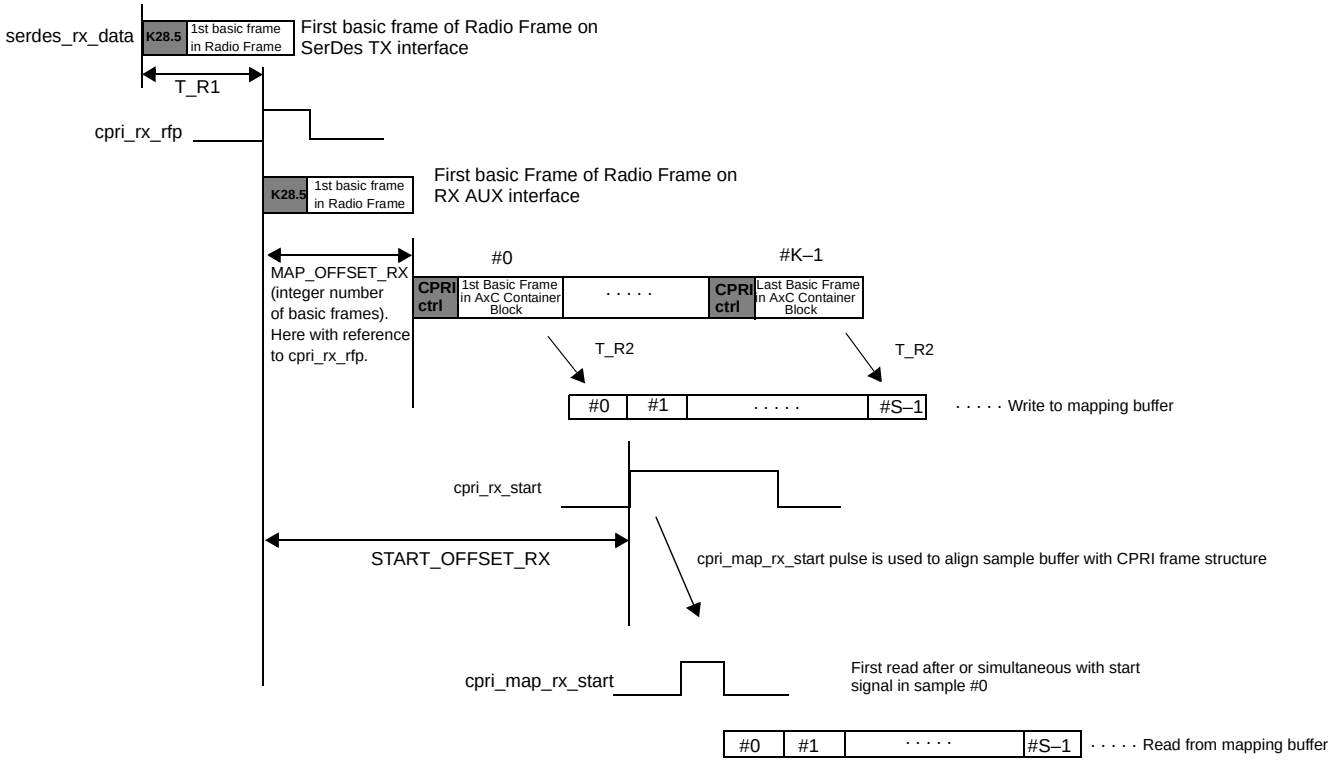


Figure 18-34. CPRI Receive Delay Path

18.3.12.3 Delay Accuracy

The CPRI units fulfill requirement R-19 of the CPRI v4.1 standard defining that the link delay accuracy in downlink between SAPs master port and SAPs slave port excluding the cable length must be ± 8.138 ns.

Requirement R-20 of the CPRI v4.1 standard defines the round trip absolute accuracy excluding cable length to be ± 16.276 ns. In order to comply with this specification, REC chaining using only MSC8157/8 devices can done with up to 6 devices using daisy-chain mode.

18.4 CPRI Programming Model

All the CPRI complex registers are 32-bit registers. The bits are ordered 31 (most-significant bit) to 0 (least-significant bit). All the read and write accesses are executed through a 32-bit bus interface; reads and writes always access all the bits of the register. The registers are categorized as follows:

- *CPRI framer registers.* Configures the data framing operations, including Ethernet and HDLC frame handling. Each of the 6 units has its own registers.
- *CPRI complex registers.* Used to configure the CPRI complex. Each of the 6 units have their own registers
- *Control registers*. Each of the 6 units have their own registers
- *Status registers*. Read-only registers that can be accessed any time. Each of the 6 units have their own registers
- *Configuration memories.* Configures the mapping tables. Each of the 6 units have their own memories.
- *CPRI General Registers.* Registers used by all the 6 units.

The base address of the CPRIn (n=1...6) registers is 0xfff40000 + (n–1)*0x3000.

The address of the registers can be computed by adding the offset from the table to the relevant base address. The General registers use the base address of CPRI1.

For example, the address of CPRIn_HFN register in CPRI3 can be computed by:

0xfff40000 + (3–1)*0x3000 + 0x24 = 0xfff46024.

Table 18-12. CPRI Complex Memory Map

Offset	Register Name	Register Acronym	Description
CPRI Framer Registers			
0x4	CPRI Status Register	CPRIn_STATUS	page 18-66
0x8	CPRI Configuration	CPRIn_CONFIG	page 18-67
0x1C	CPRI Receive Line Coding Violation Counter	CPRIn_LCV	page 18-68
0x20	CPRI Recovered BFN Counter	CPRIn_BFN	page 18-69
0x24	CPRI Recovered HFN Counter	CPRIn_HFN	page 18-69

Table 18-12. CPRI Complex Memory Map (Continued)

Offset	Register Name	Register Acronym	Description
0x2C	CPRI Hardware Reset from Control Word	CPRIn_HW_RESET	page 18-70
0x3C	CPRI Control and Management Configuration	CPRIn_CM_CONFIG	page 18-71
0x40	CPRI Control and Management Status	CPRIn_CM_STATUS	page 18-72
0x48	CPRI Receive Delay	CPRIn_RX_DELAY	page 18-73
0x4C	CPRI Round Trip Delay	CPRIn_ROUND_DELAY	page 18-73
0x50	CPRI Extended Delay Measurement Configuration	CPRIn_EX_DELAY_CONFIG	page 18-74
0x54	CPRI Extended Delay Measurement Status	CPRIn_EX_DELAY_STATUS	page 18-75
0x58	CPRI Transmit Protocol Version	CPRIn_TX_PROT_VER	page 18-75
0x5C	CPRI Transmit Scrambler Seed	CPRIn_TX_SCR_SEED	page 18-76
0x60	CPRI Receive Scrambler Seed	CPRIn_RX_SCR_SEED	page 18-77
0x80	CPRI SerDes Interface Configuration	CPRIn_SERDES_CONFIG	page 18-77
0xC0	CPRI Mapping Configuration	CPRIn_MAP_CONFIG	page 18-78
0xC4	CPRI Mapping Counter Configuration	CPRIn_MAP_CNT_CONFIG	page 18-79
0xD0	CPRI Mapping Table Configuration	CPRIn_MAP_TBL_CONFIG	page 18-79
0xE4	CPRI Mapping RX AxC Container Block Offset	CPRIn_MAP_OFFSET_RX	page 18-80
0xE8	CPRI Mapping TX AxC Container Block Offset	CPRIn_MAP_OFFSET_TX	page 18-81
0xF0	Offset for CPRIn_TX_START Synchronization Output	CPRIn_START_OFFSET_TX	page 18-81
0x100, 0x104	CPRI Mapping Buffer RX Status Register	CPRIn_IQ_RX_BUF_STATUS	page 18-82
0x120, 0x124	CPRI Mapping Buffer TX Status Register	CPRIn_IQ_TX_BUF_STATUS	page 18-83
0x200	Ethernet Receive Status	CPRIn_ETH_RX_STATUS	page 18-84
0x208	Ethernet Feature Enable/Disable and Interrupt Enable Bits	CPRIn_ETH_CONFIG_1	page 18-85
0x20C	Ethernet Miscellaneous Configuration	CPRIn_ETH_CONFIG_2	page 18-86
0x210	Ethernet RX Packet Discard	CPRIn_ETH_RX_CONTROL	page 18-87
0x228	Ethernet RX Additional Status	CPRIn_ETH_RX_EX_STATUS	page 18-87
0x22C	Ethernet MSB of MAC Address (16 bits)	CPRIn_ETH_ADDR_MSB	page 18-88
0x230	Ethernet LSB of MAC Address (32 bits)	CPRIn_ETH_ADDR_LSB	page 18-89
0x234	Ethernet Small 32 Entries Hash Table to Filter Multicast Traffic	CPRIn_ETH_HASH_TABLE	page 18-89
0x244	Ethernet Configuration 3	CPRIn_ETH_CONFIG_3	page 18-90
0x248	Ethernet Receive Frame Counter	CPRIn_ETH_CNT_RX_FRAME	page 18-91
0x300	HDLC Receive Status	CPRIn_HDLC_RX_STATUS	page 18-91
0x308	HDLC Feature Enable/Disable and Interrupt Enable Bits	CPRIn_HDLC_CONFIG_1	page 18-92
0x30C	HDLC Miscellaneous Configuration	CPRIn_HDLC_CONFIG_2	page 18-93
0x310	HDLC RX Packet Discard	CPRIn_HDLC_RX_CONTROL	page 18-94
0x328	HDLC RX External Status	CPRIn_HDLC_RX_EX_STATUS	page 18-94
0x344	HDLC Configuration 3	CPRIn_HDLC_CONFIG_3	page 18-95
0x348	HDLC Receive Frame Counter	CPRIn_HDLC_CNT_RX_FRAME	page 18-96
CPRI Complex Registers			
0x400	Receive IQ MBus Transaction Size	CPRInRIQMTS	page 18-97
0x404	Receive IQ Second Destination Mbus Transaction Size	CPRInRIQSDMTS	page 18-98
0x408	Transmit IQ MBus Transaction Size	CPRInTIQMTS	page 18-99
0x40c	Receive VSS MBus Transaction Size	CPRInRVSSMTS	page 18-100
0x410	Transmit VSS MBus Transaction Size	CPRInTVSSMTS	page 18-100
0x45c	Receive IQ Second Dest Base Address	CPRInRIQSDBA	page 18-101
0x460	Receive IQ Buffer Size	CPRInRIQBS	page 18-102

Table 18-12. CPRI Complex Memory Map (Continued)

Offset	Register Name	Register Acronym	Description
0x464	Receive IQ Second Destination Buffer Size	CPRInRIQSDBS	page 18-102
0x468	Transmit IQ Buffer Size	CPRInTIQBS	page 18-103
0x46c	Receive VSS Buffer Size	CPRInRVSSBS	page 18-103
0x470	Transmit VSS Buffer Size	CPRInTVSSBS	page 18-104
0x474	Receive ETH Buffer Size	CPRInRETHBS	page 18-104
0x478	Receive HDLC Buffer Size	CPRInRHDLCBS	page 18-105
0x47c	Receive VSS Buffer Base Address	CPRInRVSSBA	page 18-105
0x480	Transmit VSS Buffer Base Address	CPRInTVSSBA	page 18-106
0x484	Receive Ethernet BD Ring Base Address	CPRInREBDRBA	page 18-106
0x488	Transmit Ethernet BD Ring Base Address	CPRInTEBDRBA	page 18-107
0x48c	Receive HDLC BD Ring Base Address	CPRInRHBDRBA	page 18-107
0x490	Transmit HDLC BD Ring Base Address	CPRInTHBDRBA	page 18-108
0x494	Receive Ethernet Buffer Descriptor Ring Size	CPRInREBDRS	page 18-108
0x498	Transmit Ethernet Buffer Descriptor Ring Size	CPRInTEBDRS	page 18-109
0x49c	Receive HDLC Buffer Descriptor Ring Size	CPRInRHBDRS	page 18-109
0x4a0	Transmit HDLC Buffer Descriptor Ring Size	CPRInTHBDRS	page 18-110
0x4A8	Receive General CPRI Mode	CPRInRGCM	page 18-110
0x4Ac	Transmit General CPRI Mode	CPRInTGCM	page 18-111
0x4B4	Transmit Synchronization Configuration Register	CPRInTSCR	page 18-112
0x4B8	Transmit CPRI Framer Buffer Size	CPRInTCFBS	page 18-113
0x4BC	Tx Control Table Insert Enable 1	CPRInTCTIE1	page 18-114
0x4C0	Tx Control Table Insert Enable 2	CPRInTCTIE2	page 18-115
0x4C8	Timer Configuration	CPRInTMRC	page 18-116
0x4CC	Receive Frame Pulse Width	CPRInRFPW	page 18-118
0x4D0	Transmit Frame Pulse Width	CPRInTFPW	page 18-119
Control Registers			
0x500	Receive Control Register	CPRInRCR	page 18-120
0x504	Transmit Control Register	CPRInTCR	page 18-121
0x508	Receive AxC Control Register	CPRInRACCR	page 18-122
0x510	Transmit AxC Control Register	CPRInTACCR	page 18-124
0x514	Receive Control Attribute	CPRInRCA	page 18-125
0x518	Receive Control Data0	CPRInRCD0	page 18-126
0x51C	Receive Control Data1	CPRInRCD1	page 18-126
0x520	Receive Control Data2	CPRInRCD2	page 18-127
0x524	Transmit Control Attribute	CPRInTCA	page 18-128
0x528	Transmit Control Data0	CPRInTCD0	page 18-129
0x52C	Transmit Control Data1	CPRInTCD1	page 18-130
0x530	Transmit Control Data2	CPRInTCD2	page 18-130
0x534	Receive IQ First Threshold	CPRInRIQFT	page 18-131
0x538	Receive IQ Second Threshold	CPRInRIQST	page 18-132
0x53C	Receive IQ Threshold	CPRInRIQT	page 18-133
0x540	Transmit IQ First Threshold	CPRInTIQFT	page 18-134
0x544	Transmit IQ Second Threshold	CPRInTIQST	page 18-135
0x548	Transmit IQ Threshold	CPRInTIQT	page 18-136
0x54C	Receive VSS Threshold	CPRInRVSST	page 18-136
0x550	Transmit VSS Threshold	CPRInTVSST	page 18-137
0x554	Receive Ethernet Coalescing Threshold	CPRInRETHCT	page 18-138
0x558	Transmit Ethernet Coalescing Threshold	CPRInTETHCT	page 18-139

Table 18-12. CPRI Complex Memory Map (Continued)

Offset	Register Name	Register Acronym	Description
0x560	CPRI Receive Control and Timing interrupt Enable Register	CPRInRCIER	page 18-140
0x564	CPRI transmit Control and Timing Interrupt Enable Register	CPRInTCIER	page 18-141
0x568	Receive IQ Threshold Second Destination	CPRInRIQTSD	page 18-133
0x570	CPRI Error Interrupt Enable Register	CPRInEIER	page 18-142
0x574	Timer Enable Register	CPRInTMRE	page 18-144
0x578	Receive Ethernet Write Pointer Ring	CPRInREWPR	page 18-145
0x57C	Transmit Ethernet Write Pointer Ring	CPRInTEWPR	page 18-146
0x580	Receive HDLC Write Pointer Ring	CPRInRHWPR	page 18-147
0x584	Transmit HDLC Write Pointer Ring	CPRInTHWPR	page 18-148
0x5B0–0x60F	Receive AxCy Parameter Register	CPRInRACPRy	page 18-149
0x640– 0x69F	Transmit AxCy Parameter Register	CPRInTACPRy	page 18-150
0x700-0x79f	CPRI Auxiliary Interface Mask Registers	CPRInMASKRy	page 18-151
0x7a0	CPRI Auxiliary Control Register	CPRInAUXCR	page 18-152
Status Registers			
0x800	Receive IQ Buffer Displacement Register	CPRInRIQBDR	page 18-153
0x804	Receive IQ Buffer Second Destination Displacement Register	CPRInRIQSDBDR	page 18-154
0x808	Transmit IQ Buffer Displacement Register	CPRInTIQBDR	page 18-154
0x80c	Receive Chips Counter Register	CPRInRCCR	page 18-155
0x810	Receive VSS Buffer Displacement Register	CPRInRVSSBDR	page 18-156
0x814	Transmit VSS Buffer Displacement Register	CPRInTVSSBDR	page 18-157
0x818–0x81f	Receive Ethernet Buffer Descriptor	CPRInRETHBD	page 18-158
0x820-827	Transmit Ethernet Buffer Descriptor	CPRInTETHBD	page 18-159
0x828	Receive Ethernet Read Pointer Ring	CPRInRERPR	page 18-160
0x82c	Transmit Ethernet Read Pointer Ring	CPRInTERPR	page 18-161
0x830–0x837	Receive HDLC Buffer Descriptor	CPRInRHDLCBD	page 18-162
0x838–0x83f	Transmit HDLC Buffer Descriptor	CPRInTHDLCBD	page 18-163
0x840	Receive HDLC Read Pointer Ring	CPRInRHRPR	page 18-164
0x844	transmit HDLC Read Pointer Ring	CPRInTHRPR	page 18-165
0x848	Receive Event Register	CPRInRER	page 18-166
0x84c	Transmit Event Register	CPRInTER	page 18-167
0x850	Error Event Register	CPRInEER	page 18-169
0x854	Receive Ethernet Coalescing Status	CPRInRETHCS	page 18-171
0x858	Transmit Ethernet Coalescing Status	CPRInTETHCS	page 18-172
0x85c	Timer Status Register.	CPRInTMRSR	page 18-172
0x860	Receive Status Register	CPRInRSR	page 18-173
0x864	Transmit Status Register	CPRInTSR	page 18-174
Configuration Memories			
0x900–0x1800	Receive Configuration Memory	RCM	page 18-174
0x1800–0x2700	Transmit Configuration Memory	TCM	page 18-176

Offset	Register Name	Register Acronym	Description
CPRI General Register			
0x18000	CPRI Clocks Control Register	CPRICCR	page 18-178

Offset	Register Name	Register Acronym	Description
0x18010–0x18050	CPRI Interrupt Control Register y	CPRIICRy	page 18-179
0x18050	CPRI Receive CPU Control Interrupt Enable Register	CPRIRCCIER	page 18-181
0x18054	CPRI Transmit CPU Control Interrupt Enable Register	CPRITCCIER	page 18-183
0x18058	General Receive Synchronization Register	CPRIGRSR	page 18-185
0x1805c	General TRansmit Synchronization Register	CPRIGTSR	page 18-186
0x18094	CPRI Error Status Register	CPRIESR	page 18-187

18.4.1 CPRI Framer Registers

The following sections describe the CPRI Framing registers.

18.4.1.1 CPRI Status Register (CPRIn_STATUS)

address 0x4 (CPRIn_STATUS)

	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	—				RX_RFP_HOLD	RX_FREQ_ALARM_HOLD	RX_STATE_HOLD	RX_LOS_HOLD	—				RX_BFN_STATE	RX_HFN_STATE	RX_STATE	RX_LOS
W																
Reset:	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1

Figure 18-35. CPRIn_STATUS

Table 18-13. CPRIn_STATUS Bit Descriptions

Name	Reset	Description	Setting
— 31–12	0	Reserved. Write to zero for future compatibility.	
RX_RFP_HOLD 11	0	**Radio Frame Pulse (RFP) Received** Indicates that an RFP was received. This bit is set every 10 ms when hyperframe, basic frame =0,0. This bit is read-to-clear.	0 RFP not detected 1 RFP detected
RX_FREQ_ALARM_HOLD 10	1	**Receive Clock Frequency Alarm Received** This bit indicates that a CPRI CDR Clock is not synchronous with the Framer Clock. The alarm is set every time the phase of the CDR Clock has changed more than 4 clock periods of the Framer Clock. This bit is read-to-clear.After clearing it takes a new drift of 4 cycles to set the alarm again	0 No frequency alarm occurred 1 Frequency alarm occurred

Table 18-13. CPRIn_STATUS Bit Descriptions

Name	Reset	Description	Setting
RX_STATE_HOLD 9	1	**Receive State Hold** Indicates that the internal RX state machine is out of synchronization. It happens when either K.28.5 was at least once not in its expected position or an error occurred in the HFN,BFN counter sequence (If for 4 hyperframes the counter values recovered from the incoming CPRI frame structure do not match the expected values, a loss of SYNC is declared). This bit is read-to-clear.	0 No receive loss of SYNC (LOS). 1 Receive LOS occurred
RX_LOS_HOLD 8	1	**Receive Loss of Signal (LOS) Occurred** Indicates that a loss of SYNC occurred. This bit is a sticky read-to-clear bit.	0 No LOS detected 1 LOS detected
— 7–4	0	Reserved. Write to zero for future compatibility.	
RX_BFN_STATE 3	0	**Receive NodeB Frame (BFN) Alignment Detected** Indicates whether BFN alignment was detected.	0 No BFN alignment detected 1 BFN alignment detected
RX_HFN_STATE 2	0	**Receive HFN Alignment Detected** Indicates whether HFN alignment was detected.	0 No HFN alignment detected 1 HFN alignment detected
RX_STATE 1	0	**Receive State** Indicates the receive state. The state is sync if the receive state is ok and HFN and BFN states are ok (bIts RX_BFN_STATE and RX_HFN_STATE are set)	0 Global Receive State is not sync. 1 Global Receive State is sync.
RX_LOS 0	1	**Receive Loss of Signal (LOS)** Indicates either excessive 8b/10b violations (>15) or external indication of Loss of Signal from optical modules.	0 LOS not detected. 1 LOS detected.

18.4.1.2 CPRI Configuration (CPRIn_CONFIG)

address 0x8 (CPRIn_CONFIG)

	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	—						RX_ENABLE	—			TX_ENABLE	—			SYNC_MODE	TX_CTRL_INSERT_EN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-36. CPRIn_CONFIG

Table 18-14. CPRIn_CONFIG Bit Descriptions

Name	Reset	Description	Setting
— 31–9	0	Reserved. Write to zero for future compatibility.	
RX_ENABLE 9	0	**Receive Enable** Enables or disables receive operations.	0 Disable receive operations 1 Enable receive operations.
— 8–6	0	Reserved. Write to zero for future compatibility.	

Table 18-14. CPRIn_CONFIG Bit Descriptions (Continued)

Name	Reset	Description	Setting
TX_ENABLE 5	0	**Transmit Enable** Enables or disables transmit operations.	0 Disable transmit operations. 1 Enable transmit operations.
— **4–**2	0	Reserved. Write to zero for future compatibility.	
SYNC_ MODE 1	0	End Point Slave Mode When this bit is cleared, the lane operates in Master Mode. As a master, it can generate a reset request to the Slave and receive a reset acknowledge. It uses the 10 ms pulse generated by the Timer for transmission. When the bit is set, the lane operates in Slave Mode, in which it can receive a reset request from the Master and generate a reset acknowledge. It uses the recovered 10 ms pulse for transmission.	0 Master Mode 1 Slave Mode
TX_CTRL_ INSERT_EN 0	0	**CPRI Control Word Insertion l Enable** Transmit Control Word insertion enable.	0 Transmit control word insertion disabled. 1 Transmit control word insertion enabled,.

18.4.1.3 CPRI Receive Line Coding Violation Counter (CPRIn_LCV)

address 0x1C (CPRIn_LCV)

	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	—								CNT_LCV							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-37. CPRIn_LCV

Table 18-15. CPRIn_LCV Bit Descriptions

Name	Reset	Description	Setting
— 31–8	0	Reserved. Write to zero for future compatibility.	
CNT_LCV 7–0		**Line Code Violation Count** Stores the recorded number of line code violation (LCV) indications from the 8b/10b decoding block. The counter saturates when it reaches the maximum value. This field is read-to-clear.	

18.4.1.4 CPRI Recovered BFN Counter (CPRIn_BFN)

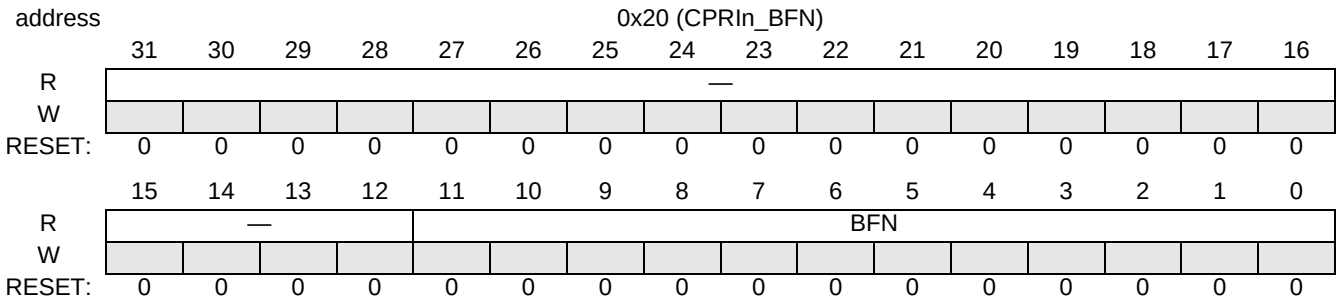


Figure 18-38. CPRIn_BFN

Table 18-16. CPRIn_BFN Bit Descriptions

Name	Reset	Description	Setting
— 31–12	0	Reserved. Write to zero for future compatibility.	
BFN 11–0		**Current Mode B Radio Frame Number (BFN)** Stores the current BFN number aligned obtained from the BFN alignment state machine.	

18.4.1.5 CPRI Recovered HFN Counter (CPRIn_HFN)

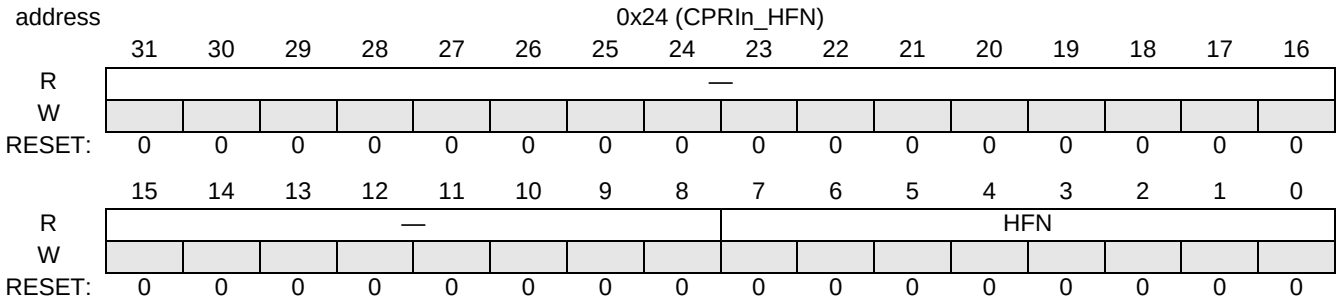


Figure 18-39. CPRIn_HFN

Table 18-17. CPRIn_HFN Bit Descriptions

Name	Reset	Description	Setting
— 31–8	0	Reserved. Write to zero for future compatibility.	
HFN 7–0		**Current Hyper Frame Number (HFN)** Stores the current HFN number aligned obtained from the HFN alignment state machine.	

18.4.1.6 CPRI Hardware Reset from Control Word (CPRIn_HW_RESET)

address 0x2C (CPRIn_HW_RESET)

	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	—								RESET_GEN_DONE_HOLD	RESET_GEN_DONE	RESET_DETECT_HOLD	RESET_DETECT	—	RESET_OUT_EN	RESET_GEN_FORCE	RESET_GEN_EN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-40. CPRIn_HW_RESET

This register is used to control and monitor reset request or acknowledges mapped to bit 0 of Z.130.0 CPRI control word. The register is used to transmit reset request to RE nodes.

A reset request can be sent by setting RESET_GEN_EN and RESET_GEN_FORCE signals. The reset generation request will be hold as long as RESET_GEN_FORCE is asserted and until reset acknowledge have been received from the RE. However reset generation request can always be aborted by clearing the RESET_GEN_EN signal.

Table 18-18. CPRIn_HW_RESET Bit Descriptions

Name	Reset	Description	Setting
— 31–8	0	Reserved. Write to zero for future compatibility.	
RESET_GEN_DONE_HOLD 7	0	**Hold Reset Done** This bit holds the Reset Generation Done value. **Note:** This bit is sticky read-to-clear.	0 Reset generation not done. 1 Reset generation done.
RESET_GEN_DONE 6	0	**Reset Generation Done** Indicates whether the reset request or acknowledge is done. Set when CPRI completes sending 10 consecutive reset request or acknowledge indications	0 Reset request/ acknowledge is not done. 1 Reset request/ acknowledge is done.
RESET_DETECT_HOLD 5	0	**Hold Reset Detected** This bit holds the Reset Detected value. **Note:** This bit is sticky read-to-clear.	0 Reset request not detected. 1 Reset request detected.
RESET_DETECT 4	0	**Reset Detected** Indicates that a reset request/acknowledge was detected on the CPRI RX side.	0 Reset request/acknowledge not detected. 1 Reset request/acknowledge detected.
— 3	0	Reserved. Write to zero for future compatibility.	
RESET_OUT_EN 2	0	**Reset Indication Output Enable** Used to enable the reset output request.	0 Disable reset indication output 1 Enable reset indication output

Table 18-18. CPRIn_HW_RESET Bit Descriptions (Continued)

Name	Reset	Description	Setting
RESET_GEN_FORCE 1	0	**Forces Reset Request/Acknowledge** Setting this bit forces a reset request/acknowledge on the CPRI TX side.	0 Disable reset request/acknowledge. 1 Force reset request/acknowledge.
RESET_GEN_EN 0	0	**Reset Request/Acknowledge Generation Enable** Setting this bit enables generation of a reset request/acknowledge on the CPRI TX side via the dedicated control word (Z.130.0).	0 Generation of reset request/acknowledge disabled. 1 Generation of reset request/acknowledge enabled.

18.4.1.7 CPRI Control and Management Configuration (CPRIn_CM_CONFIG)

address 0x3C (CPRIn_CM_CONFIG)

	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	—					TX_SLOW_CM_RATE			—		TX_FAST_CM_PTR					
W																
RESET:	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0

Figure 18-41. CPRIn_CM_CONFIG

Table 18-19. CPRIn_CM_CONFIG Bit Descriptions

Name	Reset	Description	Setting
— 31–11	0	Reserved. Write to zero for future compatibility.	
TX_SLOW_CM_RATE 10–8	0	**Slow Transmit Control and Management Rate** Specifies the slow transmit rate.	000 No HDLC. 001 240 Kbps 010 480 Kbps 011 960 Kbps 100 1920 Kbps 101 2400 Kbps 110 Highest possible rate when the line rate is > 3072 Mbps. 111 reserved.
— 7–6	0	Reserved. Write to zero for future compatibility.	
TX_FAST_CM_PTR 5–0	0x24	**Pointer to First CPRI Control Word** Contains the pointer to the first CPRI control word used for fast Control and Management. This value is inserted into the CPRI control byte Z.194.0.	Valid values: 0x14–0x35

18.4.1.8 CPRI Control and Management Status (CPRIn_CM_STATUS)

address 0x40 (CPRIn_CM_STATUS)

	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	—				RX_SLOW_CM_RATE_VALID	RX_SLOW_CM_RATE			—	RX_FAST_CM_PTR_VALID	RX_FAST_CM_PTR					
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-42. CPRIn_CM_STATUS

Table 18-20. CPRIn_CM_STATUS Bit Descriptions

Name	Reset	Description	Setting
— 31–12	0	Reserved. Write to zero for future compatibility.	
RX_SLOW_CM_RATE_VALID 11		**Receive Slow Control and Management Rate Valid** Indicates whether a valid slow Control and Management rate is accepted.	0 Slow rate not accepted. 1 Slow rate accepted.
RX_SLOW_CM_RATE 10–8		**Receive Slow Control and Management Rate** Contains the value of the accepted slow Control and Management rate.	000 No HDLC. 001 240 Kbps 010 480 Kbps 011 960 Kbps 100 1920 Kbps 101 2400 Kbps 110 Highest possible rate when the line rate is > 3072 Mbps. 111 reserved.
7	0	Reserved. Write to zero for future compatibility.	
RX_FAST_CM_PTR_VALID 6		**Receive Fast Control and Management Pointer Valid** Indicates whether a valid fast Control and Management pointer is accepted.	0 Pointer not accepted. 1 Pointer accepted.
RX_FAST_CM_PTR 5–0		**Receive Fast Control and Management Pointer** Contains the value of the accepted fast Control and Management pointer.	

18.4.1.9 CPRI Receive Delay (CPRIn_RX_DELAY)

address 0x48 (CPRIn_RX_DELAY)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—									RX_ALIGN_DELAY						
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	—						RX_BUF_DELAY								RX_BYTE_DELAY	
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-43. CPRIn_RX_DELAY

Table 18-21. CPRIn_RX_DELAY Bit Descriptions

Name	Reset	Description	Setting
— 31–23	0	Reserved. Write to zero for future compatibility.	
RX_ALIGN_DELAY 22–16	0	**Bit Delay in 8b/10b Comma Alignment** Resolution is in bit periods before 8b/10b decoding. Maximum RX_BUF_DELAY value is 39.	
— 15–10	0	Reserved. Write to zero for future compatibility.	
RX_BUF_DELAY 9–2	0	**Current Receive Buffer Delay** Stores the current receive buffer delay. Resolution is 32-bit words (or 40-bit before 8b/10b decoding).	
RX_BYTE_DELAY 1–0	0	**RX Byte Alignment Delay** Resolution is 8-bit words (or 10-bit before 8b/10b decoding).	

18.4.1.10 CPRI Round Trip Delay (CPRIn_ROUND_DELAY)

address 0x4C (CPRIn_ROUND_DELAY)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—												RX_ROUND_TRIP_DELAY			
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	RX_ROUND_TRIP_DELAY															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-44. CPRIn_ROUND_DELAY

Table 18-22. CPRIn_ROUND_DELAY Bit Descriptions

Name	Reset	Description	Setting
— 31–20	0	Reserved. Write to zero for future compatibility.	

Table 18-22. CPRIn_ROUND_DELAY Bit Descriptions

Name	Reset	Description	Setting
RX_ROUND_TRIP_DELAY 19–0	0	**CPRI Round Trip Delay** Measures the round trip delay in clock periods of the Framer Clock between CPRIn_TX_RFP and CPRIn_RX_RFP.	

18.4.1.11 CPRI Extended Delay Measurement Configuration (CPRIn_EX_DELAY_CONFIG)

address 0x50 (CPRIn_EX_DELAY_CONFIG)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—										TX_EX_DELAY					
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	—							RX_EX_DELAY_PERIOD								
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-45. CPRIn_EX_DELAY_CONFIG

Table 18-23. CPRIn_EX_DELAY_CONFIG Bit Descriptions

Name	Reset	Description	Setting
— 31–22	0	Reserved. Write to zero for future compatibility.	
TX_EX_DELAY 21–16		**Additional Transmit Delay** Additional transmit delay in units of bit-periods. Valid range is 0 to 39.	
— 15–9	0	Reserved. Write to zero for future compatibility.	
RX_EX_DELAY_PERIOD 8–0		**Integrated Period for Extended Delay Measurement** Note that the value 0 gives an integration period of 1, that is, the integration period will be rx_ex_delay_period + 1. For MSC8157E, the value is according the line rate: - For line rates 6.144 Gbps and 4.914 Gbps:40 - For line rates 3.072 Gbps and 2.457 Gbps:80 - For line rate 1.228 Gbps:160	

18.4.1.12 CPRI Extended Delay Measurement Status (CPRIn_EX_DELAY_STATUS)

address 0x54 (CPRIn_EX_DELAY_STATUS)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—															RX_EX_BUF_DELAY_VALID
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	—	RX_EX_BUF_DELAY														
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-46. CPRIn_EX_DELAY_STATUS

Table 18-24. CPRIn_EX_DELAY_STATUS Bit Descriptions

Name	Reset	Description	Setting
— 31–17	0	Reserved. Write to zero for future compatibility.	
RX_EX_BUF_DELAY_VALID 16	0	**Receive Extended Delay Measurement Updated** Indicates whether the receive extended delay measurement is updated by a new measurement, **Note:** This bit is read-to-clear.	0 Measurement not valid. 1 Measurement valid.
— 15	0	Reserved. Write to zero for future compatibility.	
RX_EX_BUF_DELAY 14–0	0	**Receive Extended Delay Buffer Measurement** Stores the extended delay buffer measurement value.	

18.4.1.13 CPRI Transmit Protocol Version (CPRIn_TX_PROT_VER)

address 0x58 (CPRIn_TX_PROT_VER)

	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	—								PROT_VER							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-47. CPRIn_TX_PROT_VER

It is possible to scramble the CPRI frames for rates higher than 3.072 Gbps for Protocol Version 2. The scrambling is enabled by configuring Protocol Version 2 in the CPRIn_TX_PROT_VER register for non-daisy-chain topologies. The scrambler seed should be set in the CPRIn_TX_SRC_SEED register.

Table 18-25. CPRIn_TX_PROT_VER Bit Descriptions

Name	Reset	Description	Setting
— 31–8	0	Reserved. Write to zero for future compatibility.	
PROT_VER 7–0	0	**Transmitted Protocol Version** Transmit protocol version mapped into Z.2.0.	001 transmit protocol version 1 010 transmit protocol version 2

18.4.1.14 CPRI Transmit Scrambler Seed (CPRIn_TX_SCR_SEED)

address 0x5C (CPRIn_TX_SCR_SEED)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W		TX_SCR_SEED														
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	TX_SCR_SEED															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-48. CPRIn_TX_SCR_SEED

Table 18-26. CPRIn_TX_SCR_SEED Bit Descriptions

Name	Reset	Description	Setting
— 31	0	Reserved. Write to zero for future compatibility.	
TX_SCR_SEED 30–0	0	**Transmit Scrambling Seed** Seed for Tx scrambler. Value 0 means that the scrambler is disabled. Can be used only for line rates higher than 3.072 Gbps.	

18.4.1.15 CPRI Receive Scrambler Seed (CPRIn_RX_SCR_SEED)

address 0x60 (CPRIn_RX_SCR_SEED)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RX_SCR_EN	RX_SCR_SEED														
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	RX_SCR_SEED															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-49. CPRIn_RX_SCR_SEED

Table 18-27. CPRIn_RX_SCR_SEED Bit Descriptions

Name	Reset	Description	Setting
RX_SCR_EN 31	0	**Receive Scrambling Seed Enable** Detected from the received protocol version	0 Detected Scrambling disabled 1 Detected Scrambling enabled.
RX_SCR_SEED 30–0	0	**Receive Scrambling Seed** Seed for Rx scrambler. Detected from incoming data.	

18.4.1.16 CPRI SerDes Interface Configuration (CPRIn_SERDES_CONFIG)

address 0x80 (CPRIn_SERDES_CONFIG)

	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	—														SWAP_SERDES_RX	SWAP_SERDES_TX
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-50. CPRIn_SERDES_CONFIG

Table 18-28. CPRIn_SERDES_CONFIG Bit Descriptions

Name	Reset	Description	Setting
— 31–2	0	Reserved. Write to zero for future compatibility.	
SWAP_SERDES_RX 1		**Swap SerDes RX Bits** Indicates whether to swap receive bit order. **Note:** This is only used when INCLUDE_8B10B is TRUE.	0 Swap disabled. 1 Swap enabled. Swap receive bit order on the SerDes interface.
SWAP_SERDES_TX 0		**Swap SerDes TX Bits** Indicates whether to swap transmit bit order. **Note:** This is only used when INCLUDE_8B10B is TRUE.	0 Swap disabled. 1 Swap enabled. Swap transmit bit order on the SerDes interface.

18.4.1.17 CPRI Mapping Configuration (CPRIn_MAP_CONFIG)

address 0xC0 (CPRIn_MAP_CONFIG)

	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	—								CPRIn_MAP_TX_TR_MODE	CPRIn_MAP_RX_TR_MODE	—	MAP_15BIT_MODE	—		MAP_MODE	
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-51. CPRIn_MAP_CONFIG

Table 18-29. CPRIn_MAP_CONFIG Bit Descriptions

Name	Reset	Description	Setting
— 31–8	0	Reserved. Write to zero for future compatibility.	
CPRIn_MAP_TX_TR_MODE 7		**Transmit Transparent Mode Select** Used to select transmit transparent mode.	0 Transmit transparent mode disabled. 1 Transmit transparent mode enabled.
CPRIn_MAP_RX_TR_MODE 6		**Receive Transparent Mode Select** Used to select receive transparent mode.	0 Receive transparent mode disabled. 1 Receive transparent mode enabled.
5		Reserved. Write to zero for future compatibility.	
MAP_15BIT_MODE 4		**15-bit Mode Select** Used to select 15-bit or 16-bit sample mode. When this bit is set TSW field in register CPRInTGCM and RSW field in register CPRInRGCM should be set to 0x1	0 2 × 16-bit I/Q sample width 1 2 × 15-bit I/Q sample width
— 3–2		Reserved. Write to zero for future compatibility.	
MAP_MODE 1–0	0	**Mapping Mode** Selects the mapping mode.	00 Basic mapping mode. 01 Advanced mapping method 1. 10 Advanced mapping method 3 11 reserved

18.4.1.18 CPRI Mapping Counter Configuration (CPRIn_MAP_CNT_CONFIG)

address 0xC4 (CPRIn_MAP_CNT_CONFIG)

	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	—			MAP_AC					—			MAP_N_AC				
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-52. CPRIn_MAP_CNT_CONFIG

This register is used when MAP_MODE = 00 to configure the number of active AxCs and the sample rate for each AxC interface. This register configures the basic UMTS/LTE mapping. The register is not used for MAP_MODE = 01 or 10.

Table 18-30. CPRIn_MAP_CNT_CONFIG Bit Descriptions

Name	Reset	Description	Setting
— 31–13	0	Reserved. Write to zero for future compatibility.:1	
MAP_AC 12–8	0	**Number of AxCs Enabled** This field determines the number of enabled AxCs	
— 7–5	0	Reserved. Write to zero for future compatibility.	
MAP_N_AC 4–0	0	**Oversampling Factor** Indicates the oversampling ratio. Sample rate is computed as MAP_N_AC × 3.84 Msps	1: Oversampling ratio 1 2: Oversampling ratio 2 . 8: Oversampling ratio 8

18.4.1.19 CPRI Mapping Table Configuration (CPRIn_MAP_TBL_CONFIG)

address 0xD0 (CPRIn_MAP_TBL_CONFIG)

	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	—									K						
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-53. CPRIn_MAP_TBL_CONFIG

This register is used when MAP_MODE = 01 or 10 to configure the number of basic frames in an AxC Container Group.

Table 18-31. CPRIn_MAP_TBL_CONFIG Bit Descriptions

Name	Reset	Description	Setting
K 6–0	0	**Number of Basic Frames in AxC Container Block** Configures the K parameter for advanced table-based mapping. **Note:** This field is variable and ranges from bit x (determined by the value of WIDTH_K generic value used in the system) to bit 0.	

18.4.1.20 CPRI RX AxC Container Mapping Block Offset (CPRIn_MAP_OFFSET_RX)

address 0xE4 (CPRIn_MAP_OFFSET_RX)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—															MAP_RX_HF_RESYNC
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 W	MAP_RX_OFFSET_Z								MAP_RX_OFFSET_X							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-54. CPRIn_MAP_OFFSET_RX

This register is used to control where the first I/Q sample is mapped on the RX CPRI frame. The RX mapping interface starts to send data to the DMA controller as soon as four I/Q samples are available in the buffer.

Table 18-32. CPRIn_MAP_OFFSET_RX Bit Descriptions

Name	Reset	Description	Setting
— 31–17	0	Reserved. Write to zero for future compatibility.	
MAP_RX_HF_RESYNC 16		**RX Resynchronization Every Hyper Frame Enable** This bit enables synchronization every Hyper Frame instead of every Radio Frame. When asserted the MAP_RX_OFFSET_Z field is ignored.	0 Synchronization every Radio Frame. 1 Synchronization every Hyper Frame.
MAP_RX_OFFSET_Z 15–8		**Hyper Frame Number RX Side** Contains the hyper frame number for the start of the RX side AxC Container Block.	
MAP_RX_OFFSET_X 7–0		**Basic Frame Number RX Side** Contains the basic frame number for the start of the RX side AxC Container Block.	

18.4.1.21 CPRI TX AxC Container Mapping Block Offset (CPRIn_MAP_OFFSET_TX)

address 0xE8 (CPRIn_MAP_OFFSET_TX)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—															MAP_TX_HF_RESYNC
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	MAP_TX_OFFSET_Z								MAP_TX_OFFSET_X							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-55. CPRIn_MAP_OFFSET_TX

This register is used to control where the first I/Q sample is mapped on the TX CPRI frame.

Table 18-33. CPRIn_MAP_OFFSET_TX Bit Descriptions

Name	Reset	Description	Setting
— 31–17	0	Reserved. Write to zero for future compatibility.	
MAP_TX_HF_RESYNC 16		**TX Resynchronization Every Hyper Frame Enable** This bit enables synchronization every Hyper Frame instead of every Radio Frame. When asserted the MAP_TX_OFFSET_Z field is ignored.	0 Synchronization every Radio Frame. 1 Synchronization every Hyper Frame.
MAP_TX_OFFSET_Z 15–8		**Hyper Frame Number TX Side** Contains the hyper frame number for the start of TX side AxC Container Block.	
MAP_TX_OFFSET_X 7–0		**Basic Frame Number TX Side** Contains the basic frame number for the start of TX side AxC Container Block.	

18.4.1.22 CPRI Offset for TX Start Synchronization Output (CPRIn_START_OFFSET_TX)

address 0xF0 (CPRIn_START_OFFSET_TX)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—							START_TX_HF_RESYNC	—							
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	START_TX_OFFSET_Z								START_TX_OFFSET_X							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-56. CPRIn_START_OFFSET_TX

This register is used to control when TX mapping starts to request data. The register must be programmed to a position before the CPRIn_MAP_OFFSET_TX position with a margin to ensure that there is always transmit data available.

Table 18-34. CPRIn_START_OFFSET_TX Bit Descriptions

Name	Reset	Description	Setting
— 31–25	0	Reserved. Write to zero for future compatibility.	
START_TX_HF_RESYNC 24	0	**TX Resynchronization Every Hyper Frame Enable** This bit enables synchronization every Hyper Frame instead of every Radio Frame. When asserted the START_TX_OFFSET_Z field is ignored.	0 Synchronization every Radio Frame. 1 Synchronization every Hyper Frame.
— 23–16	0	Reserved. Write to zero for future compatibility	
START_TX_OFFSET_Z 15–8	0	**TX Start Synchronization Hyper Frame Number** Stores the hyper frame number for the start of the CPRIn_TX_START synchronization output.	
START_TX_OFFSET_X 7–0	0	**TX Start Synchronization Basic Frame Number** Stores the basic frame number for the start of the CPRIn_TX_START synchronization output.	

18.4.1.23 CPRI Mapping Buffer RX Status register <y> (CPRIn_IQ_RX_BUF_STATUS<y>)

address 0x100, 0x104(CPRIn_IQ_RX_BUF_STATUS<y>)

	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	BUFFER_RX_OVERFLOW															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-57. CPRIn_IQ_RX_BUF_STATUS<Y>

This register stores the MAP receiver FIFO buffering status. The register contains the mapping interface status for each of the AxC interfaces. If there are more than 16 AxC interfaces, the status splits into multiple registers, so that the status for AxC numbers 0 to 15 are available at the lowest memory address and status for AxC numbers 16 to 23 at the next memory address (+4).

Table 18-35. CPRIn_IQ_RX_BUF_STATUS Bit Descriptions

Name	Reset	Description	Setting
31–16		Reserved. Write to zero for future compatibility	
BUFFER_RX_OVERFLOW 15–0		**RX Buffer Overflow** Each bit represents the RX buffer overflow status for one AxC interface. Each bit is read-to-clear	0 No buffer overflow. 1 Buffer overflow.

18.4.1.24 CPRI Mapping Buffer TX Status Register<y> (CPRIn_IQ_TX_BUF_STATUS<y>)

address 0x120, 0x124(CPRIn_IQ_TX_BUF_STATUS<y>)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BUFFER_TX_UNDERFLOW															
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	—															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-58. CPRIn_IQ_TX_BUF_STATUS<Y>

This register stores the MAP transmit FIFO buffering status. The register contains the mapping interface status for each of the AxC interfaces. If there are more than 16 AxC interfaces, the status is split into multiple registers, so that the status for AxC numbers 0 to 15 are available at the lowest memory address and status for AxC numbers 16 to 23 at the next memory address (+4).

Table 18-36. CPRIn_IQ_TX_BUF_STATUS Bit Descriptions

Name	Reset	Description	Setting
BUFFER_TX_UNDERFLOW 31–16	0	**TX Buffer Underflow** Each bit represents the RX buffer overflow status for one AxC interface. Each bit is read-to-clear.	0 No buffer underflow. 1 Buffer underflow.
— 15–0	0	Reserved. Write to zero for future compatibility.	

18.4.1.25 Ethernet Receive Status (CPRIn_ETH_RX_STATUS)

address 0x200 (CPRIn_ETH_RX_STATUS)

	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	—			RX_CRC_ERROR	—	RX_MII_ERROR	RX_SMALL_PACKET	RX_OVERFLOW	RX_DMAC_MISMATCH	—				RX_ABORT	—	
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-59. CPRIn_ETH_RX_STATUS

Table 18-37. CPRIn_ETH_RX_STATUS Bit Descriptions

Name	Reset	Description	Setting
— 31–13	0	Reserved. Write to zero for future compatibility.	
RX_CRC_ERROR 12	0	**Ethernet RX CRC Error** Indicates if the Ethernet RX CRC error is detected.	0 Frame not aborted due to CRC error. 1 Frame aborted due to CRC error.
11		Reserved. Write to zero for future compatibility.	
RX_MII_ERROR 10	0	**Ethernet MII Error** Indicates if the Ethernet RX MII protocol error is detected.	0 Frame not aborted due to mii protocol error. 1 Frame aborted due to MII protocol error.
RX_SMALL_PACKET 9	0	**Ethernet RX Small Packet** Indicates if the Ethernet packet is too short.	0 Frame not aborted due too short packet. 1 Frame aborted due too short frame.
RX_OVERFLOW 8	0	**Ethernet RX Overflow** Indicates if the Ethernet RX buffer overflows.	0 Frame not aborted due to RX buffer overflow. 1 Frame aborted due to RX buffer overflow.
RX_DMAC_MISMATCH 7	0	**Ethernet RX DMAC Mismatch** Indicates if the Ethernet RX DMAC mismatch is detected.	0 Frame not aborted due to RX DMAC mismatch. 1 Frame aborted due to RX DMAC mismatch.
6–3	0	Reserved. Write to zero for future compatibility.	
RX_ABORT 2	0	**Ethernet RX Packet Aborted** Indicates if an Ethernet RX packet is aborted.	0 Ethernet RX packet not aborted. 1 Ethernet RX packet aborted
1–0	0	Reserved. Write to zero for future compatibility.	

18.4.1.26 Ethernet Feature Enable/Disable and Trigger Enable Bits (CPRIn_ETH_CONFIG_1)

address 0x208 (CPRIn_ETH_CONFIG_1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—													TRIG_TX_ABORT_EN	TRIG_TX_READY_EN	—
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	—	TRIG_RX_ABORT_EN	TRIG_RX_READY_EN	TRIG_TX_EN	TRIG_RX_EN	TRIG_EN	RX_LONG_FRAME_EN	RX_PREAMBLE_ABORT_EN	BROADCAST_EN	MULTCAST_LT_EN	MAC_CHECK	LENGT_CHECK	—		LITTLE_ENDIAN_E	—
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-60. CPRIn_ETH_CONFIG_1

Bits 18–17 and 14–10 of this register must be set when using Ethernet. This is required for and Ethernet data transfer between the framer and the DMA.

Table 18-38. CPRIn_ETH_CONFIG_1 Bit Descriptions

Name	Reset	Description	Setting
— 31–19	0	Reserved. Write to zero for future compatibility.	
TRIG_TX_ABORT_EN 18	0	**Ethernet TX Abort Trigger Enable** Enables TX abort trigger.	0 Trigger disabled. 1 Trigger enabled.
TRIG_TX_READY_EN 17	0	**Ethernet TX Ready Trigger Enable** Enables TX ready trigger.	0 Trigger disabled. 1 Trigger enabled.
16–15	0	Reserved. Write to zero for future compatibility.	
TRIG_RX_ABORT_EN 14	0	**Ethernet RX Abort Trigger Enable** Enables RX abort trigger.	0 Trigger disabled. 1 Trigger enabled.
TRIG_RX_READY_EN 13	0	**Ethernet RX Ready Trigger Enable** Enables RX ready trigger.	0 Trigger disabled. 1 Trigger enabled.
TRIG_TX_EN 12	0	**Ethernet TX Trigger Enable** Enables TX trigger.	0 Trigger disabled. 1 Trigger enabled.
TRIG_RX_EN 11	0	**Ethernet RX Trigger Enable** Enables RX trigger.	0 Trigger disabled. 1 Trigger enabled.
TRIG_EN 10	0	**Ethernet Global Trigger Enable** Enables global Ethernet Trigger.	0 Trigger disabled. 1 Trigger enabled.
RX_LONG_FRAME_EN 9	0	**RX Long Ethernet Frame Enable** Enables receipt of RX frames >1536 bytes.	0 Long Ethernet frame RX disabled. 1 Long Ethernet frame RX enabled.
RX_PREAMBLE_ABORT_EN 8	0	**RX Illegal Preamble Discard Enable** Enables discard of RX frames with illegal preamble nibble before receipt of SFD.	0 Discard of illegal preamble disabled. 1 Discard of illegal preamble enabled.

Table 18-38. CPRIn_ETH_CONFIG_1 Bit Descriptions (Continued)

Name	Reset	Description	Setting
BROAD CAST_EN 7	0	**RX Broadcast Enable** Enables RX of broadcast packets.	0 RX of broadcast packets disabled. 1 RX of broadcast packets enabled.
MULTI CAST_FLT_ EN 6	0	**RX Multicast Filter Enable** Enables RX of the subset of multicast Ethernet packets enabled by the hash function.	0 RX Multicast filter disabled. 1 RX Multicast filter enabled.
MAC_ CHECK 5	0	**RX MAC Address Check** Enables RX MAC address checking.	0 RX MAC Address check disabled. 1 RX MAC Address check enabled.
LENGTH_ CHECK 4	0	**RX Packet Length Check** Enables RX packet length checking and discarding of packet fragments (packets <64 bytes).	0 Packet length check disabled. 1 Short RX packet discard enabled.
— 3–2	0	Reserved. Write to zero for future compatibility.	
LITTLE_ ENDIAN_E 1	0	**Little Endian Selection** Select the endian mode to use for Ethernet RX and TX data.	0 Use big-endian byte data representation for Ethernet RX and TX data. 1 Use little-endian byte data representation for Ethernet RX and TX data.
— 0	0	Reserved. Write to zero for future compatibility.	

18.4.1.27 Ethernet Miscellaneous Configuration (CPRIn_ETH_CONFIG_2)

address 0x20C (CPRIn_ETH_CONFIG_2)

	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	—															CRC_EN ABLE
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-61. CPRIn_ETH_CONFIG_2

Table 18-39. ETHI_CONFIG_2 Bit Descriptions

Name	Reset	Description	Setting
— 31–1	0	Reserved. Write to zero for future compatibility.	
CRC_ ENABLE 0		**CRC Enable** Enable the insertion of the Ethernet Frame Check Sequence at the end of the frame. If the bit is not set, the FCS must be included in the injected frame.	0 Function disabled. 1 Function enabled.

18.4.1.28 Ethernet RX Packet Discard (CPRIn_ETH_RX_CONTROL)

address 0x210 (CPRIn_ETH_RX_CONTRO)

	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	—															
W																RX_DISCARD
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-62. CPRIn_ETH_RX_CONTROL

Table 18-40. CPRIn_ETH_RX_CONTROL Bit Descriptions

Name	Reset	Description	Setting
— 31–1	0	Reserved. Write to zero for future compatibility.	
RX_DISCARD 0	0	**Discard Current Ethernet RX Frame** This bit indicates whether to discard the current Ethernet RX frame.	0 Do not discard current RX frame. 1 Discard current RX frame.

18.4.1.29 Ethernet RX External Status (CPRIn_ETH_RX_EX_STATUS)

address 0x228 (CPRIn_ETH_RX_EX_STATUS)

	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	—					MII_PROTOCOL_ERROR	SMALL_PACKET	OVERFLOW	DMAC_MISMATCH	CRC_ERROR	—	RX_ERROR	LONG_FRAME	SHORT_FRAME	—	
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-63. CPRIn_ETH_RX_EX_STATUS

Table 18-41. CPRIn_ETH_RX_EX_STATUS Bit Descriptions

Name	Reset	Description	Setting
— 31–11	0	Reserved. Write to zero for future compatibility.	
MII_PROTOCOL_ERROR 10	0	**MII Protocol Error** Indicates whether an MII protocol error occurred (4b/5b decoding violation).	0 No protocol error. 1 Protocol error.
SMALL_PACKET 9	0	**Small Packet** Indicates whether a frame that was too short was detected.	0 Short frame not detected 1 Frame too short, packet aborted.
OVERFLOW 8	0	**RX Buffer Overflow** Indicates whether an RX buffer overflow occurred.	0 No RX buffer overflow. 1 RX buffer overflow.

Table 18-41. CPRIn_ETH_RX_EX_STATUS Bit Descriptions

Name	Reset	Description	Setting
DMAC_ MISMATCH 7	0	**DMAC Address Mismatch** Indicates whether a DMAC address mismatch occurred.	0 No DMAC mismatch. 1 DMAC mismatch.
CRC_ ERROR 6	0	**CRC Error** Indicates whether a CRC error occurred.	0 No CRC error. 1 CRC error.
— 5	0	Reserved. Write to zero for future compatibility.	
RX_ERROR 4	0	**RX Error** Indicates whether an RX error occurred.	0 No RX error. 1 RX error, aborted from CPRI layer.
LONG_ FRAME 3	0	**Long Frame** Indicates whether a long frame was detected.	0 No long frame detected. 1 Long frame detected.
SHORT_ FRAME 2	0	**Short Frame** Indicates whether a short frame was detected.	0 No short frame detected. 1 Short frame detected.
— 1–0	0	Reserved. Write to zero for future compatibility.	

18.4.1.30 Ethernet 16 MSB of MAC Address (CPRIn_ETH_ADDR_MSB)

address 0x22C (CPRIn_ETH_ADDR_MSB)

	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	MAC[47:32]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-64. CPRIn_ETH_ADDR_MSB

CPRIn_ETH_ADDR_MSB together with CPRIn_ETH_ADDR_LSB register is used to configure Ethernet address for unicast MAC address filtering.

Table 18-42. CPRIn_ETH_ADDR_MSB Bit Descriptions

Name	Reset	Description	Setting
— 31–16	0	Reserved. Write to zero for future compatibility.	
MAC 15–0	0	**Ethernet Address Most Significant Bits** Contains the most significant bits [47:32] of the Ethernet MAC address.	

18.4.1.31 Ethernet 32 LSB of MAC Address (CPRIn_ETH_ADDR_LSB)

address 0x230 (CPRIn_ETH_ADDR_LSB)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MAC[31:16]															
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	MAC[15:0]															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-65. CPRIn_ETH_ADDR_LSB

CPRIn_ETH_ADDR_LSB together with CPRIn_ETH_ADDR_MSB register is used to configure Ethernet address for unicast MAC address filtering.

Table 18-43. CPRIn_ETH_ADDR_LSB Bit Descriptions

Name	Reset	Description	Setting
MAC 31–0	0	**Ethernet Address Least Significant Bits** Contains the least significant bits [31:0] of the Ethernet MAC address.	

18.4.1.32 Ethernet Small 32-Entries Hash Table to Filter Multicast Traffic (CPRIn_ETH_HASH_TABLE)

address 0x234 (CPRIn_ETH_HASH_TABLE)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HASH															
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	HASH															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-66. CPRIn_ETH_HASH_TABLE

Table 18-44. CPRIn_ETH_HASH_TABLE Bit Descriptions

Name	Reset	Description	Setting
HASH 31–0	0	**Ethernet Multicast Hash Table** Value used to configure the HASH table used for multicast MAC address filtering.	

18.4.1.33 Ethernet Configuration 3 (CPRIn_ETH_CONFIG_3)

address 0x244 (CPRIn_ETH_CONFIG_3)

	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	—	TX_START_THR							—						RX_CRC_EN	TX_ST_FWD
W																
RESET:	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0

Figure 18-67. CPRIn_ETH_CONFIG_3

Table 18-45. CPRIn_ETH_CONFIG_3 Bit Descriptions

Name	Reset	Description	Setting
— 31–15	0	Reserved. Write to zero for future compatibility.	
TX_START_THR 14–8	0000100	**Transmit Start Threshold** When store and hold mode is not selected, transmission starts when the specified number of 32-bit words are stored in the TX buffer.	
— 7–2	0	Reserved. Write to zero for future compatibility.	
RX_CRC_EN 1	1	**RX CRC Enable** Used to enable FCS validation of received packets. If the check fails, the packet is discarded.	0 No FCS validation. All packets accepted. 1 Enable FCS validation.
TX_ST_FWD 0	0	**TX Store and Forward Mode Select** Used to select the TX store and forward mode. If selected, a full packet is stored before starting transmission. Packets longer than the transmit buffer are aborted. In cut-through mode, the transmission starts when the buffer level exceeds the transmit start threshold.	0 Cut-through mode. 1 Transmit store and forward mode.

18.4.1.34 Ethernet Receive Frame Counter (CPRIn_ETH_CNT_RX_FRAME)

address 0x248 (CPRIn_ETH_CNT_RX_FRAME)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ETH_CNT_RX_FRAME															
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	ETH_CNT_RX_FRAME															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-68. CPRIn_ETH_CNT_RX_FRAME

Table 18-46. CPRIn_ETH_CNT_RX_FRAME Bit Descriptions

Name	Reset	Description	Setting
ETH_CNT_RX_FRAME 31–0	0	**Ethernet RX Frame Count** Records the number of received frames.	

18.4.1.35 HDLC Receive Status (CPRIn_HDLC_RX_STATUS)

address 0x300 (CPRIn_HDLC_RX_STATUS)

	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	—			RX_CRC_ERROR	—	RX_MII_ERROR	—	RX_OVERFLOW	—					RX_ABORT	—	
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-69. CPRIn_HDLC_RX_STATUS

Table 18-47. CPRIn_HDLC_RX_STATUS Bit Descriptions

Name	Reset	Description	Setting
— 31–13	0	Reserved. Write to zero for future compatibility.	
RX_CRC_ERROR 12	0	**HDLC RX CRC Error** Indicates when the HDLC RX CRC error is detected.	0 Frame not aborted due to CRC error. 1 Frame aborted due to CRC error.
11	0	Reserved. Write to zero for future compatibility.	
RX_MII_ERROR 10	0	**HDLC MII Error** Indicates when the HDLC RX mii protocol error is detected.	0 Frame not aborted due to mii protocol error. 1 Frame aborted due to mii protocol error.
— 9	0	Reserved. Write to zero for future compatibility.	

Table 18-47. CPRIn_HDLC_RX_STATUS Bit Descriptions

Name	Reset	Description	Setting
RX_OVERFLOW 8	0	**HDLC RX Overflow** Indicates when the HDLC RX buffer overflows.	0 Frame not aborted due to RX buffer overflow. 1 Frame aborted due to RX buffer overflow.
— 7–3	0	Reserved. Write to zero for future compatibility.	
RX_ABORT 2	0	**HDLC RX Packed Aborted** Indicates when the HDLC RX packet is aborted.	0 HDLC RX packet not aborted. 1 HDLC RX packet aborted
1–0	0	Reserved. Write to zero for future compatibility.	

18.4.1.36 HDLC Different Feature Enable/Disable and Trigger Enable (CPRIn_HDLC_CONFIG_1)

address 0x308 (CPRIn_HDLC_CONFIG_1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—													TRIG_TX_ABORT_EN	TRIG_TX_READY_EN	—
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	—	TRIG_RX_ABORT_EN	TRIG_RX_READY_EN	TRIG_TX_EN	TRIG_RX_EN	TRIG_EN	RX_LONG_FRAME_EN	—				LENGTH_CHECK	—		LITTLE_ENDIAN_H	—
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-70. CPRIn_HDLC_CONFIG_1

Bits 18–17 and 14–10 of this register must be set when using Ethernet. It's a must for HDLC data transfer between the framer and the DMA.

Table 18-48. CPRIn_HDLC_CONFIG_1 Bit Descriptions

Name	Reset	Description	Setting
— 31–19	0	Reserved. Write to zero for future compatibility.	
TRIG_TX_ABORT_EN 18	0	**HDLC TX Abort Trigger Enable** Enables TX abort trigger.	0 Trigger disabled. 1 Trigger enabled.
TRIG_TX_READY_EN 17	0	**HDLC TX Ready Trigger Enable** Enables TX ready trigger.	0 Trigger disabled. 1 Trigger enabled.
16–15	0	Reserved. Write to zero for future compatibility.	
TRIG_RX_ABORT_EN 14	0	**HDLC RX Abort Trigger Enable** Enables RX abort trigger.	0 Trigger disabled. 1 Trigger enabled.
TRIG_RX_READY_EN 13	0	**HDLC RX Ready Trigger Enable** Enables RX ready trigger.	0 Trigger disabled. 1 Trigger enabled.
TRIG_TX_EN 12	0	**HDLC TX Trigger Enable** Enables TX trigger.	0 Trigger disabled. 1 Trigger enabled.

Table 18-48. CPRIn_HDLC_CONFIG_1 Bit Descriptions

Name	Reset	Description	Setting
TRIG_RX_EN 11	0	**HDLC RX Trigger Enable** Enables RX trigger.	0 Trigger disabled. 1 Trigger enabled.
TRIG_EN 10	0	**HDLC Global Trigger Enable** Enables global HDLC Trigger.	0 Trigger disabled. 1 Trigger enabled.
RX_LONG_FRAME_EN 9	0	**RX Long HDLC Frame Enable** Enables receipt of RX frames >1536 bytes.	0 Long HDLC frame RX disabled. 1 Long HDLC frame RX enabled.
— 8–5	0	Reserved. Write to zero for future compatibility.	
LENGTH_CHECK 4	0	**RX Packet Length Check** Enables RX packet length checking and discard of packet fragments (packets <64 bytes).	0 Packet length check disabled. 1 Short RX packet discard enabled.
— 3–2	0	Reserved. Write to zero for future compatibility.	
LITTLE_ENDIAN_H 1	0	**Little Endian Selection** Select the endian mode to use for HDLC RX and TX data.	0 Use big-endian byte data representation for HDLC RX and TX data. 1 Use little-endian byte data representation for HDLC RX and TX data.
— 0	0	Reserved. Write to zero for future compatibility.	

18.4.1.37 HDLC Miscellaneous Configuration (CPRIn_HDLC_CONFIG_2)

address 0x30C (CPRIn_HDLC_CONFIG_2)

	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	—															CRC_ENABLE
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-71. CPRIn_HDLC_CONFIG_2

Table 18-49. CPRIn_HDLC_CONFIG_2 Bit Descriptions

Name	Reset	Description	Setting
— 31–1	0	Reserved. Write to zero for future compatibility.	
CRC_ENABLE 0		**CRC Enable** Enables the insertion of the HDLC CRC at the end of the frame. If the bit is not set, the CRC must be included in the injected frame.	0 Function disabled. 1 Function enabled.

18.4.1.38 HDLC RX Packet Discard (CPRIn_HDLC_RX_CONTROL)

address 0x310 (CPRIn_HDLC_RX_CONTROL)

	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	—															
W																RX_ DISCARD
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-72. CPRIn_HDLC_RX_CONTROL

Table 18-50. CPRIn_HDLC_RX_CONTROL Bit Descriptions

Name	Reset	Description	Setting
— 31–1	0	Reserved. Write to zero for future compatibility.	
RX_ DISCARD 0	0	**Discard Current HDLC RX Frame** This bit indicates whether to discard the current HDLC RX frame.	0 Do not discard current RX frame. 1 Discard current RX frame.

18.4.1.39 HDLC RX External Status (CPRIn_HDLC_RX_EX_STATUS)

address 0x328 (CPRIn_HDLC_RX_EX_STATUS)

	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	—					MII_PRO TOCOL_ ERROR	—	OVER FLOW	—	CRC_ ERROR	—	RX_ ERROR	—			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-73. CPRIn_HDLC_RX_EX_STATUS

Table 18-51. CPRIn_HDLC_RX_EX_STATUS Bit Descriptions

Name	Reset	Description	Setting
— 31–11	0	Reserved. Write to zero for future compatibility.	
MII_PROTO COL_ ERROR 10		**MII Protocol Error** Indicates whether an MII protocol error occurred.	0 No protocol error. 1 Protocol error.
9		Reserved. Write to zero for future compatibility.	
OVERFLOW 8		**RX Buffer Overflow** Indicates whether an RX buffer overflow occurred.	0 No RX buffer overflow. 1 RX buffer overflow.
— 7	0	Reserved. Write to zero for future compatibility.	

Table 18-51. CPRIn_HDLC_RX_EX_STATUS Bit Descriptions

Name	Reset	Description	Setting
CRC_ERROR 6		**CRC Error** Indicates whether a CRC error occurred.	0 No CRC error. 1 CRC error.
— 5	0	Reserved. Write to zero for future compatibility.	
RX_ERROR 4		**RX Error** Indicates whether an RX error occurred.	0 No RX error. 1 RX error, aborted from CPRI layer.
— 3–0	0	Reserved. Write to zero for future compatibility.	

18.4.1.40 HDLC Configuration 3 (CPRIn_HDLC_CONFIG_3)

address 0x344 (CPRIn_HDLC_CONFIG_3)

	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	—	TX_START_THR							—						RX_CRC_EN	TX_ST_FWD
W																
RESET:	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0

Figure 18-74. CPRIn_HDLC_CONFIG_3

Table 18-52. CPRIn_HDLC_CONFIG_3 Bit Descriptions

Name	Reset	Description	Setting
— 31–15	0	Reserved. Write to zero for future compatibility.	
TX_START_THR 14–8	0000100	**Transmit Start Threshold** When store and hold mode is not selected, transmission starts when the specified number of 32-bit words are stored in the TX buffer. **Note:** The field is variable in size and is based on the value of the WIDTH_CPRIn_HDLC_BUFFER generic value (x) used in the system. The field ranges from bit (x –1)+8 to bit 8.	
— 2–7	0	Reserved. Write to zero for future compatibility.	
RX_CRC_EN 1	1	**RX CRC Enable** Used to enable CRC checking of received packets. If the check fails, the packet is discarded.	0 No CRC checking. All packets accepted. 1 Enable CRC checking.
TX_ST_FWD 0	0	**TX Store and Forward Mode Select** Used to select the TX store and forward mode. If selected, a full packet is stored before starting transmission. Packets longer than the transmit buffer are aborted. In cut-through mode, the transmission starts when the buffer level exceeds the transmit start threshold.	0 Cut-through mode. 1 Transmit store and forward mode.

18.4.1.41 HDLC Receive Frame Counter (CPRIn_HDLC_CNT_RX_FRAME)

address 0x348 (CPRIn_HDLC_CNT_RX_FRAME)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HDLC_CNT_RX_FRAME															
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	HDLC_CNT_RX_FRAME															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-75. CPRIn_HDLC_CNT_RX_FRAME

Table 18-53. CPRIn_HDLC_CNT_RX_FRAME Bit Descriptions

Name	Reset	Description	Setting
HDLC_CNT_RX_FRAME 31–0	0	**HDLC RX Frame Count** Records the number of received frames.	

18.4.2 CPRI Complex Registers

The following sections describe the CPRI complex registers.

18.4.2.1 Receive IQ MBus Transaction Size (CPRInRIQMTS)

address 0x400 (CPRInRIQMTS)

	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	—				0	—					RIQMTS					
W					RSM											
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-76. CPRInRIQMTS

Table 18-54. CPRInRIQMTS Bit Descriptions

Name	Reset	Description	Setting
31–12	0	Reserved. Write to zero for future compatibility.	
RSM **11**	0	**Receive Shared Mode** Selects whether the CPRI pair operates in shared mode. In shared mode the DMA buffer of both CPRIs in the pair can be used by the only enabled CPRI. The other CPRI must be disabled in this case. (Write-only bit, read always 0)	0b0 normal operation 0b1 shared mode
10–6	0	Reserved. Write to zero for future compatibility.	
RIQMTS 5–0		**Receive IQ MBus Transaction Generation Size** Selects the amount of data that should be ready in the IQ DMA buffer in order to generate a write access on the MBus **256 bytes transaction generation size is allowed if one of the following conditions is met:** - CPRIxRGCM[[RMAP_AC] < 0xD - CPRIxRIQMTS[RSM] bit is set **512 bytes transaction generation size is allowed if one of the following condition is met:** - CPRIxRGCM[[RMAP_AC] < 0x7 - CPRIxRIQMTS[RSM] bit is set and CPRIxRGCM[[RMAP_AC] < 0xD **1024 bytes transaction generation size is allowed in the following condition:** - CPRIxRGCM[[RMAP_AC] < 0x4 -- CPRIxRIQMTS[RSM] bit is set and CPRIxRGCM[[RMAP_AC] < 0x7 **2048 bytes transaction generation size is allowed in the following condition:** - CPRIxRGCM[[RMAP_AC] =1 -- CPRIxRIQMTS[RSM] bit is set and CPRIxRGCM[[RMAP_AC] < 0x3	0x1 64 bytes ready 0x2 128 bytes ready 0x4 256 bytes ready 0x8 512 bytes ready 0x10 1024 bytes ready 0x20 2048 bytes ready **The access size on the MBus is maximum 256 bytes. In case of more data (512 bytes or more) a number of 256 byte accesses are generated.**

18.4.2.2 Receive IQ Second Destination Mbus Transaction Size (CPRInRIQSDMTS)

address 0x404 (CPRInRIQSDMTS)

	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	—															
W											RIQSDMTS					
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-77. CPRInRIQSDMTS

This register should be programmed only in Receive Multi Unicast mode (CPRInRGCM[RMUM] = 0x1.)

Table 18-55. CPRInRIQSDMTS Bit Descriptions

Name	Reset	Description	Setting
31–12	0	Reserved. Write to zero for future compatibility.	
RIQSDMTS 5–0		**Receive IQ Second Destination MBus Transaction Generation Size** Selects the amount of data that should be ready in the IQ DMA buffer in order to generate a Second Destination write access on the MBus Note: The RIQSDMTS must be smaller or equal to CPRInRIQMTS[RIQMTS].	0x1 64 bytes ready 0x2 128 bytes ready 0x4 256 bytes ready 0x8 512 bytes ready 0x10 1024 bytes ready 0x20 2048 bytes ready **Note:** The access size on the MBus is maximum 256 bytes. In case of more data (512 bytes or more) a number of 256 byte accesses are generated.

18.4.2.3 Transmit IQ MBus Transaction Size (CPRInTIQMTS)

address 0x408 (CPRInTIQMTS)

	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	—				0	—					TIQMTS					
W					TSM											
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-78. CPRInTIQMTS

Table 18-56. CPRInTIQMTS Bit Descriptions

Name	Reset	Description	Setting
31–12	0	Reserved. Write to zero for future compatibility.	
TSM **11**	0	**Transmit Shared Mode** Selects if the CPRI pair operates in shared mode. In shared mode the DMA buffer of both CPRIs in the pair can be used by the only enabled CPRI. The other CPRI must be disabled in this case. (Write-only bit, read always 0)	0x0 normal operation 0x1 shared mode
10–6	0	Reserved. Write to zero for future compatibility.	
TIQMTS 5–0		**Transmit IQ MBus Transaction Generation Size** Selects the amount of empty place that should be in the IQ DMA buffer in order to generate a read access on the MBus **256 bytes transaction size is allowed if one of the following conditions is met:** - CPRIxTGCM[[TMAP_AC] < 0xD - CPRIxTIQMTS[TSM] bit is set **512 bytes transaction size is allowed if one of the following condition is met:** - CPRIxTGCM[[TMAP_AC] < 0x7 - CPRIxTIQMTS[TSM] bit is set and CPRIxTGCM[[TMAP_AC] < 0xD **1024 bytes transaction size is allowed in the following condition:** - CPRIxTGCM[[TMAP_AC] < 0x4 - CPRIxTIQMTS[TSM] bit is set and CPRIxTGCM[[RMAP_AC] < 0x7 **2048 bytes transaction size is allowed in the following condition:** - CPRIxTGCM[[TMAP_AC] =1 -- CPRIxTIQMTS[TSM] bit is set and CPRIxTGCM[[TMAP_AC] < 0x3	0x1 64 bytes empty 0x2 128 bytes empty 0x4 256 bytes empty 0x8 512 bytes empty 0x10 1024 bytes empty 0x20 2048 bytes empty **Note:** The access size on the MBus is maximum 256 bytes. In case of more data (512 bytes or more) a number of 256 byte accesses are generated.

18.4.2.4 Receive VSS MBus Transaction Size (CPRInRVSSMTS)

address 0x40c (CPRInRVSSMTS)

	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	—												RVSSMTS			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 18-79. CPRInRVSSMTS

Table 18-57. CPRInRVSSMTS Bit Descriptions

Name	Reset	Description	Setting	
31–4	0	Reserved. Write to zero for future compatibility.		
RVSSMTS 3–0	8	**Receive VSS MBus Transaction Generation Size** Selects the amount of data that should be ready in the VSS DMA buffer in order to generate a write access on the MBus	0x0 0x1 0x2 0x4 0x8	Reserved. 16 bytes ready. 32 bytes ready. 64 bytes ready. 128 bytes ready.

18.4.2.5 Transmit VSS MBus Transaction Size (CPRInTVSSMTS)

address 0x410 (CPRInTVSSMTS)

	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	—															
W													TVSSMTS			
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 18-80. CPRInTVSSMTS

Table 18-58. CPRInTVSSMTS Bit Descriptions

Name	Reset	Description	Setting	
— 31–4	0	Reserved. Write to zero for future compatibility.		
TVSSMTS 3–0	0	**Transmit VSS MBus Transaction Size** Selects the amount of empty space that should be in the VSS DMA buffer in order to generate a read access on the MBus	0x0 0x1 0x2 0x4 0x8	Reserved. 16 bytes empty. 32 bytes empty. 64 bytes empty. 128 bytes empty

18.4.2.6 Receive IQ Second Destination Base Address (CPRInRIQSDBA)

This register determines the location of AxC 0 in the second destination address. The AxC buffers are located in the second destination in consecutive order. This register should be programmed in Receive Multi Unicast Mode(CPRInRGCM[RMUCM]=0x1) and in Receive Multicast Mode(CPRInRGCM[RMCM]=0x1).

address 0x45c (CPRInRIQSDBA)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RIQSDBA															
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	RIQSDBA															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-81. CPRInRSDBA

Table 18-59. CPRInRSDBA Bit Descriptions

Name	Reset	Description	Setting
RIQSDBA 31–0	0	Receive Second Destination Base Address. This field determines the location of the AxC 0 in the second destination. Note the base address must be aligned to 16 bytes.	0x0– 0xFFFFFFF0

18.4.2.7 Receive IQ Buffer Size (CPRInRIQBS)

address 0x460 (CPRInRIQBS)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								RIQBS							
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	RIQBS															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 18-82. CPRInRIQBS

Table 18-60. CPRInRIQBS Bit Descriptions

Name	Reset	Description	Setting
31–24	0	Reserved. Write to zero for future compatibility.	
RIQBS 23–0	0x3F	**Receive IQ Buffer Size** RIQBS field is equal to the receive IQ buffer size in bytes minus 1. The buffer size should be an integer multiple of the receive IQ MBus Transaction Generation Size (Register CPRInRIQMTS). (The 6 LSBs are always ones) **Note:** The buffer size of all the antenna carriers is the same.	0x00003f–0xFFFFBF

18.4.2.8 Receive IQ Second Destination Buffer Size (CPRInRIQSDBS)

address 0x464 (CPRInRIQSDBS)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								RIQSDBS							
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	RIQSDBS															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 18-83. CPRInRIQSDBS

Table 18-61. CPRInRIQSDBS Bit Descriptions

Name	Reset	Description	Setting
31–24	0	Reserved. Write to zero for future compatibility.	
RIQSDBS 23–0	0x3F	**Receive IQ Second Destination Buffer Size** RIQSDBS field is equal to the second destination receive IQ buffer size in bytes minus 1. The buffer size should be an integer multiple of the second destination receive IQ MBus Transaction Generation Size (Register CPRInRIQSDMTS). (The 6 LSBs are always ones) **Note:** The buffer size of all the antenna carriers is the same.	0x00003f–0xFFFFBF

18.4.2.9 Transmit IQ Buffer Size (CPRInTIQBS)

address 0x468 (CPRInTIQBS)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								TIQBS							
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	TIQBS															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 18-84. CPRInTIQBS

Table 18-62. CPRInTIQBS Bit Descriptions

Name	Reset	Description	Setting
31–24	0	Reserved. Write to zero for future compatibility.	
TIQBS 23–0	0x3F	**Transmit IQ Buffer Size** TIQBS field is equal to the transmit IQ buffer size in bytes minus 1. The buffer size should be an integer multiple of the transmit IQ MBus Transaction Generation Size (Register CPRInTIQMTS). (The 6 LSBs are always ones) **Note:** The buffer size of all the antenna carriers is the same.	0x00003f–0xFFFFBF

18.4.2.10 Receive VSS Buffer Size (CPRInRVSSBS)

address 0x46c (CPRInRVSSBS)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								RVSSBS							
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	RVSSBS															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Figure 18-85. CPRInRVSSBS

Table 18-63. CPRInRVSSBS Bit Descriptions

Name	Reset	Description	Setting
31–24	0	Reserved. Write to zero for future compatibility.	
RVSSBS 23–0	0	**Receive VSS Buffer Size** RVSSBS field is equal to the receive VSS buffer size in bytes minus 1. The buffer size should be integer multiple of the receive VSS MBus transaction size (Register CPRInRVSSMTS). (The 4 LSBs are always ones).	0x00000F–0xFFFF7F

18.4.2.11 Transmit VSS Buffer Size (CPRInTVSSBS)

address 0x470 (CPRInTVSSBS)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								TVSSBS							
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	TVSSBS															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Figure 18-86. CPRInTVSSBS

Table 18-64. CPRInTVSSBS Bit Descriptions

Name	Reset	Description	Setting
— 31–24	0	Reserved. Write to zero for future compatibility.	
TVSSBS 23–0	0	**Transmit VSS Buffer Size** The TVSSBS field is equal to the transmit VSS buffer size in bytes minus 1. The buffer size should be integer multiple of the transmit VSS MBus transaction size. (Register CPRInRVSSMTS). (The 4 LSBs are always ones).	0x00000F–0xFFFF7F

18.4.2.12 Receive Ethernet Buffer Size (CPRInRETHBS)

address 0x474 (CPRInRETHBS)

	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	RETHBS															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Figure 18-87. CPRInRETHBS

Table 18-65. CPRInRETHBS Bit Descriptions

Name	Reset	Description	Setting
31–16	0	Reserved. Write to zero for future compatibility.	
RETHBS 15–0	0	**Receive ETH Buffer Size** The RETHBS field is equal to the receive Ethernet buffer size in bytes minus 1. The buffer size should be aligned to 16 bytes. (The 4 LSBs are always ones) **Note:** Always make sure that the definition is larger than the expected Ethernet packet size.	0x000F–0xFFBF

18.4.2.13 Receive HDLC Buffer Size (CPRInRHDLCBS)

address 0x478 (CPRInRHDLCBS)

	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	RHDLCBS															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Figure 18-88. CPRInRHDLCBS

Table 18-66. CPRInRHDLCBS Bit Descriptions

Name	Reset	Description	Setting
— 31–16	0	Reserved. Write to zero for future compatibility.	
RHDLCBS 15–0	0	**Receive HDLC Buffer Size** The RHDLCBS field is equal to the receive HDLC buffer size in bytes minus 1. The buffer size should be aligned to 16 bytes. (The 4 LSBs are always ones)	0x000F–0xFFBF

18.4.2.14 Receive VSS Base Address (CPRInRVSSBA)

address 0x47c (CPRInRVSSBA)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RVSSBA															
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	RVSSBA															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-89. CPRInRVSSBA

Table 18-67. CPRInRVSSBA Bit Descriptions

Name	Reset	Description	Setting
REBDRBA 31–0	0	**Receive VSS buffer Base Address** Indicates the receive VSS buffer base address in system memory. The receive VSS buffer base address should be aligned to the receive VSS MBus transaction size (Register CPRInRVSSMTS).	0x0–0xFFFFFFF0

18.4.2.15 Transmit VSS Base Address (CPRInTVSSBA)

address 0x480 (CPRInTVSSBA)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TVSSBA															
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	TVSSBA															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-90. CPRInTVSSBA

Table 18-68. CPRInTVSSBA Bit Descriptions

Name	Reset	Description	Setting
TEBDRBA 31–0	0	**Transmit VSS Buffer Base Address** Indicates the transmit VSS buffer base address in the system memory. The transmit VSS buffer base address should be aligned to the transmit VSS MBus transaction size (Register CPRInTVSSMTS)	0x0–0xFFFFFFF0

18.4.2.16 Receive Ethernet BD Ring Base Address (CPRInREBDRBA)

address 0x484 (CPRInREBDRBA)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	REBDRBA															
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	REBDRBA															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-91. CPRInREBDRBA

Table 18-69. CPRInREBDRBA Bit Descriptions

Name	Reset	Description	Setting
REBDRBA 31–0	0	**Receive Ethernet Buffer Descriptor Ring Base Address** Indicates the receive Ethernet buffer descriptor ring base address. The buffer descriptor ring base address should be aligned to 16bytes.	0x0–0xFFFFFFF0

18.4.2.17 Transmit Ethernet BD Ring Base Address (CPRInTEBDRBA)

address 0x488 (CPRInTEBDRBA)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TEBDRBA															
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	TEBDRBA															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-92. CPRInTEBDRBA

Table 18-70. CPRInTEBDRBA Bit Descriptions

Name	Reset	Description	Setting
TEBDRBA 31–0	0	**Transmit Ethernet Buffer Descriptor Ring Base Address** Indicates the transmit Ethernet buffer descriptor ring base address. The buffer descriptor ring base address should be aligned to 16 bytes.	0x0–0xFFFFFFF0

18.4.2.18 Receive HDLC BD Ring Base Address (CPRInRHBDRBA)

address 0x48c (CPRInRHBDRBA)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RHBDRBA															
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	RHBDRBA															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-93. CPRInRHBDRBA

Table 18-71. CPRInRHBDRBA Bit Descriptions

Name	Reset	Description	Setting
RHBDRBA 31–0	0	**Receive HDLC Buffer Descriptor Ring Base Address** Indicates the receive HDLC buffer descriptor ring base address. The buffer descriptor ring base address should be aligned to 16bytes.	0x0–0xFFFFFFF0

18.4.2.19 Transmit HDLC BD Ring Base Address (CPRInTHBDRBA)

address 0x490 (CPRInTHBDRBA)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	THBDRBA															
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	THBDRBA															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-94. CPRInTHBDRBA

Table 18-72. CPRInTHBDRBA Bit Descriptions

Name	Reset	Description	Setting
THBDRBA 31–0	0	**Transmit HDLC Buffer Descriptor Ring Base Address** Indicates the transmit HDLC buffer descriptor ring base address. The buffer descriptor ring base address should be aligned to 16 bytes.	0x0–0xFFFFFFF0

18.4.2.20 Receive Ethernet BD Ring Size (CPRInREBDRS)

address 0x494 (CPRInREBDRS)

	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 W	—								REBDRS							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-95. CPRInREBDRS

Table 18-73. CPRInREBDRS Bit Descriptions

Name	Reset	Description	Setting	
— 31–8	0	Reserved. Write to zero for future compatibility.		
REBDRS 7–0	0	**Receive Ethernet Buffer Descriptor Ring Size** Defines the number of buffer descriptors in the receive Ethernet ring. The size of the BD ring must be at least 4 and an even number.	0x04 0x06 0x08 .... 0xFE	4 buffer descriptors 6 buffer descriptors 8 buffer descriptors 254 buffer descriptors

18.4.2.21 Transmit Ethernet BD Ring Size (CPRInTEBDRS)

address 0x498 (CPRInTEBDRS)

	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	—								TEBDRS							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-96. CPRInTEBDRS

Table 18-74. CPRInTEBDRS Bit Descriptions

Name	Reset	Description	Setting	
— 31–8	0	Reserved. Write to zero for future compatibility.		
TEBDRS 7–0	0	**Transmit Ethernet Buffer Descriptor Ring Size** Defines the number of buffer descriptors in the transmit Ethernet ring. Note: The size of the BD ring must be even	0x04 0x06 0x08 ..... 0xFE	4 buffer descriptors 6 buffer descriptors 8 buffer descriptors 254 buffer descriptors

18.4.2.22 Receive HDLC BD Ring Size (CPRInRHBDRS)

address 0x49c (CPRInRHBDRS)

	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	—								RHBDRS							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-97. CPRInRHBDRS

Table 18-75. CPRInRHBDRS Bit Descriptions

Name	Reset	Description	Setting	
— 31–8	0	Reserved. Write to zero for future compatibility.		
RHBDRS 7–0	0	**Receive HDLC Buffer Descriptor Ring Size** Defines the number of buffer descriptors in the receive HDLC ring. Note: the size of the BD ring must be even	0x04 0x06 0x08 ... 0xFE	4 buffer descriptors 6 buffer descriptors 8 buffer descriptors 254 buffer descriptors

18.4.2.23 Transmit HDLC BD Ring Size (CPRInTHBDRS)

address 0x4a0 (CPRInTHBDRS)

	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	—								THBDRS							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-98. CPRInTHBDRS

Table 18-76. CPRInTHBDRS Bit Descriptions

Name	Reset	Description	Setting
— 31–8	0	Reserved. Write to zero for future compatibility.	
THBDRS 7–0	0	**Transmit HDLC Buffer Descriptor Ring Size** Defines the number of buffer descriptors in the transmit Ethernet ring. Note: The size of the ring must be even	0x04 4 buffer descriptors 0x06 6 buffer descriptors 0x08 8 buffer descriptors ... 0xFE 254 buffer descriptors

18.4.2.24 Receive General CPRI Mode (CPRInRGCM)

address 0x4a8 (CPRInRGCM)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ROVS	RMUCM	—	RMCM	—										RSW	
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	RTM	—										RMAP_AC				
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-99.

Figure 18-100. CPRInRGCM

Table 18-77. CPRInRGCM Bit Descriptions

Name	Reset	Description	Setting
ROVS 31	0	**Receive Double Oversampling Mode** If this bit is set the received oversampling data is spread to two different buffers in the SOC memories. **Note**: In This mode the RSW field must be set to 0x2.(8-bit sample width.)	0b0 Normal Mode 0b1 Receive double oversampling mode

Table 18-77. CPRInRGCM Bit Descriptions (Continued)

Name	Reset	Description	Setting
RMUCM 30	0	**Receive Multi Unicast Mode** Determines if the CPRI works in multi unicast mode or not. In multi unicast mode the received data is send to two different destinations in the SOC memory.(First and Second destination). If this bit is set RMCM bit must be zero	0b0 Non multi cast mode 0b1 Receive multi unicast mode
29	0	Reserved. Write to zero for future compatibility.	
RMCM 28	0	**Receive MultiCast Mode** Determines if the CPRI works in multicast mode or not. In multicast mode the received data is sent to the SOC memory (as a first destination) and directly to the MAPLE (as a second destination). If this bit is set RMUCM bit must be zero.	0b0 Non multicast mode 0b1 Receive multicast mode
— 27–18	0	Reserved. Write to zero for future compatibility.	
RSW 17–16	0	**Receive Sample Width** Determines the receive sample width. Must be the equal to the TSW field in CPRInTGCM.	0x0 16 bit receive sample width 0x1 15 bit receive sample width 0x2 8 bit receive sample width
RTM 15	0	**Receive Transparent Mode** For details on transparent mode, refer to **Section 18.3.4.** Bits ROVS,RMUCM and RMCM should be zero.	0 Normal mode 1 Transparent mode
— 14–5	0	Reserved. Write to zero for future compatibility.	
RMAP_AC 4–0	0	**Receive Number of Antenna Carriers** This field describes the maximal number of the active AxCs. In basic mapping mode this field should get the same value as field MAP_AC in the CPRIn_MAP_CNT_CONFIG register.	0x1 1 antenna carrier 0x2 2 antenna carrier 0x3 3 antenna carrier : 0x18 24 antenna carriers 0x19–0x1F not valid

18.4.2.25 Transmit General CPRI Mode (CPRInTGCM)

address 0x4ac (CPRInTGCM)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—														TSW	
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	TTM	—										TMAP_AC				
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-101. CPRInTGCM

Table 18-78. CPRInTGCM Bit Descriptions

Name	Reset	Description	Setting
— 31–18	0	Reserved. Write to zero for future compatibility.	

Table 18-78. CPRInTGCM Bit Descriptions (Continued)

Name	Reset	Description	Setting
TSW 17–16	0	**Transmit Sample Width** Determines the transmit sample width. Must be the equal to the RSW field in register CPRInRGCM	0x0 16 bit transmit sample width 0x1 15 bit transmit sample width 0x2 8 bit transmit sample width
TTM 15	0	**Transmit Transparent Mode** For details on transparent mode, refer to **Section 18.3.4**	0 Normal mode 1 Transparent mode
— 14–5	0	Reserved. Write to zero for future compatibility.	
TMAP_AC 4–0	0	**Transmit Number of Antenna Carriers** This field describes the maximal number of the active AxCs. In basic mapping mode, this field should get the same value as field MAP_AC in the CPRIn_MAP_CNT_CONFIG register. Depending on the value of TMAP_AC, you must write the value designated below to the CPRInTCFBS register: • If TMAP_AC is 0x17 or 0x18, TCFBS = 0x10. • IF TMAP_AC is 0x10–0x16, TCFBS=0x14. • If TMAP_AC is 0x4–0x9, TCFBS=0x28. • If TMAP_AC is 0x1–0x3, TCFBS=0x50.	0x1 1 antenna carrier 0x2 2 antenna carrier 0x3 3 antenna carrier : 0x18 24antenna carriers 0x19–0x1F not valid

18.4.2.26 Transmit Synchronization Configuration Register (CPRInTSCR)

address 0x4b4 (CPRInTSCR)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TSA	—										TNS				
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	TNS															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-102. CPRInTSCR

In WiMAX, even if only one CPRI operates, it may be desirable to synchronize the transmit to the 10 ms frame. In the case of n AxCs with oversampling m, TNS should be n*m*256*150 (n*m samples in a basic frame, 256 basic frame in a hyperframe, 150 hyperframes in a 10 ms frame)

Table 18-79. CPRInTSCR Bit Descriptions

Name	Reset	Description	Setting
TSA 31	0	**Transmit Synchronization Active** Determines whether synchronization is required. The synchronization is done to the 10ms frame. If this bit is set TNS field should be written.	0b0 Transmit synchronization is not required 0b1 Transmit synchronization is required
— 30–21	0	Reserved. Write to zero for future compatibility.	

Table 18-79. CPRInTSCR Bit Descriptions

Name	Reset	Description	Setting
TNS 20–0	0	**Transmit Number of Samples** Determines the number of transmit samples during a single Node B frame. The value of 0x0 is valid only if TSA is cleared. TNS can range from 0x 1 to 0x177000 (1 to 153600 samples in the Transmit Node B frame).	

18.4.2.27 Transmit CPRI Framer Buffer Size (CPRInTCFBS)

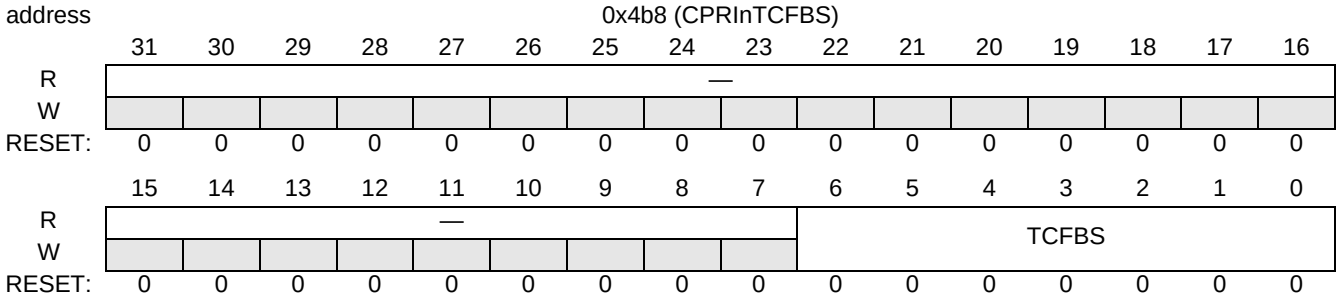


Figure 18-103. CPRInTCFBS

The buffer size actually determines the CPRI framer transmit latency. The CPRIn framer latency in chips time for the case of Sample Width 15 or 16 is TCFBS/(Oversampling Ratio)

The default value depends on the number of AXCs as defined by CPRInTGCM[TMAP_AC]. The default value can be overwritten by using TCFBS, but requires consideration of the following:

- A larger buffer means higher transmit latency. In the case of a few AxCs with no oversampling, the default buffer size is quite large and can be set to a smaller value to reduce latency.
- Using a buffer that is too small may cause data overrun.
- For higher sampling rates, the buffer size should be larger.

Table 18-80. CPRInTCFBS Bit Descriptions

Name	Reset	Description	Setting
— 31–7	0	Reserved. Write to zero for future compatibility.	

Table 18-80. CPRInTCFBS Bit Descriptions

Name	Reset	Description	Setting
TCFBS 6–0	0	Transmit CPRI Framer Buffer Size. This field determines the transmit AxCs buffers size inside the CPRI framer. Here are the default values as function of tmap_ac field: If TMAP_AC >16 - 0x10 If 13=< TMAP_AC=<16- 0x12 If 9=<TMAP_AC =<12 - 0x18 If 7<=TMAP_AC =<8 - 0x20 If 5<=TMAP_AC =<6 - 0x24 If TMAP_AC =4 - 0x30 If TMAP_AC =3 - 0x48 If TMAP_AC =2 - 0x60 If TMAP_AC =1 - 0x90	0x0 Use default value 0x10 Transmit buffer size inside the CPRI framer is 16 entries. 0x12 Transmit buffer size inside the CPRI framer is 18 entries. ... 0x90 Transmit buffer size inside the CPRI framer is 144 entries. **Note:** Each entry is 32 bits.

18.4.2.28 Transmit Control Table Insert Enable 1(CPRInTCTIE1)

address 0x4bc (CPRInTCTIE1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ITCW_79_64															
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	ITCW_15_0															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-104. CPRInTCTIE1

This register determines if the CPRIn framer will generate the control words 0–15 and 64–79 or these control words should be taken from the transmit control word table.

Table 18-81. CPRInTCTIE1 Bit Descriptions

Name	Reset	Description	Setting
ITCW_79_64 31–16	0	**Insert transmit control words 79–64** This field determines if the control words 64–79 should be inserted by the table or by the framer. If this bit is set than the control word should be inserted by the transmit control table. Bit 31 refer to control word 79 and bit 16 refer to control word 64. **Note:** It's recommended to clear bits 16, 17, and 18. Bit 16 refers to timing word, bit 17 refers to HDLC, bit 18 refers to L1 inband.	0x0 The control words 79–64 should be generated by the CPRI framer. : 0xFFFF The control words 79–64 should be taken by the CPRI framer from the control table.

Table 18-81. CPRInTCTIE1 Bit Descriptions

Name	Reset	Description	Setting
ITCW_15_0 15–0	0	**Insert Transmit Control Words 15–0** This field determines if the control words 15–0 should be inserted by the table or by the framer. If this bit is set than the control word should be inserted by the transmit control table. Bit 15 refer to control word 15 and bit 0 refer to control word 0. **Note:** Clear bits 0, 1,. and 2. Bit 0 refers to the Comma byte,bit 1 refers to HDLC, bit 2 refers to L1 inband.	0x0000 The control words 15 – 0 should be generated by the CPRI framer. : 0xFFFF The control words 15–0 should be taken by the CPRI framer from the control table.

18.4.2.29 Transmit Control Table Insert Enable 2(CPRInTCTIE2)

address 0x4c0 (CPRInTCTIE2)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ITCW_207_192															
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	ITCW_143_128															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-105. CPRInTCTIE2

This register determines if the CPRIn framer will generate the control words 143–128 and 207–192 or these control words should be taken from the transmit control word table.

Table 18-82. CPRInTCTIE2 Bit Descriptions

Name	Reset	Description	Setting
ITCW_207_192 31–16	0	**Insert transmit control words 207–192** This field determines if the control words 207–192 should be inserted by the table or by the framer. If this bit is set than the control word should be inserted by the transmit control table. Bit 31 refers to control word 207 and bit 16 refer to control word 192. **Note:** It's recommended to clear bits 16, 17 and 18. Bit 16 refers to timing word,bit 17 refers to HDLC, bit 18 refers to L1 inband.	0x0000 The control words 207–192 should be generated by the CPRI framer. : 0xFFFF The control words 207–192 should be taken by the CPRI framer from the control table.
ITCW_143_128 15–0	0	**Insert transmit control words 143–128** This field determines if the control words 143–128 should be inserted by the table or by the framer. If this bit is set than the control word should be inserted by the transmit control table. Bit 15 refers to control word 143 and bit 0 refer to control word 128. **Note:** It's recommended to clear bits 0,1 and 2. Bit 0 refers to timing word,bit 1 refers to HDLC, bit 2 refers to L1 inband.	0x0000 The control words 143–128 should be generated by the CPRI framer. : 0xFFFF The control words 143–128 should be taken by the CPRI framer.from the control table.

18.4.2.30 Timer Configuration (CPRInTMRC)

address 0x4c8 (CPRInTMRC)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—	TSO	ESA		SSME	SSM	—									
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	NUMERR		—						SDLY							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 18-106. CPRInTMRC

This register determines the CPRIn timer operation mode. For details, refer to **Section 18.3.8, *Timers*.**

Table 18-83. CPRInTMRC

Name	Reset	Description	Setting
— 31	0	Reserved. Write to zero for future compatibility.	
TSO 30	0	**Transmit Sync Output** When this bit is set to a value of 1, the sync out signal is driven out through the GPIO ports **Note:** This field is valid only if ESA field is clear	0 Transmit sync is not an output. 1 Transmit sync should be an output from the device.
ESA 29–28	0	**External Sync Is Active** This field indicates whether the local timer works independently or the sync is generated in reference to external sync or the received sync of the CPRI Framer pair.	0x0 10 ms sync pulse is generated locally 0x1 10 ms sync pulse is generated refer to external sync signal 0x2 10 ms sync pulse is generated referred to received sync of the CPRI Framer pair
SSME 27	0	**Sync State Machine Enable**	0 Sync state machine disabled 1 Sync state machine enabled
SSM 26	0	**Shared Sync Mode.** Determines whether the CPRIn Framer receives the transmit sync signal generated by local TIMERn (independent mode) or receives the transmit sync signal generated by local TIMER1 If the SSM bit is cleared (independent mode), each CPRI Framer receives the transmit sync signal from its local timer. **Note:** Do not change this bit during normal operation.	0 Independent sync mode 1 Shared sync mode.
— 27–16	0	Reserved. Write to zero for future compatibility.	
NUMERR 15–14	0	**Number of Errors** This field determines the number of sync errors that should occur before the sync state machine returns to hunt state. This field is active only if the SSME bit is set.	00 1 sync error 01 2 sync errors. 10 3 sync errors 11 4 sync errors.
— 13–8	0	Reserved. Write to zero for future compatibility.	

Table 18-83. CPRInTMRC (Continued)

Name	Reset	Description	Setting
SDLY 7–0	0	**Sync Delay** Use the following guidelines: • When CPRInTMR[ESA] field is clear, then SDLY should be 0x0 • When CPRInTMR[ESA] field is set to 0x01, then the delay is negative and the maximum delay is 255 Framer Clock cycles. • When CPRInTMR[ESA] field is set to 0x02, then the delay is positive and the maximum delay is 16 Framer Clock cycles. For details and timing diagrams, refer to **Section 18.3.6**, *Timers,* on page 18–29. The value can range from 0x00 to 0xFF (No sync delay to 255 cycles of sync delay).	

18.4.2.31 Receive Frame Pulse Width (CPRInRFPW)

address 0x4cc (CPRInRFPW)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								RNBFPW							
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	RHFPW								RBFPW							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-107. CPRInRFPW

This register determines the pulse width of the timing signals that output toward the device timers. For MSC8157E all the fields should be set to 0x8.

Table 18-84. CPRInRFPW

Name	Reset	Description	Setting
31–24	0	**Reserved. Write to zero for future compatibility.**	
RNBFPW 23–16	0	**Receive NodeB Frame Pulse Width** Determines the receive NodeB frame pulse width that is outputted toward the device timers.	0x0 No receive NodeB frame pulse 0x1 The receive Node B frame pulse width is 1 cycle of CPRI System Clock 0x2 The receive Node B frame pulse width is 2 cycles of CPRI System Clock. 0xFF The receive Node B frame pulse width is 255 cycles of CPRI System Clock.
RHFPW 15–8	0	**Receive Hyper Frame Pulse Width** Determines the Hyper frame pulse width that is outputted toward the device timers.	0x0 No receive hyper frame pulse 0x1 The receive hyper frame pulse width is 1 cycle of CPRI System Clock 0x2 The receive hyper frame pulse width is 2 cycles of CPRI System Clock. 0xff The receive hyper frame pulse width is 255 cycles of CPRI System Clock.
RBFPW 7–0	0	**Receive Basic Frame Pulse Width** Determines the receive basic frame pulse width that is outputted toward the device timers. Please note the Maximum frame pulse width for this field is 64 cycles.	0x0 No receive basic frame pulse 0x1 The receive basic frame pulse width is 1 cycle of CPRI System Clock 0x2 The receive basic frame pulse width is 2 cycles of CPRI System Clock. 0x40 The receive basic frame pulse width is 64 cycles of CPRI System Clock.

18.4.2.32 Transmit Frame Pulse Width (CPRInTFPW)

address 0x4d0 (CPRInTFPW)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								TNBFPW							
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	THFPW								TBFPW							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-108. CPRInTFPW

This register determines the pulse width of the timing signals that output toward the device timers. For MSC8157E all the fields should be set to 0x8.

Table 18-85. CPRInTFPW

Name	Reset	Description	Setting	
31–24	0	**Reserved. Write to zero for future compatibility.**		
TNBFPW 23–16	0	**Transmit NodeB Frame Pulse Width** Determines the transmit NodeB frame pulse width that is outputted toward the device timers.	0x0 0x1 0x2 0xFF	No transmit NodeB frame pulse The transmit Node B frame pulse width is 1 cycle of CPRI System Clock The transmit Node B frame pulse width is 2 cycles of CPRI System Clock. The transmit Node B frame pulse width is 255 cycles of CPRI System Clock.
THFPW 15–8	0	**Transmit Hyper Frame Pulse Width** Determines the Hyper frame pulse width that is outputted toward the device timers.	0x0 0x1 0x2 0xFF	No transmit hyper frame pulse The transmit hyper frame pulse width is 1 cycle of CPRI System Clock The transmit hyper frame pulse width is 2 cycles of CPRI System Clock. The transmit hyper frame pulse width is 255 cycles of CPRI System Clock.
TBFPW 7–0	0	**Transmit Basic Frame Pulse Width** Determines the transmit basic frame pulse width that is outputted toward the device timers. Please note the Maximum frame pulse width for this field is 64 cycles.	0x0 0x1 0x2 0x40	No transmit basic frame pulse The transmit basic frame pulse width is 1 cycle of CPRI System Clock The transmit basic frame pulse width is 2 cycles of CPRI System Clock. The transmit basic frame pulse width is 64 cycles of CPRI System Clock.

18.4.3 Control Registers

18.4.3.1 Receive Control Register (CPRInRCR)

address 0x500 (CPRInRCR)

	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	RIQSYNC	—											RVSSE	RHDLCE	RETHE	RIQE
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-109. CPRInRCR

CPRInRCR controls the activation/deactivation of the receive IQ, receive HDLC, receive Ethernet and Receive VSS. It also controls if the write access of the different CPRI modules to the system memory should be synchronized. After setting the RIQSYNC bit in the CPRI modules we want to synchronize and after all the relevant lanes are synchronized we set bit CPRIGRSR.GRIQS that will enable the write access of the different CPRI modules at the same time.

This register can be changed during normal operation. This register is reset due to auto negotiation reset.

Table 18-86. CPRInRCR Bit Descriptions

Name	Reset	Description	Setting
— 31–16	0	Reserved. Write to zero for future compatibility.	
RIQSYNC 15	0	**Receive IQ Synchronized** The bit determines if the received IQ data should be synchronized to the other CPRI. Setting this bit is the last step in the initialization.	0 The receive IQ DMA operation should not be synchronized to other CPRI modules. 1 The receive IQ DMA operation should be synchronized to other CPRI modules.
— 14–4	0	Reserved. Write to zero for future compatibility.	
RVSSE 3	0	**Receive VSS Enable** This bit determines if the VSS data is transferred to the system memory or not. Setting this bit is the last step in the initialization.	0 The received VSS data is not transferred to the VSS buffer in the system memory. 1 The received VSS data is transferred to the VSS buffer in the system memory.
RHDLCE 2	0	**Receive HDLC Enable** This bit determines if the HDLC data is transferred to the system memory or not. Setting this bit is the last step in the initialization.	0 The received HDLC data is not transferred to the HDLC buffers in the system memory. 1 The received HDLC data is transferred to the HDLC buffers in the system memory.
RETHE 1	0	**Receive Ethernet Enable** This bit determines if the Ethernet data is transferred to the system memory or not. Setting this bit is the last step in the initialization	0 The received Ethernet data is not transferred to the Ethernet buffers in the system memory. 1 The received Ethernet data is transferred to the Ethernet Buffers in the system memory.

Table 18-86. CPRInRCR Bit Descriptions

Name	Reset	Description	Setting
RIQE 0	0	**Receive IQ Enable** This bit determines if the IQ data is transferred to the system memory or not. Setting this bit is the last step in the initialization	0 The received IQ data is not transferred to the AxC buffers in the system memory. 1 The received IQ data is transferred to the AxC Buffers in the system memory.

18.4.3.2 Transmit Control Register (CPRInTCR)

address 0x504 (CPRInTCR)

	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	TIQSYNC						—						TVSSE	THDLCE	TETHE	TIQE
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-110. CPRInTCR

CPRInTCR controls the activation/deactivation of the transmit IQ, transmit HDLC, transmit Ethernet and Transmit VSS. It also controls if the transmit of the different CPRI modules should be synchronized. After setting the TIQSYNC bit in the CPRI modules we want to synchronize and after their DMA memories are full we set bit CPRIGTSR.GTIQS that will start the transmit of the different CPRI modules at the same time. The CPRIGTSR.GTIQS bit can be asserted after a time of a hyperframe after the TIQE bit is set as the DMA memories can be considered full after this time.

This register can be changed during normal operation. If the Ethernet transmit is stopped after it is enabled, set the ETH_RX_CONTROL[RX_DISCARD] bit before re-enabling the Ethernet transmit. This register is reset due to auto negotiation reset.

Table 18-87. CPRInTCR Bit Descriptions

Name	Reset	Description	Setting
— 31–16	0	Reserved. Write to zero for future compatibility.	
TIQSYNC **15**	0	**Transmit IQ Synchronization** The bit determines if the transfer of transmit IQ data should be synchronized to the other CPRI modules.	0 The transmit IQ data should not be synchronized to other CPRI modules 1 The transmit IQ data should be synchronized to other CPRI modules.
— 14–4	0	Reserved. Write to zero for future compatibility.	
TVSSE 3	0	**Transmit VSS Enable** This bit determines if the VSS data is transferred from the system memory buffer or not. Setting this bit is the last step in the initialization.	0 The transmit VSS data is not transferred from the VSS buffer in the system memory. 1 The transmit VSS data is transferred from the VSS buffer in the system memory.

Table 18-87. CPRInTCR Bit Descriptions

Name	Reset	Description	Setting
THDLCE 2	0	**Transmit HDLC Enable** This bit determines if the HDLC data is transferred from the system memory buffers or not. Setting this bit is the last step in the initialization.	0 The transmit HDLC data is not transferred from the HDLC buffers in the system memory. 1 The transmit HDLC data is transferred from the HDLC buffers in the system memory.
TETHE 1	0	**Transmit Ethernet Enable** This bit determines if the Ethernet data is transferred from the Ethernet buffers in the system memory or not. Setting this bit is the last step in the initialization	0 The transmit Ethernet data is not transferred from the Ethernet buffers in the system memory. 1 The transmit Ethernet data is transferred from the Ethernet Buffers in the system memory.
TIQE 0	0	**Transmit IQ Enable** This bit determines if the IQ data is transferred from the AxC buffers in the system memory or not. Setting this bit is the last step in the initialization	0 The transmit IQ data is not transferred from AxC buffers in the system memory. 1 The transmit IQ data is transferred from the AxC Buffers in the system memory.

18.4.3.3 Receive AxC Control Register (CPRInRACCR)

address 0x508 (CPRInRACCR)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								RAC23ACT	RAC22ACT	RAC21ACT	RAC20ACT	RAC19ACT	RAC18ACT	RAC17ACT	RAC16ACT
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	RAC15ACT	RAC14ACT	RAC13ACT	RAC12ACT	RAC11ACT	RAC10ACT	RAC9ACT	RAC8ACT	RAC7ACT	RAC6ACT	RAC5ACT	RAC4ACT	RAC3ACT	RAC2ACT	RAC1ACT	RAC0ACT
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-111. CPRInRACCR

This register determines the active receive antenna carriers. It is used only if the Receive Control Register (CPRInRCR)[RIQE] bit is set.

Table 18-88. CPRInRACCR Bit Descriptions

Name	Reset	Description	Setting
— 31–24	0	Reserved. Write to zero for future compatibility.	
RAC23ACT 23	0	**Receive AxC 23 is Active**	0 Receive AxC23 is disabled 1 Receive AxC23 is active
RAC22ACT 22	0	**Receive AxC 22 is Active**	0 Receive AxC22 is disabled 1 Receive AxC22 is active
RAC21ACT 21	0	**Receive AxC 21 is Active**	0 Receive AxC21 is disabled 1 Receive AxC21 is active
RAC20ACT 20	0	**Receive AxC 20 is Active**	0 Receive AxC20 is disabled 1 Receive AxC20 is active
RAC19ACT 19	0	**Receive AxC 19 is Active**	0 Receive AxC19 is disabled 1 Receive AxC19 is active
RAC18ACT 18	0	**Receive AxC 18 is Active**	0 Receive AxC18 is disabled 1 Receive AxC18 is active

Table 18-88. CPRInRACCR Bit Descriptions

Name	Reset	Description	Setting
RAC17ACT 17	0	**Receive AxC 17 is Active**	0 Receive AxC17 is disabled 1 Receive AxC17 is active
RAC16ACT 16	0	**Receive AxC 16 is Active**	0 Receive AxC16 is disabled 1 Receive AxC16 is active
RAC15ACT 15	0	**Receive AxC 15 is Active**	0 Receive AxC15 is disabled 1 Receive AxC15 is active
RAC14ACT 14	0	**Receive AxC 14 is Active**	0 Receive AxC14 is disabled 1 Receive AxC14 is active
RAC13ACT 13	0	**Receive AxC 13 is Active**	0 Receive AxC13 is disabled 1 Receive AxC13 is active
RAC12ACT 12	0	**Receive AxC 12is Active**	0 Receive AxC12 is disabled 1 Receive AxC12 is active
RAC11ACT 11	0	**Receive AxC 11 is Active**	0 Receive AxC11 is disabled 1 Receive AxC11 is active
RAC10ACT 10	0	**Receive AxC 10 is Active**	0 Receive AxC10 is disabled 1 Receive AxC10 is active
RAC9ACT 9	0	**Receive AxC 9 is Active**	0 Receive AxC9 is disabled 1 Receive AxC9 is active
RAC8ACT 8	8	**Receive AxC 8 is Active**	0 Receive AxC8 is disabled 1 Receive AxC8 is active
RAC7ACT 7	0	**Receive AxC 7 is Active**	0 Receive AxC7 is disabled 1 Receive AxC7 is active
RAC6ACT 6	0	**Receive AxC 6 is Active**	0 Receive AxC6 is disabled 1 Receive AxC6 is active
RAC5ACT 5	0	**Receive AxC 5 is Active**	0 Receive AxC5 is disabled 1 Receive AxC5 is active
RAC4ACT 4	0	**Receive AxC 4is Active**	0 Receive AxC4 is disabled 1 Receive AxC4 is active
RAC3ACT 3	0	**Receive AxC 3 is Active**	0 Receive AxC3 is disabled 1 Receive AxC3 is active
RAC2ACT 2	0	**Receive AxC 2 is Active**	0 Receive AxC2 is disabled 1 Receive AxC2 is active
RAC1ACT 1	0	**Receive AxC 1 is Active**	0 Receive AxC1 is disabled 1 Receive AxC1 is active
RAC0ACT 0	0	**Receive AxC 0 is Active**	0 Receive AxC0 is disabled 1 Receive AxC0 is active

18.4.3.4 Transmit AxC Control Register (CPRInTACCR)

address 0x510 (CPRInTACCR)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								TAC23ACT	TAC22ACT	TAC21ACT	TAC20ACT	TAC19ACT	TAC18ACT	TAC17ACT	TAC16ACT
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	TAC15ACT	TAC14ACT	TAC13ACT	TAC12ACT	TAC11ACT	TAC10ACT	TAC9ACT	TAC8ACT	TAC7ACT	TAC6ACT	TAC5ACT	TAC4ACT	TAC3ACT	TAC2ACT	TAC1ACT	TAC0ACT
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-112. CPRInTACCR

This register determines the transmit active antenna carriers.

Table 18-89. CPRInTACCR Bit Descriptions

Name	Reset	Description	Setting
— 31–24	0	Reserved. Write to zero for future compatibility.	
TAC23ACT 23	0	**Transmit AxC 23 Active**	0 Transmit AxC23 is disabled 1 Transmit AxC23 is active
TAC22ACT 22	0	**Transmit AxC 22 Active**	0 Transmit AxC22 is disabled 1 Transmit AxC22 is active
TAC21ACT 21	0	**Transmit AxC 21Active**	0 Transmit AxC21 is disabled 1 Transmit AxC21 is active
TAC20ACT 20	0	**Transmit AxC 20 Active**	0 Transmit AxC20 is disabled 1 Transmit AxC20 is active
TAC19ACT 19	0	**Transmit AxC 19 Active**	0 Transmit AxC19 is disabled 1 Transmit AxC19 is active
TAC18ACT 18	0	**Transmit AxC 18 Active**	0 Transmit AxC18 is disabled 1 Transmit AxC18 is active
TAC17ACT 17	0	**Transmit AxC 17 Active**	0 Transmit AxC17 is disabled 1 Transmit AxC17 is active
TAC16ACT 16	0	**Transmit AxC 16 Active**	0 Transmit AxC16 is disabled 1 Transmit AxC16 is active
TAC15ACT 15	0	**Transmit AxC 15 Active**	0 Transmit AxC15 is disabled 1 Transmit AxC15 is active
TAC14ACT 14	0	**Transmit AxC 14 Active**	0 Transmit AxC14 is disabled 1 Transmit AxC14 is active
TAC13ACT 13	0	**Transmit AxC 13 Active**	0 Transmit AxC13 is disabled 1 Transmit AxC13 is active
TAC12ACT 12	0	**Transmit AxC 12 Active**	0 Transmit AxC12 is disabled 1 Transmit AxC12 is active
TAC11ACT 11	0	**Transmit AxC 11 Active**	0 Transmit AxC11 is disabled 1 Transmit AxC11 is active
TAC10ACT 10	0	**Transmit AxC 10 Active**	0 Transmit AxC10 is disabled 1 Transmit AxC10 is active
TAC9ACT 9	0	**Transmit AxC 9 Active**	0 Transmit AxC9 is disabled 1 Transmit AxC9 is active

Table 18-89. CPRInTACCR Bit Descriptions (Continued)

Name	Reset	Description	Setting
TAC8ACT 8	8	**Transmit AxC 8 Active**	0 Transmit AxC8 is disabled 1 Transmit AxC8 is active
TAC7ACT 7	0	**Transmit AxC 7 Active**	0 Transmit AxC7 is disabled 1 Transmit AxC7 is active
TAC6ACT 6	0	**Transmit AxC 6 Active**	0 Transmit AxC6 is disabled 1 Transmit AxC6 is active
TAC5ACT 5	0	**Transmit AxC 5 Active**	0 Transmit AxC5 is disabled 1 Transmit AxC5 is active
TAC4ACT 4	0	**Transmit AxC 4 Active**	0 Transmit AxC4 is disabled 1 Transmit AxC4 is active
TAC3ACT 3	0	**Transmit AxC 3 Active**	0 Transmit AxC3 is disabled 1 Transmit AxC3 is active
TAC2ACT 2	0	**Transmit AxC 2 Active**	0 Transmit AxC2 is disabled 1 Transmit AxC2 is active
TAC1ACT 1	0	**Transmit AxC 1 Active**	0 Transmit AxC1 is disabled 1 Transmit AxC1 is active
TAC0ACT 0	0	**Transmit AxC 0 Active**	0 Transmit AxC0 is disabled 1 Transmit AxC0 is active

18.4.3.5 Receive Control Attribute Register (CPRInRCA)

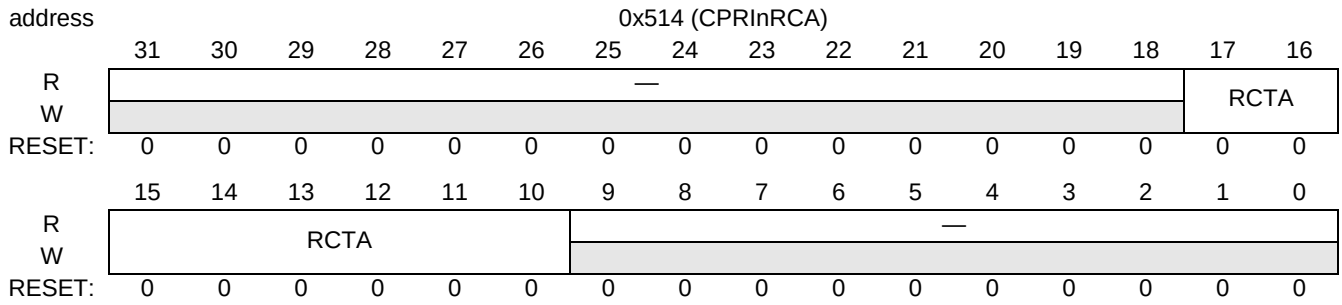


Figure 18-113. CPRInRCA

This register is used to read from the receive control table.

This register is reset due to auto negotiation reset.

Table 18-90. CPRInRCA Bit Descriptions

Name	Reset	Description	Setting
— 31–18	0	Reserved. Write to zero for future compatibility.	
RCTA 17–10	0	**Receive Control Table Address** Determines the number (address) of the control word.	0x00 receive control table address 0 0x01 receive control table address 1 : 0xFF receive control table address is 255
— 9–0	0	Reserved. Write to zero for future compatibility.	

18.4.3.6 Receive Control Data register 0 (CPRInRCD0)

address 0x518 (CPRInRCD0)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RCD0															
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	RCD0															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-114. CPRInRCD0

When a read operation is performed, this register contains bytes 0–3 read from address CPRInRCA[RCTA] of the receive control table.

This register is reset due to auto negotiation reset.

Table 18-91. CPRInRCD0 Bit Descriptions

Name	Reset	Description	Setting
RCD0 31–0	0	**Receive Control Data 0.** Please note: • byte 0 = bits 31–24 • byte 1 = bits 23–16 • byte 2 = bits 15–8 • byte 3 = bits 7–0	

18.4.3.7 Receive Control Data Register 1 (CPRInRCD1)

address 0x51C (CPRInRCD1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RCD1															
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	RCD1															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-115. CPRInRCD1

When a read operation is performed, this register contains bytes 4–7 read from address CPRInRCA[RCTA] of the receive control table.

This register is reset due to auto negotiation reset.

Table 18-92. CPRInRCD1 Bit Descriptions

Name	Reset	Description	Setting
RCD1 31–0	0	**Receive Control Data 1.** Please note: • byte 4 = bits 31–24 • byte 5 = bits 23–16 • byte 6 = bits 15–8 • byte 7 = bits 7–0	

18.4.3.8 Receive Control Data Register 2 (CPRInRCD2)

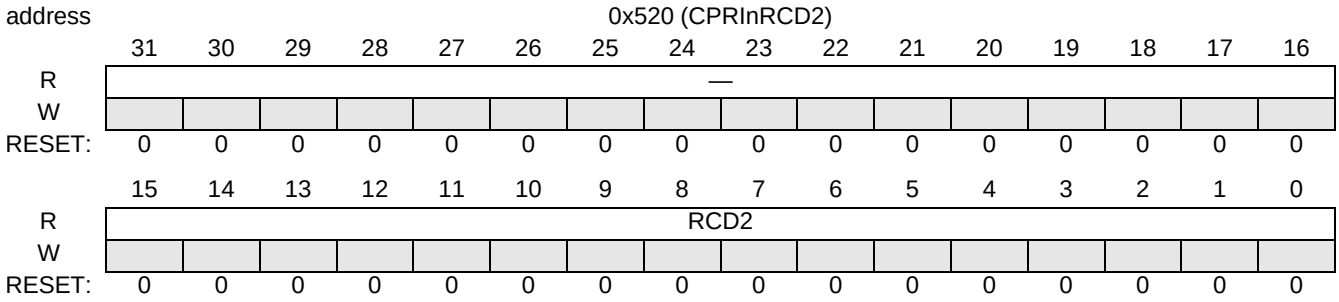


Figure 18-116. CPRInRCD2

When a read operation is performed, this register contains bytes 8–9 read from address CPRInRCA[RCTA] of the receive control table.

This register is reset due to auto negotiation reset.

Table 18-93. CPRInRCD2 Bit Descriptions

Name	Reset	Description	Setting
— 31–16	0	Reserved. Write to zero for future compatibility.	
RCD2 15–0	0	**Receive Control Data 2.** Please note: • byte 8 = bits 15–8 • byte 9 = bits 7–0	

18.4.3.9 Transmit Control Attribute Register (CPRInTCA)

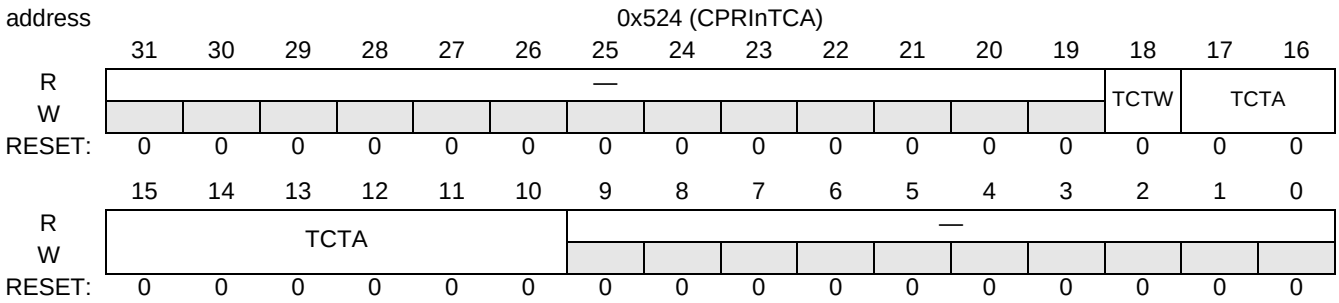


Figure 18-117. CPRInTCA

This register is used to read and write from the transmit control table. The access attributes (read/write, address, byte enable) are determined by this register. The read address can be used for debug purposes. This register is reset due to auto negotiation reset.

Table 18-94. CPRInTCA Bit Descriptions

Name	Reset	Description	Setting
— 31–19	0	Reserved. Write to zero for future compatibility.	
TCWT 18	0	**Transmit Control Table Write** Determines whether the transmit control table operation is write operation.If the bit is set then the operation is write.	0x0 read operation 0x1 write operation
TCTA 17–10	0	**Transmit Control Table Address** Determines the number (address) of the control word.	0x0 transmit control table address 0 0x1 transmit control table address 1 : 0xFF transmit control table address is 255
— 9–0	0	Reserved. Write to zero for future compatibility.	

18.4.3.10 Transmit Control Data Register 0 (CPRInTCD0)

address 0x528 (CPRInTCD0)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TCD0															
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	TCD0															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-118. CPRInTCD0

When a write operation is performed, this register contains the first four bytes of data to insert into transmit control table address CPRInTCA[TCTA]. When a read operation is performed, this register contains bytes 0–3 read from address CPRInTCA[TCTA] of the receive control table.

This register is reset due to auto negotiation reset.

Table 18-95. CPRInTCD0 Bit Descriptions

Name	Reset	Description	Setting
TCD0 31–0	0	**Transmit Control Data 0.** Please note: • byte 0 = bits 31–24 • byte 1 = bits 23–16 • byte 2 = bits 15–8 • byte 3 = bits 7–0	0x0–0xFFFFFFFF

18.4.3.11 Transmit Control Data register 1(CPRInTCD1)

address 0x52C (CPRInTCD1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TCD1															
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	TCD1															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-119. CPRInTCD1

When write operation is performed this register contain bytes 4–7 of data that should be insert to transmit control table at address CPRInTCA[TCTA]. When read operation is performed this register contain bytes 4–7 that was read from address CPRInTCA[TCTA] of the transmit control table.

This register is reset due to auto negotiation reset.

Table 18-96. CPRInTCD1 Bit Descriptions

Name	Reset	Description	Setting
TCD1 31–0	0	**Transmit Control Data 1.** Please note: • byte 4= bits 31–24 • byte 5= bits 23–16 • byte 6= bits 15–8 • byte 7= bits 7–0	0x0–0xFFFFFFFF

18.4.3.12 Transmit Control Data Register 2 (CPRInTCD2)

address 0x530 (CPRInTCD2)

	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 W	TCD2															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-120. CPRInTCD2

When write operation is performed this register contain bytes 8–9 of data that should be insert to transmit control table at address CPRInTCA[TCTA]. When read operation is performed this register contain bytes 8–9 that was read from address CPRInTCA[TCTA] of the transmit control table.

This register is reset due to auto negotiation reset.

Table 18-97. CPRInTCD2 Bit Descriptions

Name	Reset	Description	Setting
— 31–16	0	Reserved. Write to zero for future compatibility.	
TCD2 15–0	0	**Transmit Control Data 2.** Please note: • byte 8 = bits 15–8 • byte 9 = bits 7–0	0x0–0xFFFF

18.4.3.13 Receive IQ First Threshold (CPRInRIQFT)

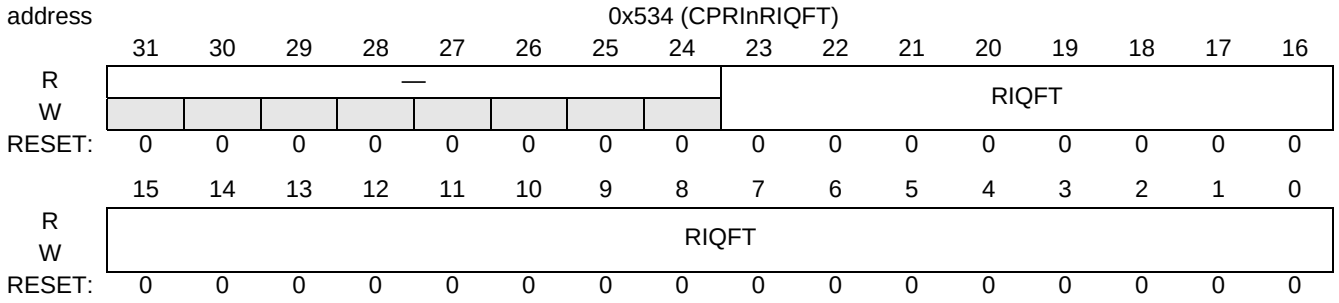


Figure 18-121. CPRInRIQFT

CPRInRIQFT determines the first threshold of the receive IQ buffers. When the receive buffers are filled up to the first threshold defined by this register, that is, the receive IQ buffer displacement register (CPRInRIQBDR) = (CPRInRIQFT) + (Receive IQ MBus Transaction Generation Size), the RIQFTE bit in the CPRInRER is set. If the associated enable bit in register CPRIICRy is also set, an interrupt is generated.

This register is reset due to auto negotiation reset.

Table 18-98. CPRInRIQFT Bit Descriptions

Name	Reset	Description	Setting
— 31–24	0	Reserved. Write to zero for future compatibility.	
RIQFT 23–0	0	**Receive IQ Data Buffer First Threshold** Determines the location of the first threshold in the AxC receive data buffers. The pointer should be integer multiple of the Receive IQ MBus Transaction Generation Size defined in register RIQMTS	0x000000 to (CPRInRIQBS–64)

18.4.3.14 Receive IQ Second Threshold (CPRInRIQST)

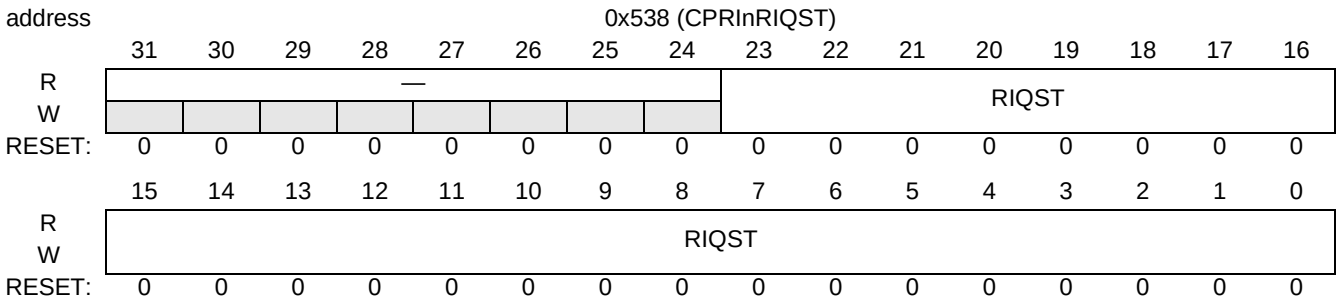


Figure 18-122. CPRInRIQST

CPRInRIQST determines the second threshold of the receive IQ buffers. When the receive buffers are filled up to the second threshold defined by this field, that is, the receive IQ buffer displacement register (CPRInRIQBDR) = (CPRInRIQST) + (Receive IQ MBus Transaction Generation Size), the RIQSTE bit in the CPRInRER is set. If the associated enable bit in register CPRIICRy is also set, an interrupt is generated.

This register is reset due to auto negotiation reset.

Table 18-99. CPRInRIQST Bit Descriptions

Name	Reset	Description	Setting
— 31–24	0	Reserved. Write to zero for future compatibility.	
RIQST 23–0	0	**Receive IQ Data Buffer Second Threshold** Determines the location of the second threshold in the AxC receive data buffers. The pointer should be integer multiple of the Receive IQ MBus Transaction Generation Size defined in register RIQMTS	0x000000 to (RIQBS–64)

18.4.3.15 Receive IQ Threshold (CPRInRIQT)

address 0x53c (CPRInRIQT)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RIQT															
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	RIQT															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-123. CPRInRIQT

A receive IQ threshold event is issued each time the number of RIQT bytes are transferred to IQ buffers in the system memory and the RIQTE bit in the CPRInRER register is set. If the associated enable bit in register CPRIICRy is also set, an interrupt is generated:

Table 18-100. CPRInRIQT Bit Descriptions

Name	Reset	Description	Setting
RIQT 31–0	0	Receive IQ Threshold. Note: RIQT must be a multiple of RIQMTS[RIQMTS].	0x0 to 0xFFFF_FFC0 Note: The value 0 means that this event is not active.

18.4.3.16 Transmit IQ First Threshold (CPRInTIQFT)

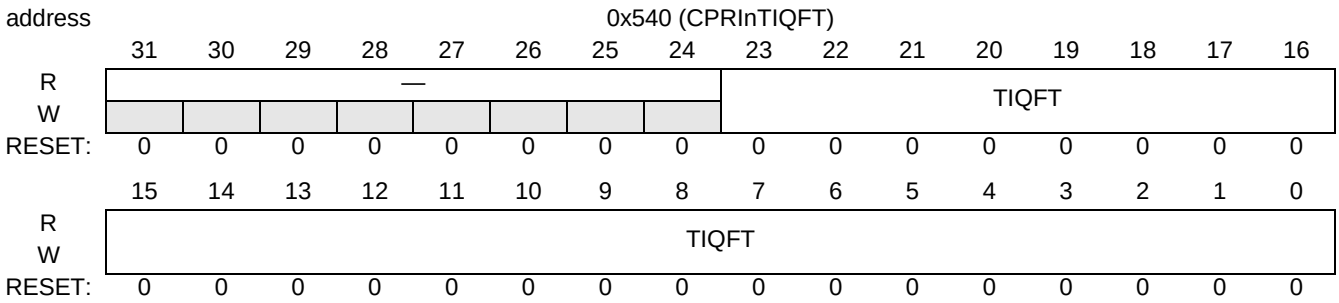


Figure 18-124. CPRInTIQFT

CPRInTIQFT determines the first threshold of the transmit IQ buffers. When the transmit buffers are emptied up to the first threshold defined by this field, that is, CPRIn Transmit IQ Buffer displacement register (CPRInTIQBDR) = (CPRInTIQFT) + Transmit IQ MBus Transaction Generation Size, the TIQFTE bit in the CPRInTER is set. If the associated enable bit in register CPRIICRy is also set, an interrupt is generated.

This register is reset due to auto negotiation reset.

Table 18-101. CPRInTIQFT Bit Descriptions

Name	Reset	Description	Setting
— 31–24	0	Reserved. Write to zero for future compatibility.	
TIQFT 23–0	0	**Transmit IQ Data Buffer First Threshold** Determines the location of the first threshold in the AxC transmit data buffers. The pointer should be integer multiple of the Transmit IQ MBus Transaction Generation Size defined in register TIQMTS	0x000000 to (TIQBS–64)

18.4.3.17 Transmit IQ Second Threshold (CPRInTIQST)

address 0x544 (CPRInTIQST)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								TIQST							
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	TIQST															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-125. CPRInTIQST

CPRInTIQST determines the second threshold of the transmit IQ buffers. When the transmit buffers are emptied up to the second threshold defined by this field, that is, the CPRIn Transmit IQ Buffer displacement register (CPRInTIQBDR) = (CPRInTIQST) + Transmit IQ MBus Transaction Generation Size, the TIQSTE bit in the CPRInTER is set. If the associated enable bit in register CPRIICRy is also set, an interrupt is generated. This register is reset due to auto negotiation reset.

Table 18-102. CPRInTIQST Bit Descriptions

Name	Reset	Description	Setting
— 31–24	0	Reserved. Write to zero for future compatibility.	
TIQST 23–0	0	**Transmit IQ Data Buffer Second Threshold** Determines the location of the second threshold in the AxC transmit data buffers. The pointer should be integer multiple of the Transmit IQ MBus Transaction Generation Size defined in register TIQMTS	0x000000 to (TIQBS–64)

18.4.3.18 Transmit IQ Threshold (CPRInTIQT)

address 0x548 (CPRInTIQT)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TIQT															
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	TIQT															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-126. CPRInTIQT

A transmit IQ threshold event is issued each time that number of TIQT bytes were transferred from the IQ buffers in the system memory and the TIQTE bit in the CPRInTER register is set. If the associated enable bit in register CPRIICRy is also set, an interrupt is generated.

Table 18-103. CPRInTIQT Bit Descriptions

Name	Reset	Description	Setting
TIQT 31–0	0	**Transmit IQ Threshold.** Note: TIQT must be multiple of TIQMTS[TIQMTS].	0x000000 to 0xffff_ffC0 Note: The value 0 determines that this event is not active.

18.4.3.19 Receive VSS Threshold (CPRInRVSST)

address 0x54c (CPRInRVSST)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RVSST															
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	RVSST															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-127. CPRInRVSST

A receive VSS event is issued each time the number of VSS bytes transferred to the VSS buffer in the system memory exceeds the RVSST value and the RVSSTE bit in the CPRInRER is set. If the associated enable bit in register CPRIICRy is also set, an interrupt is generated.

When the event is generated the Receive VSS Buffer Displacement Register (CPRInRVSSBDR) = (CPRInRVSST) *N, where N is an integer number.

Table 18-104. CPRInRVSST Bit Descriptions

Name	Reset	Description	Setting
RVSST 31–0	0	**Receive VSS Threshold** Determines the receive VSS threshold level. This field should be an integer multiple of the Receive VSS MBus Transaction Generation Size)	0x000000 to 0xFFFFFFF0 Note: The value 0 determines that this event is not active.

18.4.3.20 Transmit VSS Threshold (CPRInTVSST)

address 0x550 (CPRInTVSST)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TVSST															
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	TVSST															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-128. CPRInTVSST

A transmit VSS event is issued each time that the number of VSS bytes transferred from the VSS buffer in the system memory exceeds the TVSST value and the TVSSTE bit in the CPRInTER is set. If the associated enable bit in register CPRIICRy is also set an interrupt is generated.

When the event is generated the Transmit VSS Buffer Displacement Register (CPRInTVSSBDR) = (CPRInTVSST) *N, where N is an integer number.

Table 18-105. CPRInTVSST Bit Descriptions

Name	Reset	Description	Setting
TVSST 31–0	0	**Transmit VSS Threshold** Determines the transmit VSS threshold level. This field should be integer multiple of the Transmit VSS MBus Transaction Generation Size)	0x000000 to 0xFFFFFFF0 Note: The value 0 determines that this event is not active.

18.4.3.21 Receive Ethernet Coalescing Threshold (CPRInRETHCT)

address 0x554 (CPRInRETHCT)

	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	—								RETHCT							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-129. CPRInRETHCT

The CPRInRETHCT register sets the receive Ethernet packet interrupt threshold defined in number of packets received. The CPRInRER[RETHE] bit is set if the receive coalescing counter (indicated by the CPRInRETHCS register) is equal to the receive Ethernet coalescing threshold (RETHCT) value. If the associated enable bit in register CPRIICRy is also set, an interrupt is generated.

Note: Occasionally, the number of received packets reported by the CPRI does not correspond to the number of updated BDs. If the core detects this condition when reading the BDs, program the core to request a retransmission of the missing packets.

Table 18-106. CPRInRETHCT Bit Descriptions

Name	Reset	Description	Setting
— 31–8	0	Reserved. Write to zero for future compatibility.	
RETHCT 7–0	0	**Receive Ethernet coalescing Threshold** Receive Ethernet event is issued after (RETHCT + 1) packets	0x00 Receive Ethernet event is issued after 1 packet 0x01 Receive Ethernet event is issued after 2 packets ... 0xFE Receive Ethernet event is issued after 255 packets. 0xFF Receive Ethernet event is issued after 256 packets

18.4.3.22 Transmit Ethernet Coalescing Threshold (CPRInTETHCT)

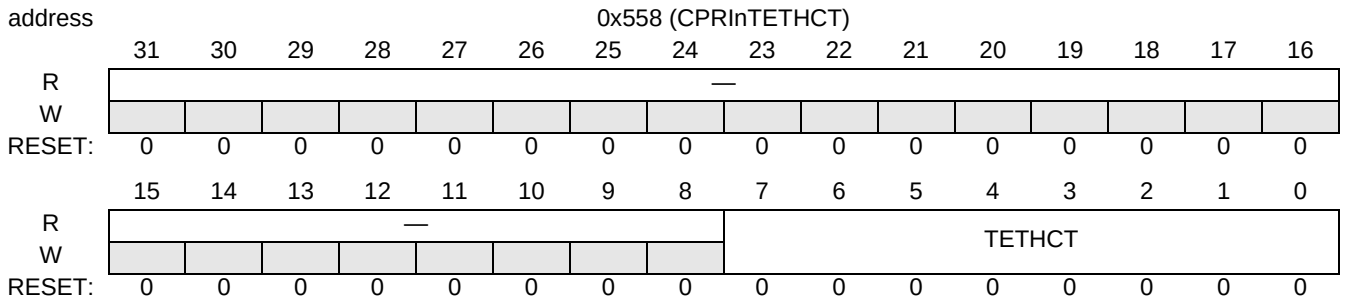


Figure 18-130. CPRInTETHC

The CPRInTETHCT register sets the transmit Ethernet packet interrupt threshold defined in number of packets transmitted. The CPRInTER[TETHE] bit is set if the transmit coalescing counter (indicated by the CPRInTETHCS register) is equal to transmit Ethernet coalescing threshold (TETHCT) value. If the associated enable bit in register CPRIICRy is also set, an interrupt is generated.

Table 18-107. CPRInTETHCT Bit Descriptions

Name	Reset	Description	Setting
— 31–8	0	Reserved. Write to zero for future compatibility.	
TETHCT 7–0	0	**Transmit Ethernet coalescing Threshold** Transmit Ethernet interrupt is issued after (TETHCT+1) packets	0x00 Transmit Ethernet event is issued after 1 packet 0x01 Transmit Ethernet event is issued after 2 packets ... 0xFE Transmit Ethernet event is issued after 255 packets 0xFF Transmit Ethernet event is issued after 256 packets

18.4.3.23 CPRI Receive Control & Timing Interrupts Enable Register (CPRInRCIER)

address 0x560 (CPRInRCIER)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TLEVEL	—													RBFE	RHFE
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	CLEVEL	—												RE-THEE	RHDLCEE	RVS-SEE
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-131. CPRInRCIER

The CPRInRCIER enables the receive control interrupt and the receive timing interrupts.

Table 18-108. CPRInRCIER Bit Descriptions

Name	Reset	Description	Setting
TLEVEL 31	0	**Timing Level Interrupt** This bit determines whether the receive timing interrupt is pulse or level	0 The timing interrupt is edge triggered 1 The timing interrupt is level triggered.
— 30–18	0	Reserved. Write to zero for future compatibility.	
RBFE 17	0	**Receive BFN Timing Event Enable**	0 Receive BFN timing interrupt is masked 1 receive BFN timing interrupt is enabled
RHFE 16	0	**Receive HFN Timing Event Enable**	0 Receive HFN timing interrupt is masked 1 Receive HFN timing interrupt is enabled
CLEVEL 15	0	**Control LeveL interrupt** This bit determines if the receive control interrupt is pulse or level	0 The control interrupt is edge triggered 1 The control interrupt is level triggered.
— 14–3	0	Reserved. Write to zero for future compatibility.	
RETHEE 2	0	**Receive Ethernet Event Enable** If this bit is set and the status bit CPRInRER[RETHE] is set, then CPRIn_RX_control interrupt is generated.	0 Receive Ethernet Interrupt is disabled 1 Receive Ethernet Interrupt is enabled
RHDLCEE 1	0	**Receive HDLC Event Enable** If this bit is set and the status bit CPRInRER[RHDLCE] is set, then CPRIn_RX_control interrupt is generated.	0 Receive HDLC Interrupt is disabled 1 Receive HDLC Interrupt is enabled
RVSSEE 0	0	**Receive VSS Event Enable** If this bit is set and the status bit CPRInRER[RVSSE] is set, then CPRIn_RX_control interrupt is generated.	0 Receive VSS Interrupt is disabled 1 Receive VSS Interrupt is enabled

18.4.3.24 CPRI Transmit Control & Timing Interrupts Enable Register (CPRInTCIER)

address 0x564 (CPRInTCIER)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TLEVEL	—													TBFE	THFE
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	CLEVEL	—												TETHEE	THDLCEE	TVS-SEE
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-132. CPRInTCIER

CPRInTCIER enables the transmit control interrupt and the transmit timing interrupt.

Table 18-109. CPRInTCIER Bit Descriptions

Name	Reset	Description	Setting
TLEVEL 31	0	**Timing LeveL Interrupt** This bit determines if the transmit timing interrupt is pulse or level	0 The timing interrupt is edge triggered 1 The timing interrupt is level triggered.
— 30–18	0	Reserved. Write to zero for future compatibility.	
TBFE 17	0	**Transmit BFN Timing Event**	0 Transmit BFN timing interrupt masked 1 Transmit BFN timing interrupt enabled
THFE 16	0	**Transmit Hyper Frame Timing Event Enable**	0 Transmit HFN timing interrupt masked 1 Transmit HFN timing interrupt enabled
CLEVEL 15	0	**Control Level Interrupt** This bit determines if the transmit control interrupt is pulse or level	0 The control interrupt is edge triggered 1 The control interrupt is level triggered.
— 14–3	0	Reserved. Write to zero for future compatibility.	
TETHEE 2	0	**Transmit Ethernet Event Enable** If this bit is set and the status bit CPRInTER[TETHE] is set, then CPRIn_tx_control interrupt is generated.	0 Transmit Ethernet interrupt masked 1 Transmit Ethernet interrupt enabled
THDLCEE 1	0	**Transmit HDLC Event Enable** If this bit is set and the status bit CPRInTER[THDLCE] is set, then CPRIn_Tx_control interrupt is generated.	0 Transmit HDLC interrupt masked 1 Transmit HDLC interrupt i enabled
TVSSEE 0	0	**Transmit VSS Event Enable** If this bit is set and the status bit CPRInTER[TVSSE] is set, then CPRIn_Tx_control interrupt is generated.	0 Transmit VSS interrupt masked 1 Transmit VSS interrupt enabled

18.4.3.25 Receive IQ Threshold Second Destination (CPRInRIQTSD)

address 0x568 (CPRInRIQTSD)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RIQTSD															
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	RIQTSD															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-133. CPRInRIQTSD

A second destination receive IQ threshold event is issued each time the number of RIQTSD bytes are transferred to IQ buffers in the second destination and the RIQTSDE bit in the CPRInRER is set. If the associated enable bit in register CPRIICRy is also set, an interrupt is generated:

Table 18-110. CPRInRIQT Bit Descriptions

Name	Reset	Description	Setting
RIQTSD 31–0	0	Receive IQ Threshold.Second Destination Note: RIQT must be multiple of RIQSDMTS[RIQSDMTS].	0x0 to 0xFFFF_FFC0 Note: The value 0 means that this event is not active.

18.4.3.26 CPRI Error Interrupt Enable Register (CPRInEIER)

address 0x570 (CPRInEIER)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								FAE	—	LLOFE	LLOSE	RSDIE	RRAIE	RLOFE	RLOSE
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	EXSSLOSE	—					TVSSUNE	RVSSOVE	RHBDUNE	THDLCOVE	RHDLCOVE	REBDUNE	TETHUNE	RETHOVE	TIQUNE	RIQOVE
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 18-134. CPRInEIER

CPRInEIER enables the error interrupt of CPRIn. If an error event bit in CPRInEER is set and the corresponding bit in this register is also set, then an error interrupt is generated.

Table 18-111. CPRInEIER Bit Description

Name	Reset	Description	Setting
— 31–24	0	Reserved. Write to zero for future compatibility.	
FAE 23	0	**Frequency Alarm Error Enable**	0 Frequency alarm error is masked 1 Frequency alarm error is enabled
22	0	Reserved. Write to zero for future compatibility.	
LLOFE 21	0	**Local Loss of Frame Error Enable**	0 Local LOF error is masked 1 Local LOF error is enabled

Table 18-111. CPRInEIER Bit Description (Continued)

Name	Reset	Description	Setting
LLOSE 20	0	**Local LOS Error Enable**	0 Local LOS error is masked 1 Local LOS error is enabled
RSDIE 19	0	**Remote SDI Error Enable**	0 Remote SDI error is masked 1 Remote SDI error is enabled
RRAIE 18	0	**Remote RAI Error Enable**	0 Remote RAI error is masked 1 Remote RAI error is enabled
RLOFE 17	0	**Remote LOF Error Enable**	0 Remote LOF error is masked 1 Remote LOF error is enabled
RLOSE 16	0	**Remote LOS Error Enable**	0 Remote LOS error is masked 1 Remote LOS error is enabled
EXSSLOSE 15	0	**External Sync Synchronization Lost Error Enable**	0 External sync synchronization error is masked 1 External sync synchronization signal is enabled.
— 14–10	0	Reserved. Write to zero for future compatibility.	
TVSSUNE 9	0	**Transmit VSS Underrun Error Enable**	0 Transmit VSS underrun error is masked 1 Transmit VSS underrun error is enabled
RVSSOVE 8	0	**Receive VSS Overrun Error Enable**	0 Receive VSS error overrun is masked 1 Receive VSS error underrun is enabled
RHBDUNE 7	0	**Receive HDLC Buffer Descriptor Underrun Error Enable**	0 Receive HDLC buffer descriptor underrun error is masked 1 Receive HDLC buffer descriptor underrun error is enabled
THDLCUNE 6	0	**Transmit HDLC Underrun Error Enable**	0 Transmit HDLC underrun error is masked 1 Transmit HDLC underrun error is enabled
RHDLCOVE 5	0	**Receive HDLC Overrun Error Enable**	0 Receive HDLC error overrun is masked 1 Receive HDLC error underrun is enabled
REBDUNE 4	0	**Receive ETH Buffer Descriptor Underrun Error Enable**	0 Receive ETH buffer descriptor underrun error is masked 1 Receive ETH buffer descriptor underrun error is enabled
TETHUNE 3	0	**Transmit ETH Underrun Error Enable**	0 Transmit ETH underrun error is masked 1 Transmit ETH underrun error is enabled
RETHOVE 2	0	**Receive ETH Memory Overrun Error Enable**	0 Receive ETH overrun error is masked 1 Receive ETH overrun error is enabled

Table 18-111. CPRInEIER Bit Description (Continued)

Name	Reset	Description	Setting
TIQUNE 1	0	**Transmit IQ Underrun Error Enable**	0 Transmit IQ underrun error is masked 1 Transmit IQ underrun error is enabled
RIQOVE 0	0	**Receive IQ Overrun Error Enable**	0 Receive IQ overrun error is masked 1 Receive IQ overrun error is enabled

18.4.3.27 Timer Enable Register (CPRInTMRE)

address 0x574 (CPRInTMRE)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TACT	—														
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	—															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-135. CPRInTMRE

CPRInTMRE register enables the timer operation. This register is reset due to auto negotiation reset.

Table 18-112. CPRInTMRE bit descriptions

Name	Reset	Description	Setting
TACT 31	0	**Timer Enable** This bit enable the timer	0 This timer is disabled. 1 This timer is enabled
— 30–0	0	Reserved. Write to zero for future compatibility.	

18.4.3.28 Receive Ethernet Write Pointer Ring (CPRInREWPR)

address 0x578 (CPRInREWPR)

	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	—							REW	REWPR							
W								W								
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-136. CPRInREWPR

The Receive Ethernet Write Pointer ring indicates the number of the next buffer descriptor that should be written by the core. This register is updated only by the core each time that a new receive buffer descriptor (or some buffer descriptors) is updated in the system memory. The core must ensure that a buffer descriptor overrun or underrun does not occur. Each time that the core wraps to the beginning of the buffer descriptor ring, the core should update the wrap bit. This register is reset due to an auto negotiation reset.

Table 18-113. CPRInREWPR Bit Descriptions

Name	Reset	Description	Setting
— 30–9	0	Reserved. Write to zero for future compatibility.	
REWW 8	0	**Receive Ethernet Write Pointer Wrap** This bit should be toggled every time that the core returns to the beginning of the Ethernet buffer descriptor ring.	0 Wrap did not occur 1 Wrap occurred
REWPR 7–0	0	**Receive Ethernet Write Pointer** Indicates the number of the next Ethernet buffer descriptor that should be written by the core	

18.4.3.29 Transmit Ethernet Write Pointer Ring (CPRInTEWPR)

address 0x57c (CPRInTEWPR)

	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	—							TEW	TEWPR							
W								W								
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-137. CPRInTEWPR

The transmit Ethernet write pointer ring indicates the next buffer descriptor that should be written by the core. This register is updated only by the core each time that a new transmit buffer descriptor (or number of descriptors) is updated in the system memory. The core must ensure that a buffer descriptor overrun or underrun does not occur. Each time that the core wraps to the beginning of the buffer descriptor ring, the core should update the wrap bit. This register is reset due to an auto negotiation reset.

Table 18-114. CPRInTEWPR Bit Descriptions

Name	Reset	Description	Setting
— 30–9	0	Reserved. Write to zero for future compatibility.	
TEWW 8	0	**Transmit Ethernet Write Pointer Wrap** This bit should be toggled every time that the core returns to the beginning of the Ethernet buffer descriptor ring.	0 Wrap did not occur 1 Wrap occurred
TEWPR 7–0	0	**Transmit Ethernet Write Pointer** Indicates the number of the next Ethernet buffer descriptor that should be written by the core	

18.4.3.30 Receive HDLC Write Pointer Ring (CPRInRHWPR)

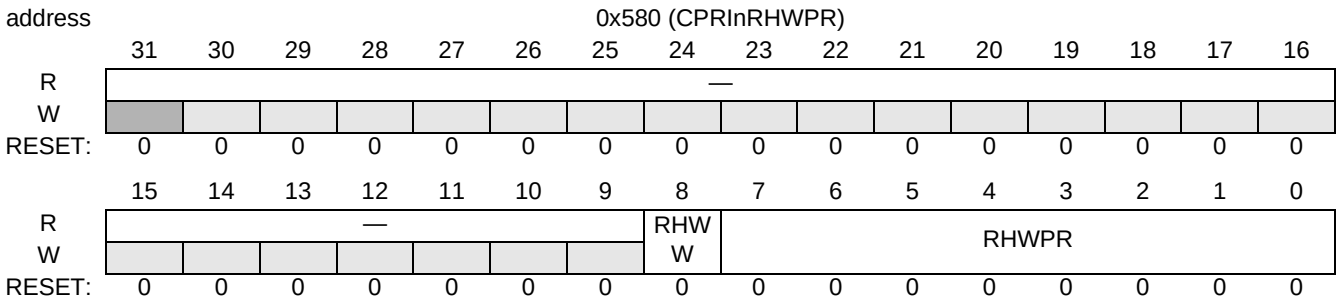


Figure 18-138. CPRInRHWPR

The receive HDLC write pointer ring indicates the next buffer descriptor that should be written by the core. This register is updated only by the core each time that a new receive buffer descriptor (or some buffer descriptors) is updated in the system memory. The core must ensure that a buffer descriptor overrun or underrun does not occur. Each time that the core wraps to the beginning of the buffer descriptor ring, the core should update the wrap bit.

This register is reset due to auto negotiation reset.

Table 18-115. CPRInRHWPR Bit Descriptions

Name	Reset	Description	Setting
— 31–9	0	Reserved. Write to zero for future compatibility.	
RHWW 8	0	**Receive HDLC Write Pointer Wrap** This bit should toggled every time that the core returns to the beginning of the buffer descriptor ring.	0 Wrap did not occur 1 Wrap occurred
RHWPR 7–0	0	**Receive HDLC Write Pointer** Indicates the number of the next HDLC buffer descriptor that should be written by the core	

18.4.3.31 Transmit HDLC Write Pointer Ring (CPRInTHWPR)

address 0x584 (CPRInTHWPR)

	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	—							THW	THWPR							
W								W								
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-139. CPRInTHWPR

The transmit HDLC write pointer ring indicates the next buffer descriptor that should be written by the core. This register is updated only by the core each time that a new transmit buffer descriptor (or some buffer descriptors) is updated in the system memory. The core must ensure that a buffer descriptor overrun or underrun does not occur. Each time that the core wraps to the beginning of the buffer descriptor ring, the core should update the wrap bit. This register is reset due to auto negotiation reset.

Table 18-116. CPRInTHWPR Bit Descriptions

Name	Reset	Description	Setting
— 30–9	0	Reserved. Write to zero for future compatibility.	
THWW 8	0	**Transmit HDLC Write Pointer Wrap** This bit should be toggled every time that the core returns to the beginning of the HDLC buffer descriptor ring.	0 Wrap not occur 1 Wrap occurred
THWPR 7–0	0	**Transmit HDLC Write Pointer** Indicates the number of the next HDLC buffer descriptor that should be written by the core	

18.4.3.32 Receive Antenna Carrier Parameter Register <y> (CPRInRACPR<y>)

address 0x5B0 (CPRInRACPRy), offset 4, range 0–23)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RACBA															
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	RACBA															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-140. CPRInRACPR<Y>

CPRInRACPR<y> determines the receive buffer location of antenna carrier y (y= 0 to 23). This register can only be changed when Receive AxC Control Register (CPRInRACCR)[RACyACT] bit is cleared.

Note: The CPRInRACPR<Y> register with an index number (Y) should be valid when setting the corresponding bit in the Receive AxC Control Register (CPRInRACCR)[RACyACT] register.

These registers are reset due to auto negotiation reset.

Table 18-117. CPRInRACCR Bit Descriptions

Name	Reset	Description	Setting
RACBA 31–0	0	**Receive Antenna Carrier Base Address** Determines the buffer base address of antenna carrier Y in the system memory. The RACBA value should be 16-byte aligned; that is, the four LSB should be 0.	0x00000000–0xFFFFFFF0

18.4.3.33 Transmit Antenna Carrier Parameter Register <y> (CPRInTACPR<y>)

address 0x640 (CPRInTACPRy), offset 4, range 0–23)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TACBA															
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	TACBA															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-141. CPRInTACPRy

TCPRInTACPR<y> determines the transmit buffer location of antenna carrier y (y= 0 to 23). This register can only be changed when Transmit AxC Control Register (CPRInTACCR)[TACyACT] bit is cleared.

Note: The CPRInTACPR<Y> register with an index number (Y) should be valid when setting the corresponding bit in the Transmit AxC Control Register (CPRInTACCR)[TACyACT] register.

These registers are reset due to auto negotiation reset.

Table 18-118. CPRInTACCR<Y> Bit Descriptions

Name	Reset	Description	Setting
TACBA 31–0	0	**Transmit Antenna Carrier Base Address** Determines the buffer base address of antenna carrier n in the system memory. The TACBA value should be 16-byte aligned; that is, the four LSB should be 0.	0x00000000–0xFFFFFFF0

18.4.3.34 CPRI Auxiliary Interface Mask Registers <y> (CPRInMASKR<y>)

address 0x700 (CPRInMASKRy, offset 4, range 0–39)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MASKR															
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	MASKR															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-142. CPRInMASKRy

CPRInMASKR registers determines the masking bits for auxiliary interface. Each bit refers to one bit in the frame to be transmitted. When a bit is set then the relevant bit of the frame to be transmitted is coming from the other CPRI unit in the pair through the auxiliary interface. When a bit is cleared then the relevant bit of the frame to be transmitted is coming from the system memory of this CPRI unit.

Table 18-119. CPRInMASKR<Y> Bit Descriptions

Name	Reset	Description	Setting
MASKR 31–0	0	**Mask Register** Determines the masking bits of the auxiliary interface. If the bit is set, then the auxiliary data for that bit is sent out. If this bit is cleared, then the transmit data for the bit is sent out.	0b0 Use transmit data for bit. 0b1 Use auxiliary data for bit.

18.4.3.35 CPRI Auxiliary Control Register (CPRInAUXCR)

address 0x7a0 (CPRInAUXCR)

	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																AUX-EN
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-143. CPRInAUXCR

This register determines if the auxiliary port is active. The enable bit should be set in both CPRI units.

This register is reset due to auto negotiation reset.

Table 18-120. CPRInAUXCR Bit Descriptions

Name	Reset	Description	Setting
— 31–1	0	Reserved. Write to zero for future compatibility.	
AUXEN 0	0	**Auxiliary Port Enable** Determines if the auxiliary port of CPRIn is active	0x0 The auxiliary port is not active 0x1 The auxiliary port is active

Note: For lanes that operate as pairs in auxiliary mode, when they generate an Error interrupt due to Local Loss of Frame error (bit CPRInEER[LLOF] is set, see **Section 18.4.4.17**), software should disable the CPRInAUXCR[AUX_EN] bit for both lanes of the pair and then immediately reenable both lanes.

18.4.4 Status Registers

18.4.4.1 Receive IQ Buffer Displacement Register (CPRInRIQBDR)

address 0x800 (CPRInRIQBDR)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								RIQBD							
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	RIQBD															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-144. CPRInRIQBDR

CPRInRIQBDR points to the current displacement in the receive IQ buffers where the received IQ data should be written by the CPRI receive DMA. For details, refer to **Section 18.3.3.2**, *Receive IQ Data Flow*

Table 18-121. CPRInRIQBDR Bit Descriptions

Name	Reset	Description	Setting
— 31–24	0	Reserved. Write to zero for future compatibility.	
RIQBD 23–0	0	**Receive IQ Buffer Displacement** Points to the current displacement of the received data in the data buffers. The value is unified to all the antenna carriers.	

18.4.4.2 Receive IQ Second Destination Buffer Displacement Register (CPRInRIQSDBDR)

address 0x804 (CPRInRIQSDBDR)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								RIQSDBD							
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	RIQSDBD															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-145. CPRInRIQSDBDR

CPRInRIQBSDDR points to the current displacement in the receive IQ buffers second destination where the received IQ data should be written by the CPRI receive DMA.

Table 18-122. CPRInRIQSDBDR Bit Descriptions

Name	Reset	Description	Setting
— 31–24	0	Reserved. Write to zero for future compatibility.	
RIQSDBD 23–0	0	**Receive IQ Buffer Second Destination Displacement** Points to the current displacement of the received data in the data buffers of the second destination. The value is unified to all the antenna carriers.	

18.4.4.3 Transmit IQ Buffer Displacement Register (CPRInTIQBDR)

Figure 18-146. CPRInTIQBDR

address 0x808 (CPRInTIQBDR)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								TIQBD							
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	TIQBD															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPRInTIQBDR points to the current displacement in the Transmit IQ buffers where the transmit IQ data should be read from by the CPRI transmit DMA. For details refer to section **Section 18.3.3.3**, *Transmit IQ Data Flow*.

Table 18-123. CPRInTIQBDR Bit Descriptions

Name	Reset	Description	Setting
— 31–24	0	Reserved. Write to zero for future compatibility.	

Table 18-123. CPRInTIQBDR Bit Descriptions

Name	Reset	Description	Setting
TIQBD 23–0	0	**Transmit IQ Buffer Displacement** Points to the current displacement of the transmit data in the IQ data buffers. The value is unified to all the antenna carriers.	

18.4.4.4 Receive Chips Counter Register (CPRInRCCR)

address 0x80c (CPRInRCCR)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RIQBC															
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	—											RCCR				
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-147. CPRInRCCR

The receive chip counter register is active only in chip rate mode (sample width 8 bits, double oversampling). It counts the number of IQ data bytes and chips (basic frames) written to the AxC oversampling buffer in the device memory. The counter granularity is 2560 chips.

Table 18-124. CPRInRCCR Bit Descriptions

Name	Reset	Description	Setting
RIQBC 31–16	0	Receive IQ Byte count. This field determines the number of bytes written to the AxC buffers in the first destination location. The maximum value of this counter is (2560 transaction size) bytes if the transaction size is smaller than 1024 bytes If the transaction size is 1024 bytes, then the maximum value 0x2400 bytes. If the transaction size is 2048 bytes, then the maximum value of the counter is 0x4800 bytes	0x0 The number of bytes that has been written to the system memory is 0. 0x40 The number of bytes that has been written to the system memory is 64. 0x80 The number of bytes that has been written to the system memory is 128. : 0x4800 The number of bytes that has been written to the system memory is 4800 bytes.
— 15–5	0	Reserved. Write to zero for future compatibility.	

Table 18-124. CPRInRCCR Bit Descriptions

Name	Reset	Description	Setting	
RCCR 4–0	0	Receive Chip Counter Registers	0x0 0x1 : 0x1D	The number of chips that has been written to the buffers in the system memory is less than 2560 chips The number of chips that has been written to the buffers in the system memory is at least 2560 chips. The number of chips that has been written to the buffers in the system memory is at least 2560*29 chips.

18.4.4.5 Receive VSS Buffer Displacement Register (CPRInRVSSBDR)

address 0x810 (CPRInRVSSBDR)

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

Figure 18-148. CPRInRVSSBDR

CPRInRVSSBDR points to the current displacement in the Receive VSS buffers where the receive VSS data should be written by the CPRI receive DMA. For details, see **Section 18.3.3.4**, *Receive VSS (Vendor Specific Data) Data Flow*

Table 18-125. CPRInRVSSBDR Bit Descriptions

Name	Reset	Description	Setting
— 31–24	0	Reserved. Write to zero for future compatibility.	
RVSSD 23–0	0	**Receive VSS Buffer Displacement** Points to the current displacement of the received VSS data in the VSS data buffer. **Note:** 16 is the minimum VSS MBus transaction size	

18.4.4.6 Transmit VSS Buffer Displacement Register (CPRInTVSSBDR)

address 0x814 (CPRInTVSSBDR()

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								TVSSD							
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	TVSSD															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-149. CPRInTVSSBDR

CPRInTVSSBDR points to the current displacement in the transmit VSS buffer where the transmit VSS data should be read from by the CPRI transmit DMA. For details, see **Section 18.3.3.5**, *Transmit VSS (Vendor Specific Data) Data Flow*

Table 18-126. CPRInTVSSBDR Bit Descriptions

Name	Reset	Description	Setting
— 31–24	0	Reserved. Write to zero for future compatibility.	
TVSSD 23–0	0	**Transmit VSS Buffer Displacement** Points to the current displacement of the transmit VSS data in the VSS data buffer. **Note:** 16 is the minimum VSS MBus transaction size	

18.4.4.7 Receive Ethernet Buffer Descriptor (CPRInRETHBD)

address 0x 818(CPRInRETHBD 32 MSBs) and 0x81c (CPRInRETHBD 32 LSBs)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	E	—														ABORT
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	DL															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RDBP															
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	RDBP															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-150. CPRInRETHBD

The receive Ethernet buffer descriptor is a 8 bytes registers and it can be read by two SBus read accesses. This register is updated whenever the receive DMA Ethernet controller fetches a new buffer descriptor from the system memory. This register is read only.

Table 18-127. CPRInRETHBD Bit Descriptions

Name	Reset	Description	Setting
E 31	0	**Empty** Indicates whether the data buffer associated with this BD is empty or full. When this bit is cleared, the status and length fields are updated as required. This bit is cleared by the Receive DMA and set by the user.	0 Data buffer associated with this BD is filled with received data, or data reception is aborted due to an error condition. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
— 30–17	0	Reserved. Write to zero for future compatibility.	
ABORT 16	0	**Receive Packer Aborted** The receive packet is aborted due either to an error in the packet or if the packet size is greater than or equal to the Receive Ethernet Buffer Size (CPRInRETHBS). **Note:** This bit is written by the receive DMA	0 Normal operation 1 Ethernet packet aborted.
DL 15–0		**Data Length** The number of bytes written by the Receive DMA into this BD data buffer.	
RDBP 31–0		**Receive Data Buffer Pointer** Points to the first location of the associated data buffer in the system memory. Must be 16byte aligned. This field is updated by the core.	

18.4.4.8 Transmit Ethernet Buffer Descriptor (CPRInTETHBD)

address 0x820 (CPRInTETHBD 32 MSBs) and 0x824 (CPRInTETHBD 32 MSBs)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	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	DL															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TDBP															
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	TDBP															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-151. CPRInTETHBD

The Transmit Ethernet buffer descriptor is 8 bytes registers and it can be read by two bus transactions. This register is updated whenever the transmit Ethernet controller fetches a new buffer descriptor from the system memory. This register is read only.

Table 18-128. CPRInTETHBD Bit Descriptions

Name	Reset	Description	Setting
R 31	0	**Ready** When this bit is cleared, the DSP is free to manipulate this BD or its associated data buffer. The transmit DMA clears this bit after the buffer is transmitted. When this bit is set, no fields of this BD can be written.	0 Data buffer associated with this BD is not ready for transmission. 1 Data buffer ready, not transmitted or is currently being transmitted.
— 30–16	0	Reserved. Write to zero for future compatibility.	
DL 15–0		**Data Length** The number of bytes that should be transmitted out by the CPRI. **Note:** This field must contain a value greater than 0	
TDBP 31–0		**Transmit Data Buffer Pointer** Points to the first location of the associated data buffer in the system memory. There are no alignment requirements for this address.	

18.4.4.9 Receive Ethernet Read Pointer Ring (CPRInRERPR)

address 0x828(CPRInRERPR)

	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	—							RERW	RERPR							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-152. CPRInRERPR

The receive Ethernet read pointer ring indicates the next buffer descriptor that should be read by the receive DMA. The wrap bit is toggled each time that the receive Ethernet controller is back to the beginning of the buffer descriptor ring. This register is updated only by the receive DMA.

Table 18-129. CPRInRERPR Bit Descriptions

Name	Reset	Description	Setting
— 31–9	0	Reserved. Write to zero for future compatibility.	
RERW 8	0	**Receive Ethernet Read Pointer Wrap** This bit is toggled every time that the receive DMA is back to the beginning of the Ethernet buffer descriptor ring.	0 Wrap did not occur 1 Wrap occurred
RERPR 7–0	0	**Receive Ethernet Read Pointer** Indicates the next Ethernet buffer descriptor that should be read by the receive DMA.	

18.4.4.10 Transmit Ethernet Read Pointer Ring (CPRInTERPR)

address 0x82c (CPRInTERPR)

	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	—							TERW	TERPR							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-153. CPRInTERPR

The transmit Ethernet read pointer ring indicates the next buffer descriptor that should be read by the transmit DMA. The wrap bit is toggled each time that the transmit DMA is back to the beginning of the buffer descriptor ring. This register is updated only by the transmit DMA.

Table 18-130. CPRInTERPR Bit Descriptions

Name	Reset	Description	Setting
— 31–9	0	Reserved. Write to zero for future compatibility.	
TERW 8	0	**Transmit Read Pointer Wrap** This bit is toggled every time that the core is back to the beginning of the Ethernet buffer descriptor ring.	0 Wrap did not occur 1 Wrap occurred
TERPR 7–0	0	**Transmit Ethernet Read Pointer Ring** Indicates the next Ethernet buffer descriptor that should be read by the transmit DMA.	

18.4.4.11 Receive HDLC Buffer Descriptor (CPRInRHDLCBD)

address 0x830 (CPRInRHDLCBD 32 MSBs) and 0x834 (CPRInRHDLCBD 32 LSBs)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	E	—														ABORT
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	DL															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RDBP															
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	RDBP															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-154. CPRInRHDLCBD

The receive HDLC buffer descriptor is 8 bytes registers and it can be read by two bus transactions. This register is updated whenever the receive DMA fetches a new HDLC buffer descriptor from the system memory. This is a read-only register.

Table 18-131. CPRInRHDLCBD Bit Descriptions

Name	Reset	Description	Setting
E 31	0	**Empty** Indicates whether the data buffer associated with this BD is empty or full. When this bit is cleared, the status and length fields are updated as required. This bit is cleared by the Receive DMA and set by the user.	0 Data buffer associated with this BD is filled with received data, or data reception is aborted due to an error condition. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
— 30–17	0	Reserved. Write to zero for future compatibility.	
ABORT 16	0	**Receive Packer Aborted** The receive packet is aborted due either to an error in the packet or if the packet size is greater than the Receive HDLC Buffer Size (CPRInRHDLCBS) **Note:** This bit is set by the receive DMA	0 Normal operation 1 HDLC packet aborted.
DL 15–0		**Data Length** The number of bytes written by the receive DMA into this BD data buffer.	
RDBP 31–0		**Receive Data Buffer Pointer** Points to the first location of the associated data buffer in the system memory. Must be 16-byte aligned	

18.4.4.12 Transmit HDLC Buffer Descriptor (CPRInTHDLCBD)

address 0x838 (CPRInTHDLCBD 32 MSBs) and 0x83c (CPRInTHDLCBD 32 LSBs)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	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	DL															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TDBP															
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	TDBP															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-155. CPRInTHDLCBD

The Transmit HDLC buffer descriptor is 8 bytes registers and it can be read by two Bus transactions. This register is updated whenever the transmit DMA fetches a new HDLC buffer descriptor from the system memory. This register is read only.

Table 18-132. CPRInTHDLCBD Bit Descriptions

Name	Reset	Description	Setting
R 31	0	**Ready** When this bit is cleared, the DSP is free to manipulate this BD or its associated data buffer. The transmit DMA Ethernet controller clears this bit after the buffer is transmitted. When this bit is set, no fields of this BD can be written.	0 Data buffer associated with this BD is not ready for transmission. 1 Data buffer not transmitted or is currently being transmitted.
— 30–16	0	Reserved. Write to zero for future compatibility.	
DL 15–0		**Data Length** The number of octets that should be transmitted out by the CPRI. **Note:** This field must contain value greater than 0	
TDBP 31–0		**Transmit Data Buffer Pointer** Points to the first location of the associated data buffer in the system memory. There are no alignment requirements for this address.	

18.4.4.13 Receive HDLC Read Pointer Ring (CPRInRHRPR)

address 0x840 (CPRInRHRPR)

	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	—							RHR	RHRPR							
W								W								
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-156. CPRInRHRPR

The receive HDLC read pointer ring indicates the next buffer descriptor that should be read by the receive DMA. The wrap bit is toggled each time that the DMA controller returns to the beginning of the buffer descriptor ring. This register is updated only by the receive DMA.

Table 18-133. CPRInRHRPR Bit Descriptions

Name	Reset	Description	Setting
— 31–9	0	Reserved. Write to zero for future compatibility.	
RHRW 8	0	**Receive HDLC Read Pointer Wrap** This bit is toggled every time that the receive DMA is back to the beginning of the buffer descriptor ring.	0 Wrap did not occur 1 Wrap occurred
RHRPR 7–0	0	**Receive HDLC Read Pointer** Indicates the next HDLC buffer descriptor that should be read by the receive DMA.	

18.4.4.14 Transmit HDLC Read Pointer Ring (CPRInTHRPR)

address 0x844 (CPRInTHRPR)

	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	—							THRW	THRPR							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-157. CPRInTHRPR

The transmit HDLC read pointer ring indicates the next buffer descriptor that should be read by the transmit DMA. The wrap bit is toggled each time that the transmit DMA returns to the beginning of the buffer descriptor ring. This register is updated only by the transmit DMA.

Table 18-134. CPRInTHRPR Bit Descriptions

Name	Reset	Description	Setting
— 31–9	0	Reserved. Write to zero for future compatibility.	
THRW 8	0	**Transmit HDLC Read Pointer Wrap** This bit is toggled every time that the core is back to the beginning of the HDLC buffer descriptor ring.	0 Wrap did not occur 1 Wrap occurred
THRPR 7–0	0	**Transmit Ethernet Read Pointer Ring** Indicates the next HDLC buffer descriptor that should be read by the transmit DMA.	

18.4.4.15 Receive Event Register (CPRInRER)

address 0x848 (CPRInRER)

	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	—							RIQTSDE	BFNT	HFNT	RETHE	RHDLCE	RVSSTE	RIQsTE	RIQFTE	RIQTE
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-158. CPRInRER

CPRInRER contains the status of the receive IQ, ETH, HDLC, and VSS buffers and general CPRI receive frame timing events. The register can be read at any time. Bits 8–0 are cleared by writing ones to them; writing zero has no effect.

Table 18-135. CPRInRER Bit Descriptions

Name	Reset	Description	Setting
— 31–9	0	Reserved. Write to zero for future compatibility.	
RIQTSD 8	0	**Receive IQ Threshold Second Destination**. This field is set each time the number of bytes transferred to the receive IQ data buffers of the second destination exceeds the RIQTSD value. The second destination receive threshold value is determined by CPRInRIQTSD	0 No receive second destination threshold event has occurred. 1 A receive second destination threshold event has occurred.
RBFNT 7	0	**Receive BFN Frame Timing**. This bit is set at the beginning of BFN frame. The BFN frame timing is 10 ms.	0 New receive BFN frame not detected 1 New receive BFN frame detected
RHFNT 6	0	**Receive Hyper Frame Timing**. This bit is set at the beginning of hyper frame. The HFN frame timing is 66.67us µ.	0 New receive hyper frame not detected 1 New receive hyper frame detected
RETHE 5	0	**Receive Ethernet Event** This field is set every time a receive Ethernet coalescing event occurred. **Note:** This bit is set each time that the coalescing counter equals the Receive Ethernet Coalescing threshold.	0 No receive Ethernet coalescing event occurred 1 Receive Ethernet coalescing event has occurred
RHDLCE 4	0	**Receive HDLC Event** This bit is set every time a receive HDLC packet was transferred to the system memory.	0 No receive HDLC packet transferred 1 Receive HDLC packet transferred
RVSSTE 3	0	**Receive VSS threshold Event** This field is set every time that the VSS buffer in the system memory is filled with RVSST bytes. The number of VSS bytes determined by the Receive VSS threshold register (CPRInRVSST)	0 No receive VSS threshold event occurred. 1 Receive VSS threshold event occurred.
RIQSTE 2	0	**Receive IQ Second Threshold Event** This field is set when the second threshold of all the receive IQ data buffers is filled with received data. The second threshold is determined by the CPRInRIQST register	0 No receive second threshold event occurred. 1 A receive second threshold event occurred.

Table 18-135. CPRInRER Bit Descriptions

Name	Reset	Description	Setting
RIQFTE 1	0	**Receive IQ First Threshold Event** This field is set when the received IQ data in the buffers exceeds the first data threshold. The receive first threshold is determined by the CPRInRIQFT register	0 No receive first threshold event occurred. 1 A receive first threshold event occurred.
RIQTE 0	0	**Receive IQ Threshold Event** This field is set each time th number of IQ bytes transferred to the receive IQ data buffers exceeds the threshold. The receive threshold value is determined by register CPRInRIQT	0 No receive threshold event occurred. 1 A receive threshold event occurred.

18.4.4.16 Transmit Event Register (CPRInTER)

address 0x84c (CPRInTER)

	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	—							TIQRE	TNFNT	THFNT	TETHE	THDLCE	TVSSTE	TIQSTE	TIQFTE	TIQTE
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-159. CPRInTER

CPRInTER register contains the status of the transmit IQ, ETH,HDLC and VSS buffers and general CPRI transmit frame timing events. The register can be read at any time. Bits 8–0 are cleared by writing ones to them; writing zero has no effect

Table 18-136. CPRInTER Bit Descriptions

Name	Reset	Description	Setting
— 31–9	0	Reserved. Write to zero for future compatibility.	
TIQRE 8	0	**Transmit IQ request Event**. This bit is set each time the CPRI DMA starts to fetch new data from system memory.	0 No start to fetch new data from system memory 1 Start to fetch new data from system memory
TBFNT 7	0	**Transmit BFN Frame Timing**. This bit is set at the beginning of transmit BFN frame. The BFN frame timing is 10 ms.	0 New transmit BFN frame not started 1 New transmit BFN frame is started
THFNT 6	0	**Transmit Hyper Frame Timing**. This bit is set at the beginning of transmit hyper frame. The HFN frame timing is 66.67us.	0 New transmit hyper frame not started 1 New transmit hyper frame started
TETHE 5	0	**Transmit Ethernet Event** This field is set every time that transmit Ethernet coalescing event occurred. **Note:** This bit is set each time that the coalescing counter is equal to Transmit Ethernet Coalescing threshold.	0 No transmit Ethernet coalescing event occur 1 Transmit Ethernet coalescing event occur
THDLCE 4	0	**Transmit HDLC Event** This field is set each time that HDLC packet was transferred from system memory.	0 No transmit HDLC packet 1 Transmit HDLC packet

Table 18-136. CPRInTER Bit Descriptions

Name	Reset	Description	Setting
TVSSTE 3	0	**Transmit VSS Threshold Event** This field is set every time the number of TVSST bytes transferred from the VSS buffer in the main memory to CPRI DMA memory exceeds the threshold.The threshold is determined by the Transmit VSS Threshold (TVSST) register	0 No transmit VSS threshold event occurred. 1 A transmit VSS threshold event occurred.
TIQSTE 2	0	**Transmit IQ Second Threshold Event** This field is set when the number of bytes transmit bytes transferred from the transmit IQ buffer exceeds the second threshold. The second threshold pointer is determined by the CPRInTIQST register.	0 No transmit second threshold event occurred. 1 A transmit second threshold event occurred.
TIQFTE 1	0	**Transmit IQ First Threshold Event** This field is set when the number of bytes transmit bytes transferred from the transmit IQ buffer exceeds the first threshold The first threshold pointer is determined by the CPRInTIQFT register	0 No transmit first threshold event occurred. 1 A transmit first threshold event occurred.
TIQTE 1	0	**Transmit IQ Threshold Event** This field is set each time the number of transmit bytes transferred from the transmit IQ buffer exceeds the transmit IQ threshold. The transmit threshold value is determined by CPRInTIQT.	0 No transmit threshold event occurred. 1 A transmit threshold event occurred.

18.4.4.17 Error Event Register (CPRInEER)

address 0x850 (CPRInEER)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—								FA		LLOF	LLOS	RSDI	RRAI	RLOF	RLOS
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	EXSSLOS	—					TVSSUN	RVSSOV	RHBDUN	THDLCOV	RHDLcOV	REBDUN	TETHUN	RETHOV	TIQUN	RIQOV
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-160. CPRInEER

This register contain the status of general error events. The register can be read in any time; the register bits are cleared by writing ones to them. Writing zeros has no effect

Table 18-137. CPRInEER Bit Description

Name	Reset	Description	Setting
— 31–23	0	Reserved. Write to zero for future compatibility.	
FA 23	0	**Frequency Alarm** This bit is set if the frequency of the CDR Clock is not synchronized to the Framer Clock	0 The CDR Clock is synchronized to the Framer Clock 1 The CDR Clock is not synchronized to the Framer Clock
22	0	Reserved. Write to zero for future compatibility.	
LLOF 21	0	**Local Loss of Frame** LLOF bit is set due to loss of received frame synchronization.	0 No local LOF detection 1 Local LOF detection.
LLOS 20	0	**Local Loss of Signal** LLOS bit is set due to 8b/10b line code errors (LCV)	0 No local LOS detection 1 Local LOS detection.
RSDI 19		**Remote SAP Defect Indication** The RSDI bit (z.130.0 b2) in the received CPRI frame is set.	0 No remote SDI detection 1 Remote SDI detect.
RRAI 18	0	**Remote Alarm Indication** The RAI bit (z.130.0 b1) in the received CPRI frame is set.	0 No remote RAI detection 1 Remote RAI detect.
RLOF 17	0	**Remote Loss of Frame** The LOF bit (z.130.0 b4) in the received CPRI frame is set.	0 No remote LOF detection 1 Remote LOF detect.
RLOS 16	0	**Remote Loss of Signal** The LOS bit (z.130.0 b3) in the received CPRI frame is set	0 No remote LOS detection 1 Remote LOS detect.
EXSSLOS 15	0	**External Sync Synchronization Lost** The external sync signal connected to the timer lost synchronization. This bit is set whenever the sync state machine of the timer returns to HUNT state from SYNC state.	0 Normal operation 1 External sync synchronization is lost
— 14–10	0	Reserved. Write to zero for future compatibility.	

Table 18-137. CPRInEER Bit Description (Continued)

Name	Reset	Description	Setting
TVSSUN 9	0	**Transmit VSS Underrun** Indicates whether a VSS underrun event has occurred. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the internal MBus bus and therefore it cannot read enough VSS data from the VSS buffers in the system memory.	0 No VSS underrun event occurred. 1 VSS underrun event occurred
RVSSOV 8	0	**Receive VSS Overrun** Indicates whether a VSS overrun event has occurred. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the MBus bus and therefore it can not write the VSS data to VSS buffer mapped on the system memory.	0 No VSS overrun event occurred 1: VSS overrun event occurred
RHBDUN 7	0	**Receive HDLC Buffer Descriptor Underrun** Indicates whether a new packet arrive and the buffer descriptor ring does not contain valid buffer descriptor. This error should not occur during normal operation. It indicates that the core program is not well defined or the CPRI does not receive enough bandwidth on the MBus.	0 No receive HDLC buffer descriptor underrun occurred 1 Receive HDLC buffer descriptor underrun occurred
THDLCUN 6	0	**Transmit HDLC Underrun** Indicates whether an underrun event has occurred. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the MBus and therefore it can not read the HDLC data from system memory	0 No transmit HDLC underrun occurred 1 Transmit HDLC underrun occurred
RHDLCOV 5	0	**Receive HDLC Overrun** Indicates whether an HDLC overrun event has occurred. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the MBus and therefore it can not write the HDLC data to system memory	0 No receive HDLC overrun occurred 1 Receive HDLC overrun occurred
REBDUN 4	0	**Receive Ethernet Buffer Descriptor Underrun** Indicates whether a new Ethernet packet arrived and the buffer descriptor ring did not contain valid buffer descriptor. This error should not occur during normal operation. It indicates that the core program is not defined well or the CPRI receive does not have enough bandwidth on the MBus.	0 No receive Ethernet buffer descriptor underrun occurred. 1 Receive Ethernet buffer descriptor underrun occurred
TETHUN 3	0	**Transmit Ethernet Underrun Occurred** Indicates whether an Ethernet underrun event has occurred. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the MBus and therefore it can not read the Ethernet data to system memory	0 No transmit Ethernet underrun occurred 1 Transmit Ethernet underrun occurred
RETHOV 2	0	**Receive Ethernet Overrun Occurred**. Indicates whether an Ethernet overrun event has occurred. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the MBus and therefore it can not write the Ethernet data to system memory	0 No receive Ethernet overrun occurred 1 Receive Ethernet overrun occurred

Table 18-137. CPRInEER Bit Description (Continued)

Name	Reset	Description	Setting
TIQUN 1	0	**Transmit IQ Underrun Occurred** Indicates whether an IQ underrun event has occurred. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the internal MBus and therefore it cannot read the data from the AxC buffers in the system memory.	0 No IQ underrun event occurred 1 IQ underrun event occurred
RIQOV 0	0	**Receive IQ Overrun Occurred** Indicates whether an IQ overrun event has occurred. This error should not occur during normal operation. It indicates that the CPRI has not received enough bandwidth on the MBus bus and therefore it can not write the IQ data to system memory.	0 No IQ overrun event occurred memory. 1 IQ Overrun event occurred

18.4.4.18 Receive Ethernet Coalescing Status (CPRInRETHCS)

address 0x854 (CPRInRETHCS)

	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	—								RETHCS							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-161. CPRInRETHCS

The RETHCS register specifies the coalescing counter value. The register is reset every time that receive coalescing event occurred.

Table 18-138. CPRInRETHCS Bit Descriptions

Name	Reset	Description	Setting
— 31–8	0	Reserved. Write to zero for future compatibility.	
RETHCS 7–0	0	**Receive Ethernet Coalescing Status** This field indicates the coalescing counter value	0x0 No Ethernet packet received 0x1 One Ethernet packet transferred to system memory. 0x2 Two Ethernet packets transferred to system memory : 0xFF 255 packets transferred to system memory.

18.4.4.19 Transmit Ethernet Coalescing Status (CPRInTETHCS)

address 0x858 (CPRInTETHCS)

	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	—								TETHCS							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-162. CPRInTETHCS

TETHCS register specifies the transmit coalescing counter value. The register is reset every time that transmit coalescing event occurred.

Table 18-139. CPRInTETHCS Bit Descriptions

Name	Reset	Description	Setting
— 31–8	0	Reserved. Write to zero for future compatibility.	
TETHCS 7–0	0	**Transmit Ethernet Coalescing Status** This field indicates the transmit coalescing counter value	0x0 No Ethernet packet transmit 0x1 One Ethernet packet transferred from system memory. 0x2 Two Ethernet packets transferred from system memory : 0xFF 255 packets transferred from system memory.

18.4.4.20 TIMERn Status Register (CPRInTMRSR)

address 0x85c(CPRInTMRSR)

	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	—														SSS	
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-163. CPRInTMRSR

This register shows the status of the sync state machine inside the timer module. For details, see **Section 18.3.8, *Timers***.

Table 18-140. CPRInTETHCS Bit Descriptions

Name	Reset	Description	Setting
— 31–2	0	Reserved. Write to zero for future compatibility.	
SSS 1–0	0	**Sync Synchronization Status** Indicates the status of the synchronization state machine of TIMERn.	00 HUNT state 01 WAIT state 10 SYNC state. 11 PRESYNC state

18.4.4.21 Receive Status Register (CPRInRSR)

address 0x860 (CPRInRSR)

	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	—												RVSSS	RHDLCS	RETHS	RIQS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-164. CPRInRSR

CPRInRSR contains the status of the receive DMA. If disable is done during normal operation, then these bits are cleared only after all the open write MBus transactions are finished.

Table 18-141. CPRInRSR Bit Descriptions

Name	Reset	Description	Setting
— 31–4	0	Reserved. Write to zero for future compatibility.	
RVSSS 3	0	**Receive VSS Enabled**	0 The receive VSS is disabled 1 The received VSS is enabled
RHDLCS 2	0	**Receive HDLC Enabled**	0 The receive HDLC is disabled 1 The received HDLC is enabled
RETHS 1	0	**Receive Ethernet Enabled**	0 The receive ETH is disabled 1 The received ETH is enabled
RIQS 0	0	**Receive IQ Enabled**.	0 The receive IQ is disabled 1 The received IQ is enabled

18.4.4.22 Transmit Status Register (CPRInTSR)

address 0x864 (CPRInTSR)

	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	—												TVSSS	THDLCS	TETHS	TIQS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-165. CPRInTSR

CPRInTSR contains the status of the transmit DMA. If disable is done during normal operation, then these bits are cleared only after all the open read MBus transactions are finished.

Table 18-142. CPRInTSR Bit Descriptions

Name	Reset	Description	Setting
— 31–2	0	Reserved. Write to zero for future compatibility.	
TVSSS 3	0	**Transmit VSS Enabled**	0 The transmit VSS is disabled 1 The transmit VSS is enabled
THDLCS 2	0	**Transmit HDLC Enabled**	0 The transmit HDLC is disabled 1 The transmit HDLC is enabled
TETHS 1	0	**Transmit Ethernet Enabled**	0 The transmit ETH is disabled 1 The transmit ETH is enabled
TIQS 0	0	**Transmit IQ Enabled**.	0 The transmit IQ is disabled 1 The transmit IQ is enabled

18.4.4.23 Receive Configuration Memory (CPRInRCM_<i>)

address 0x1800 (CPRInRCM_<i>), offset 4, range 0–959

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WIDTH0						POSITION0					AC0				EN0
RESET:																

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WIDTH1						POSITION1					AC1				EN1
RESET:																

Figure 18-166. CPRInRCM_<i>

The receive configuration memory contains 960 entries where each entry contains description of two slots. Bits 31–15 describe the ac number, position and width for slot number <i> and bits 15–0 describe the ac number, position and width for slot number <i+1>.

The sum of the width and position fields must not exceed 48, that is, the last bit of an entry must be present no later than 96 bit after the default position of the first bit.

All the 960 registers of the receive configuration memory must be reset before use.

Table 18-143. CPRInRCM_<i> Bit Descriptions

Name	Reset	Description	Setting
WIDTH0 31–27	0	**WIDTH of <i>** Width of segment<i>. The value will always be multiplied by two, that is, the value 0x1 corresponds to a width of 2, and 0x1F corresponds to a width of 62.	0x0 the AxC container width is 0.The value is valid only if EN0 is equal to 0 also 0xf The AxC container width is 30bits 0x10 The AxC container width is 32 bits
POSITION0 26–22	0	**AxC position in slot <i>** Start bit position of AxC container with respect to the start of the slot. The value will always be multiplied by two, that is, the value 0x1 corresponds to a position 2, and 0xF corresponds to position of 30.	0x0 the AxC container starts at bit 0. 0x1 The AxC container starts at bit 2 0x2 The AxC container starts at bit 4 ... 0xf The AxC container start at bit 30
AC0 21–17	0	**AxC number of slot <i>** AxC interface number mapped into segment <i>.	0x0 AxC0 0x1 AxC1 0x17 AxC23
EN0 **16**	0	**ENABLE** Enable mapping of IQ sample into slot <i>	0x0 Enabled 0x1 Disabled
WIDTH1 31–27	0	**WIDTH of <i+1>** Width of segment<i+1>. The value will always be multiplied by two, that is, the value 0x1 corresponds to a width of 2, and 0x1F corresponds to a width of 62.	0x0 the AxC container width is 0.The value is valid only if EN0 is equal to 0 also 0xf The AxC container width is 30 bits 0x10 The AxC container width is 32 bits
POSITION1 26–22	0	**AxC position in slot <i+1>** Start bit position of AxC container with respect to the start of the slot. The value will always be multiplied by two, that is, the value 0x1 corresponds to a width of 2, and 0xF corresponds to potion of 30.	0x0 the AxC container starts at bit 0. 0x1 The AxC container starts at bit 2 0x2 The AxC container starts at bit 4 ... 0xf The AxC container starts at bit 30
AC1 21–17	0	**AxC number of slot <i+1>** AxC interface number mapped into segment <i+1>.	0x0 the AxC0. 0x1 The AxC1 0x2 The AxC2 0x17 The AXC23
EN1 **16**	0	**ENABLE** Enable mapping of IQ sample into slot <i+1>	0x0 Enable 0x1 Disable

18.4.4.24 Transmit Configuration Memory (CPRInTCM_<i>)

address 0x1800 (CPRInRCM_<i>), offset 4, range 0–959

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W RESET:	WIDTH0					POSITION0					AC0					EN0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W RESET:	WIDTH1					POSITION1					AC1					EN1

Figure 18-167. CPRInRCM_<i>

The transmit configuration memory contain 960 entries where each entry contain description of two slots. Bits 31–15 describes the ac number, position and width that mapped to slot number <i> and bits 15–0 describes the ac number, position and width that mapped to slot number <i+1> The sum of the width and position fields must not exceed 48, that is, the last bit of an entry must be present no later than 96 bit after the default position of the first bit. All the 960 registers of the transmit configuration memory must be reset before use.

Table 18-144. CPRInTCM_<i> Bit Descriptions

Name	Reset	Description	Setting
WIDTH0 31–27	0	**WIDTH of <i>** Width of segment<i>. The value will always be multiplied by two, that is, the value 0x1 corresponds to a width of 2, and 0x1F corresponds to a width of 62.	0x0 The AxC container width is 0.The value is valid only if EN0 is equal to 0 also 0xF The AxC container width is 30 0x10 The AxC container width is 32
POSITION0 26–22	0	**AxC position in slot <i>** Start bit position of AxC container with respect to the start of the slot. The value will always be multiplied by two, that is, the value 0x1 corresponds to a width of 2, and 0xF corresponds to potion of 30.	0x0 The AxC container starts at bit 0. 0x1 The AxC container starts at bit 2 0x2 The AxC container starts at bit 4 0xF The AxC container starts at bit 30
AC0 21–17	0	**AxC number of slot <i>** AxC interface number mapped into segment <i>.	0x0 The AxC container start at bit 0. 0x1 The AxC container start at bit 2 0x2 The AxC container start at bit 4 0xF The AxC container start at bit 30
EN0 **16**	0	**ENABLE** Enable mapping of IQ sample into slot <i>	0b0 Enable 0b1 Disable
WIDTH1 31–27	0	**WIDTH of <i+1>** Width of segment<i+1>. The value will always be multiplied by two, that is, the value 0x1 corresponds to a width of 2, and 0x1F corresponds to a width of 62.	0x00 The AxC container width is 0.The value is valid only if EN0 is equal to 0 also 0x0F The AxC container width is 30 0x10 The AxC container width is 32

Table 18-144. CPRInTCM_<i> Bit Descriptions

Name	Reset	Description	Setting
POSITION1 26–22	0	**AxC position in slot <i+1>** Start bit position of AxC container with respect to the start of the slot. The value is always multiplied by two, that is, the value 0x1 corresponds to a width of 2, and 0xF corresponds to potion of 30.	0x0 The AxC container starts at bit 0. 0x1 The AxC container starts at bit 2 0x2 The AxC container starts at bit 4 0xF The AxC container starts at bit 30
AC1 21–17	0	**AxC number of slot <i+1>** AxC interface number mapped into segment <i+1>.	0x0 The AxC container start at bit 0. 0x1 The AxC container start at bit 2 0x2 The AxC container start at bit 4 0xF The AxC container start at bit 30
EN1 **16**	0	**ENABLE** Enable mapping of IQ sample into slot <i+1>	0b0 Enable 0b1 Disable

18.4.4.25 CPRI Control Clocks Register (CPRICCR)

address 0x18000 (CPRICCR)

	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	—										CLK6E	CLK5E	CLK4E	CLK3E	CLK2E	CLK1E
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

= Unimplemented or Reserved

Figure 18-168. CPRICCR

This register enables the clocks of the CPRIn Framer and CPRIn DMA modules. This register should be configured first.

Table 18-145. CPRICCR Bit Descriptions

Name	Reset	Description	Setting
— 31–6	—	Reserved for future use	
CLK6E 5	0	**Clocks of CPRI6 Enable** Enable the clocks of CPRI6 modules	0 CPRI6 clocks disable 1 CPRI6 clocks enable
CLK5E 4	0	**Clocks of CPRI5 Enable** Enable the clocks of CPRI5 modules	0 CPRI5 clocks disable 1 CPRI5 clocks enable
CLK4E 3	0	**Clocks of CPRI4 Enable** Enable the clocks of CPRI4 modules	0 CPRI4 clocks disable 1 CPRI4 clocks enable
CLK3E 2	0	**Clocks of CPRI3 Enable** Enable the clocks of CPRI3 modules	0 CPRI3 clocks disable 1 CPRI3 clocks enable
CLK2E 1	0	**Clocks of CPRI2 Enable** Enable the clocks of CPRI2 modules	0 CPRI2 clocks disable 1 CPRI2 clocks enable
CLK1E 0	0	**Clocks of CPRI1 Enable** Enable the clocks of CPRI2 modules	0 CPRI1 clocks disable 1 CPRI1 clocks enable

18.4.4.26 CPRI Interrupt Control Register y (CPRIICR<y>)

address 0x18000 (CPRIICRy), offset 4, range 1 –12

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ERS				—											LEVEL
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	—							EIQE	EBFNT	HFNTE	EETHE	EHDLCE	EVSSE	EIQSTE	EIQFTE	EIQTE
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPRIICR<y> determines the CPRI interrupt generation. The interrupts are asserted during normal operation. y can be any integer number in the range of 1 to 12. CPRIICR<y> register has 3 different field types: Field ERS selects the event register, field LEVEL the interrupt type (edge or level) and the LSBs the enabled interrupt having the same format as CPRInRER and CPRInTER registers.

Table 18-146. CPRIICRy Bit Descriptions

Name	Reset	Description	Setting
ERS 31–28	0	**Event Register Select** This field selects event registers.	0x0 CPRI1RER register is selected 0x1 CPRI1TER register is selected 0x2 CPRI2RER register is selected 0x3 CPRI2TER register is selected 0x4 CPRI3RER register is selected 0x5 CPRI3TER register is selected 0x6 CPRI4RER register is selected 0x7 CPRI4TER register is selected 0x8 CPRI5RER register is selected 0x9 CPRI5TER register is selected 0xA CPRI6RER register is selected 0xB CPRI6TER register is selected
— 27–17	0	Reserved. Write to zero for future compatibility.	
LEVEL 16	0	**LeveL interrupt** Selects whether the interrupt is edge or level triggered.	0 This interrupt is edge triggered 1 This interrupt is level triggered.
— 15–9	0	Reserved. Write to zero for future compatibility.	
EIQE 8	0	**Enable IQ event** Enable assertion of the CPRI_INTy interrupt when TIQRE bit is set if TER is selected. Enable assertion of the CPRI_INTy interrupt when RIQTSDT bit is set if RER is selected The event register is selected by the ERS field.	0 IQ event is masked 1 IQ event is enabled.
EBFNTE 7	0	**Enable BFN Frame Timing Event** Enable assertion of the CPRI_INTy interrupt when the R/TBFNT bit is set. The event register is selected by the ERS field.	0 BFN timing event is masked 1 BFN timing event is enabled.

Table 18-146. CPRIICRy Bit Descriptions

Name	Reset	Description	Setting
EHFNTE 6	0	**Enable Hyper Frame Timing Event** Enable assertion of the CPRI_INTy interrupt when the R/THFNT bit is set. The event register is selected by the ERS field.	0 HFN event is masked 1 HFN event is enabled.
EETHE 5	0	**Enable Ethernet Event** Enable assertion of the CPRI_INTy interrupt when the R/TETHE bit is set. The event register is selected by the ERS field.	0 Ethernet event is masked 1 Ethernet event is enabled.
EHDLCE 4	0	**Enable HDLC Event** Enable assertion of the CPRI_INTy interrupt when the R/THDLCE bit is set. The event register is selected by the ERS field.	0 HDLC event is masked 1 HDLC event is enabled.
EVSS 3	0	**Enable VSS Event** Enable assertion of the CPRI_INTy interrupt when the R/TVSSE bit is set. The event register is selected by the ERS field.	0 VSS event is masked 1 VSS event is enabled.
EIQSTE 2	0	**Enable IQ Second Threshold Event** Enable assertion of the CPRI_INTy interrupt when the R/TIQSTE bit is set. The event register is selected by the ERS field	0 IQ second threshold event is masked 1 IQ second threshold event is enabled.
EIQFTE 1	0	Enable IQ first Threshold Event Enable assertion of the CPRI_INTy interrupt when the R/TIQFTE bit is set. The event register is selected by the ERS field	0 IQ first threshold event is masked 1 IQ first threshold event is enabled.
EIQTE 0	0	Enable IQ Threshold Event Enable assertion of the CPRI_INTy interrupt when the R/TIQTE bit is set. The event register is selected by the ERS field	0 IQ threshold event is masked 1 IQ threshold event is enabled.

18.4.4.27 CPRI Receive CPU Control Interrupt Enable Register (CPRIRCCIER)

address 0x18050 (CPRIRCCIER)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RCPULEVEL														RETHE6	RHDLE6
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	RVSSE6	RETHE5	RHDLE5	RVSSE5	RETHE4	RHDLCE4	RVSSE4	RETHE3	RHDLE3	RVSSE3	RETH2	RHDLCE2	RVSSE2	RETHE1	RHDLCE1	RVSSE1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-169. CPRIRCCIER

CPRIRCCIER enables the receive CPU interrupt. The CPU interrupt is the control interrupt sent to an external host. The CPU interrupt can be pulse or level.

Table 18-147. CPRIRCCIER Bit Descriptions

Name	Reset	Description	Setting
RCPULEVEL 31	0	**Receive CPU Interrupt Level** This bit determines whether the receive CPU interrupt is pulse or level	0 The CPU interrupt is pulse triggered 1 This CPU interrupt is level triggered.
— 30–18	0	Reserved. Write to zero for future compatibility.	
RETHE6 17	0	**Receive Ethernet Event of CPRI6 is Enabled** If this bit is set and the status bit CPRI6RER[RETHE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive Ethernet Event of CPRI6 is Disabled 1 Receive Ethernet Event of CPRI6 is enabled
RHDLCE6 16	0	**Receive HDLC Event of CPRI6 is Enabled** If this bit is set and the status bit CPRI6RER[RHDLCE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive HDLC Event of CPRI6 is Disabled 1 Receive HDLC Event of CPRI6 is enabled
RVSSE6 15	0	**Receive VSS Event of CPRI6 is Enabled** If this bit is set and the status bit CPRI6RER[RVSSE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive VSS event of CPRI6 is Disabled 1 Receive VSS Event of CPRI6 is enabled
RETHE5 14	0	**Receive Ethernet Event of CPRI5 is Enabled** If this bit is set and the status bit CPRI5RER[RETHE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive Ethernet Event of CPRI5 is Disabled 1 Receive Ethernet Event of CPRI5 is enabled
RHDLCE5 13	0	**Receive HDLC Event of CPRI5 is Enabled** If this bit is set and the status bit CPRI5RER[RHDLCE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive HDLC Event of CPRI5 is Disabled 1 Receive HDLC Event of CPRI5 is enabled
RVSSE5 12	0	**Receive VSS Event of CPRI5 is Enabled** If this bit is set and the status bit CPRI5RER[RVSSE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive VSS event of CPRI5 is Disabled 1 Receive VSS Event of CPRI5 is enabled
RETHE4 11	0	**Receive Ethernet Event of CPRI4 is Enabled** If this bit is set and the status bit CPRI4RER[RETHE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive Ethernet Event of CPRI4 is Disabled 1 Receive Ethernet Event of CPRI4 is enabled

Table 18-147. CPRIRCCIER Bit Descriptions (Continued)

Name	Reset	Description	Setting
RHDLCE4 10	0	**Receive HDLC Event of CPRI4 is Enabled** If this bit is set and the status bit CPRI4RER[RHDLCE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive HDLC Event of CPRI4 is Disabled 1 Receive HDLC Event of CPRI4 is enabled
RVSSE4 9	0	**Receive VSS Event of CPRI4 is Enabled** If this bit is set and the status bit CPRI4RER[RVSSE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive VSS event of CPRI4 is Disabled 1 Receive VSS Event of CPRI4 is enabled
RETHE3 8	0	**Receive Ethernet Event of CPRI3 is Enabled** If this bit is set and the status bit CPRI3RER[RETHE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive Ethernet Event of CPRI3 is Disabled 1 Receive Ethernet Event of CPRI3 is enabled
RHDLCE3 7	0	**Receive HDLC Event of CPRI3 is Enabled** If this bit is set and the status bit CPRI3RER[RHDLCE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive HDLC Event of CPRI3 is Disabled 1 Receive HDLC Event of CPRI3 is enabled
RVSSE3 6	0	**Receive VSS Event of CPRI3 is Enabled** If this bit is set and the status bit CPRI3RER[RVSSE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive VSS event of CPRI3 is Disabled 1 Receive VSS Event of CPRI3 is enabled
RETHE2 5	0	**Receive Ethernet Event of CPRI2 is Enabled** If this bit is set and the status bit CPRI2RER[RETHE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive Ethernet Event of CPRI2 is Disabled 1 Receive Ethernet Event of CPRI2 is enabled
RHDLCE2 4	0	**Receive HDLC Event of CPRI2 is Enabled** If this bit is set and the status bit CPRI2RER[RHDLCE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive HDLC Event of CPRI2 is Disabled 1 Receive HDLC Event of CPRI2 is enabled
RVSSE2 3	0	**Receive VSS Event of CPRI 2is Enabled** If this bit is set and the status bit CPRI2RER[RVSSE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive VSS event of CPRI2 is Disabled 1 Receive VSS Event of CPRI2 is enabled
RETHE1 2	0	**Receive Ethernet Event of CPRI1 is Enabled** If this bit is set and the status bit CPRI1RER[RETHE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive Ethernet Event of CPRI 1 is Disabled 1 Receive Ethernet Event of CPRI 1 is enabled
RHDLCE1 1	0	**Receive HDLC Event of CPRI1 is Enabled** If this bit is set and the status bit CPRI1RER[RHDLCE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive HDLC Event of CPRI1 is Disabled 1 Receive HDLC Event of CPRI1 is enabled
RVSSE1 0	0	**Receive VSS Event of CPRI 1 is Enabled** If this bit is set and the status bit CPRI1RER[RVSSE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Receive VSS event of CPRI1 is Disabled 1 Receive VSS Event of CPRI1 is enabled

18.4.4.28 CPRI Transmit CPU Control Interrupt Enable Register (CPRITCCIER)

address 0x18054 (CPRITCCIER)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TCPULEVEL	—													TETHE6	THDLCE6
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	TVSSE6	TETHE5	THDLCE5	TVSSE5	TETHE4	THDLCE4	TVSSE4	TETHE3	THDLCE3	TVSSE3	TETHE2	THDLCE2	TVSSE2	TETHE1	THDLCE1	TVSSE1
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-170. CPRIxTCCIER

CPRITCCIER enables the transmit CPU interrupt. The CPU interrupt output the control interrupt to external host. The CPU interrupt may be pulse or level.

Table 18-148. CPRIxTCCIER Bit Descriptions

Name	Reset	Description	Setting
TCPULEVEL 31	0	**Transmit CPU Interrupt LeveL** This bit determines if the receive CPU interrupt will be pulse or level	0 The transmit CPU interrupt is pulse triggered 1 The transmit CPU interrupt is level triggered.
— 30–18	0	Reserved. Write to zero for future compatibility.	
TETHE6 17	0	**Transmit Ethernet Event of CPRI6 is Enabled** If this bit is set and the status bit CPRI6TER[TETHE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit Ethernet Event of CPRI6 is Disabled 1 Transmit Ethernet Event of CPRI6 is enabled
THDLCE6 16	0	**Transmit HDLC Event of CPRI6 is Enabled** If this bit is set and the status bit CPRI6TER[THDLCE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit HDLC Event of CPRI6 is Disabled 1 Transmit HDLC Event of CPRI6 is enabled
TVSSE6 15	0	**Transmit VSS Event of CPRI6 is Enabled** If this bit is set and the status bit CPRI6TER[TVSSE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit VSS event of CPRI6 is Disabled 1 Transmit VSS Event of CPRI6 is enabled
TETHE5 14	0	**Transmit Ethernet Event of CPRI5 is Enabled** If this bit is set and the status bit CPRI5TER[TETHE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit Ethernet Event of CPRI5 is Disabled 1 Transmit Ethernet Event of CPRI5 is enabled
THDLCE5 13	0	**Transmit HDLC Event of CPRI5 is Enabled** If this bit is set and the status bit CPRI5TER[THDLCE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit HDLC Event of CPRI5 is Disabled 1 Transmit HDLC Event of CPRI5 is enabled
TVSSE5 12	0	**Transmit VSS Event of CPRI5 is Enabled** If this bit is set and the status bit CPRI5TER[TVSSE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit VSS event of CPRI5 is Disabled 1 Transmit VSS Event of CPRI5 is enabled
TETHE4 11	0	**Transmit Ethernet Event of CPRI4 is Enabled** If this bit is set and the status bit CPRI4TER[TETHE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit Ethernet Event of CPRI4 is Disabled 1 Transmit Ethernet Event of CPRI4 is enabled

Table 18-148. CPRIxTCCIER Bit Descriptions

Name	Reset	Description	Setting
THDLCE4 10	0	**Transmit HDLC Event of CPRI4 is Enabled** If this bit is set and the status bit CPRI4TER[THDLCE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit HDLC Event of CPRI4 is Disabled 1 Transmit HDLC Event of CPRI4 is enabled
TVSSE4 9	0	**Transmit VSS Event of CPRI4 is Enabled** If this bit is set and the status bit CPRI4TER[TVSSE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit VSS event of CPRI4 is Disabled 1 Transmit VSS Event of CPRI4 is enabled
TETHE3 8	0	**Transmit Ethernet Event of CPRI3 is Enabled** If this bit is set and the status bit CPRI3TER[TETHE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit Ethernet Event of CPRI3 is Disabled 1 Transmit Ethernet Event of CPRI3 is enabled
THDLCE3 7	0	**Transmit HDLC Event of CPRI3 is Enabled** If this bit is set and the status bit CPRI3TER[THDLCE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit HDLC Event of CPRI3 is Disabled 1 Transmit HDLC Event of CPRI3 is enabled
TVSSE3 6	0	**Transmit VSS Event of CPRI3 is Enabled** If this bit is set and the status bit CPRI3TER[TVSSE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit VSS event of CPRI3 is Disabled 1 Transmit VSS Event of CPRI3 is enabled
TETHE2 5	0	**Transmit Ethernet Event of CPRI2 is Enabled** If this bit is set and the status bit CPRI2TER[TETHE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit Ethernet Event of CPRI2 is Disabled 1 Transmit Ethernet Event of CPRI2 is enabled
THDLCE2 4	0	**Transmit HDLC Event of CPRI2 is Enabled** If this bit is set and the status bit CPRI2TER[THDLCE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit HDLC Event of CPRI2 is Disabled 1 Transmit HDLC Event of CPRI2 is enabled
TVSSE2 3	0	**Transmit VSS Event of CPRI 2is Enabled** If this bit is set and the status bit CPRI2TER[TVSSE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit VSS event of CPRI2 is Disabled 1 Transmit VSS Event of CPRI2 is enabled
TETHE1 2	0	**Transmit Ethernet Event of CPRI1 is Enabled** If this bit is set and the status bit CPRI1TER[TETHE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit Ethernet Event of CPRI 1 is Disabled 1 Transmit Ethernet Event of CPRI 1 is enabled
THDLCE1 1	0	**Transmit HDLC Event of CPRI1 is Enabled** If this bit is set and the status bit CPRI1TER[THDLCE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit HDLC Event of CPRI1 is Disabled 1 Transmit HDLC Event of CPRI1 is enabled
TVSSE1 0	0	**Transmit VSS Event of CPRI1 is Enabled** If this bit is set and the status bit CPRI1TER[TVSSE] is set then CPRIn_RX_CPU_INT interrupt is generated.	0 Transmit VSS event of CPRI1 is Disabled 1 Transmit VSS Event of CPRI1 is enabled

18.4.4.29 General Receive Synchronization Register (CPRIGRSR)

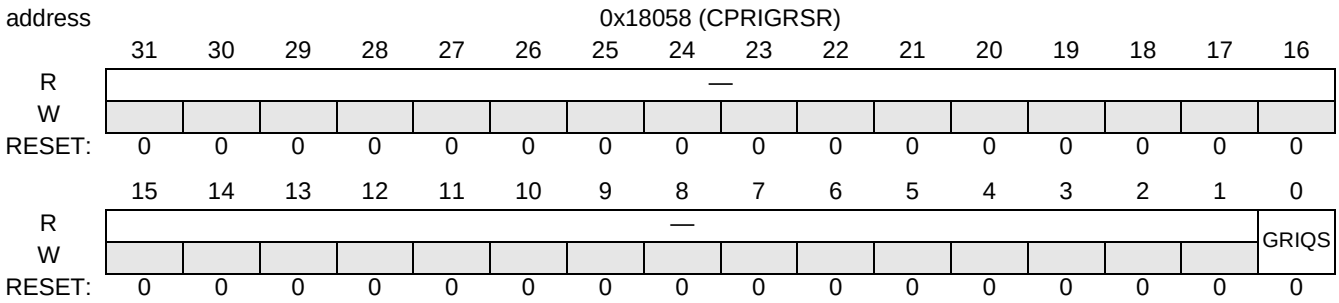


Figure 18-171. CPRIGRSR

This register determines whether synchronization between the CPRI receivers is required. This register should be written only after the RCR.RIQSYNC bits of the CPRI modules are written. See details at **Section 18.4.3.1**, *Receive Control Register (CPRInRCR)*, on page 18-120.

Table 18-149. CPRIGRSR Bit Descriptions

Name	Reset	Description	Setting
— 31–1	0	Reserved. Write to zero for future compatibility.	
GRIQS 0	0	**General Receive IQ synchronization is required** If this bit is set receive synchronization between the CPRI modules is required	0b0 Synchronization between the receive CPRI modules is not required 0b1 Synchronization between the receive CPRI modules is required

18.4.4.30 General Transmit Synchronization Register (CPRIGTSR)

address 0x1805c (CPRIGTSR)

	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	—															GTIQS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-172. CPRIGtSR

This register determines if synchronization between the CPRI transmitters is required. This register should be written after configuration of TCR.TIQSYNC registers. See details at **Section 18.4.3.2**, *Transmit Control Register (CPRInTCR)*, on page 18-121..

Table 18-150. CPRIGRSR Bit Descriptions

Name	Reset	Description	Setting
— 31–1	0	Reserved. Write to zero for future compatibility.	
GTIQS 0	0	**General Transmit IQ synchronization** If this bit is set transmit synchronization between the CPRI modules is required	0x0 Synchronization between the transmit CPRI modules is not required 0x1 Synchronization between the transmit CPRI modules is required

18.4.4.31 CPRI Error Status Register (CPRIESR)

address 0x18094 (CPRIESR)

	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	—										CPRI6 ERR	CPRI5 ERR	CPRI4 ERR	CPRI3 ERR	CPRI2 ERR	CPRI1 ERR
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-173. CPRIESR

CPRIESR register indicates if error occurred in the CPRI modules.

Table 18-151. CPRIESR Bit Descriptions

Name	Reset	Description	Setting
— 31–6	0	Reserved. Write to zero for future compatibility.	
CPRI6ERR 5	0	**CPRI 6 Error** This bit indicate if error occurred in CPRI 6	0 No error occurred in CPRI 6 1 Error occurred in CPRI 6
CPRI5ERR 4	0	**CPRI 5 Error** This bit indicate if error occurred in CPRI 5	0 No error occurred in CPRI 5 1 Error occurred in CPRI 5
CPRI4ERR 3	0	**CPRI 4 Error** This bit indicate if error occurred in CPRI 4	0 No error occurred in CPRI 4 1 Error occurred in CPRI 4
CPRI3ERR 2	0	**CPRI 3 Error** This bit indicate if error occurred in CPRI 3	0 No error occurred in CPRI 3 1 Error occurred in CPRI 3
CPRI2ERR 1	0	**CPRI 2 Error** This bit indicate if error occurred in CPRI 2	0 No error occurred in CPRI 2 1 Error occurred in CPRI 2
CPRI1ERR 0	0	**CPRI 1Error** This bit indicate if error occurred in CPRI 1	0 No error occurred in CPRI 1 1 Error occurred in CPRI 1

19 QUICC Engine Subsystem

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 MSC8157E, which include the following:

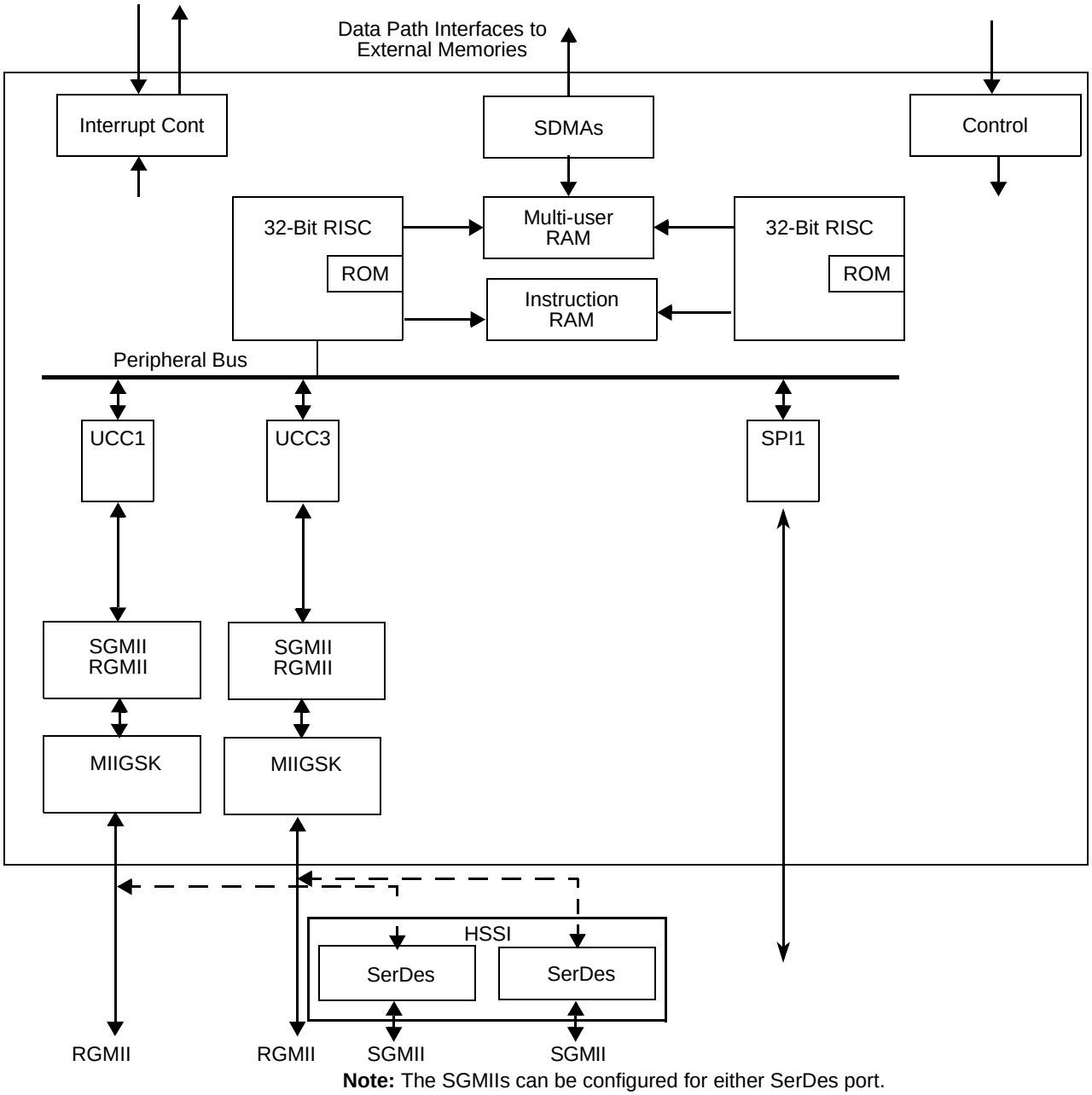
- Dual RISC engines with
  - Internal interfaces to the core and peripherals
  - Parameter RAM
  - Buffer descriptors
  - Multithreading operation
  - Serial numbers (SNUMs)
  - Instruction RAM (IRAM)
- Serial DMA controller
- Clocking
  - Signal multiplexing
  - Baud-rate generation
- Dedicated interrupt 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.

19.1 Overview

Figure 19-1 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 19-1. QUICC Engine Subsystem Architectural Block Diagram

19.2 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 19.9**. 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

19.2.1 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.

19.2.2 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.

19.2.3 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 19-1 lists the default values of the parameter RAM pages base addresses assigned by the QUICC Engine subsystem to the different peripherals.

Table 19-1. 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

19.2.4 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 19-2**. 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 19-2. Buffer Descriptor Structure

In this discussion, the BD and field values use the following convention:

```
BD.field
```

Table 19-2 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 19-2. 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.

19.2.5 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 19-3** shows a high-level diagram of the multithreading architecture.

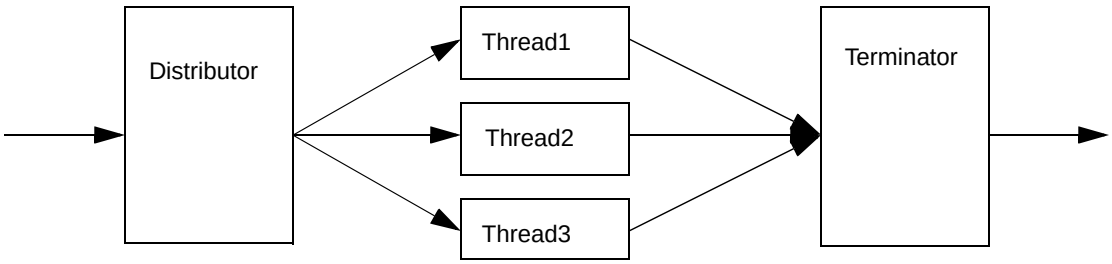


Figure 19-3. Multi-Threading Processing Mechanism

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.

19.2.6 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 19.2.5**, *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 19-3 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 19-3. 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.

19.2.7 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.

19.3 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.

19.3.1 Data Paths

Figure 19-4 is a simplified block diagram that shows the data paths in the MSC8157E. The MSC8157E may be implemented with one DDR bus (32 or 64 bit data).

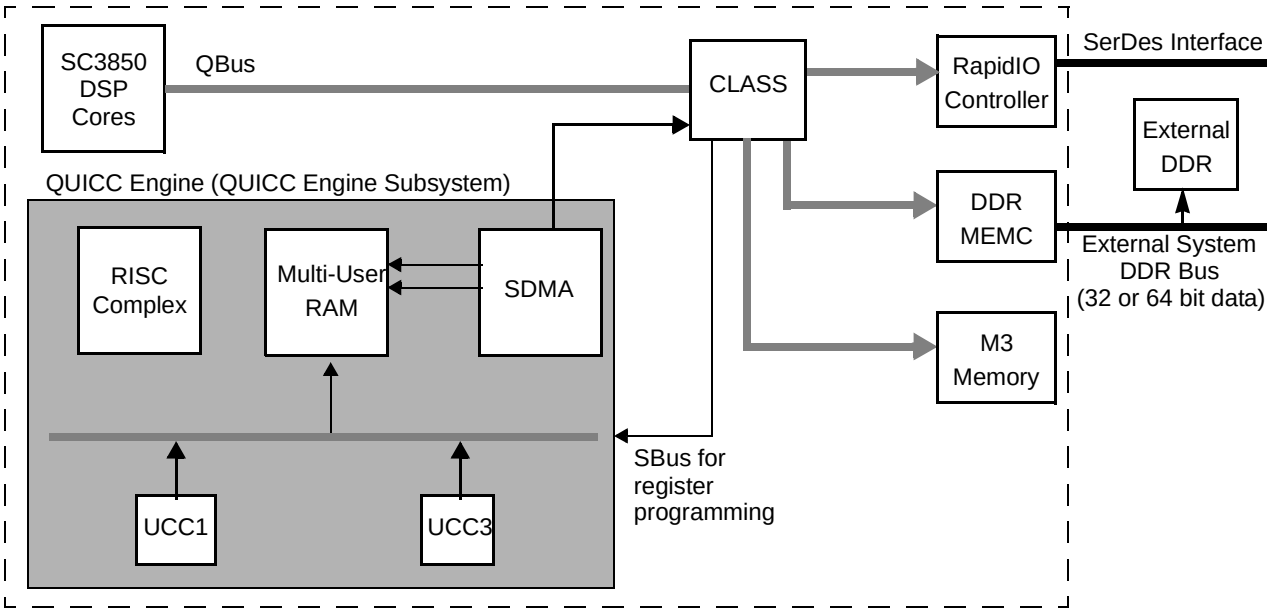


Figure 19-4. MSC8157E Data Paths

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.

19.3.2 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 19-3** 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.

19.3.2.1 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.

19.3.2.2 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.

19.3.3 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.

19.3.4 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 8.2.26**, *General Interrupt Enable Register 5 (GIER5_x)*, on page 8-44 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].

19.3.5 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.

19.4 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.

19.4.1 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 19-5** shows a block diagram of the QUICC Engine subsystem multiplexing logic.

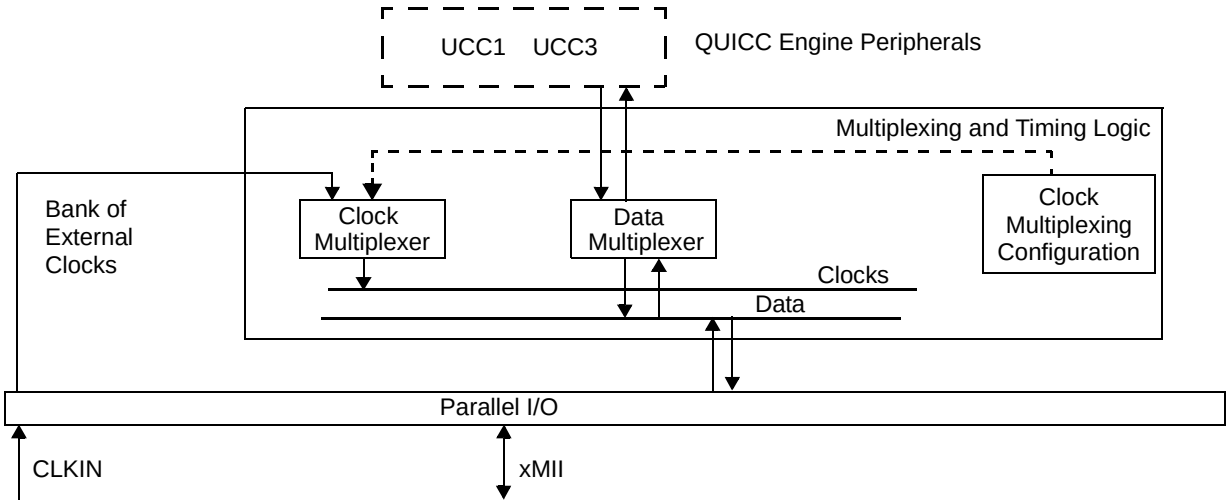


Figure 19-5. QUICC Engine Multiplexing Logic Block Diagram

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 19-6**. The clock routing options are described in **Table 19-4** and **Table 19-5**. 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.

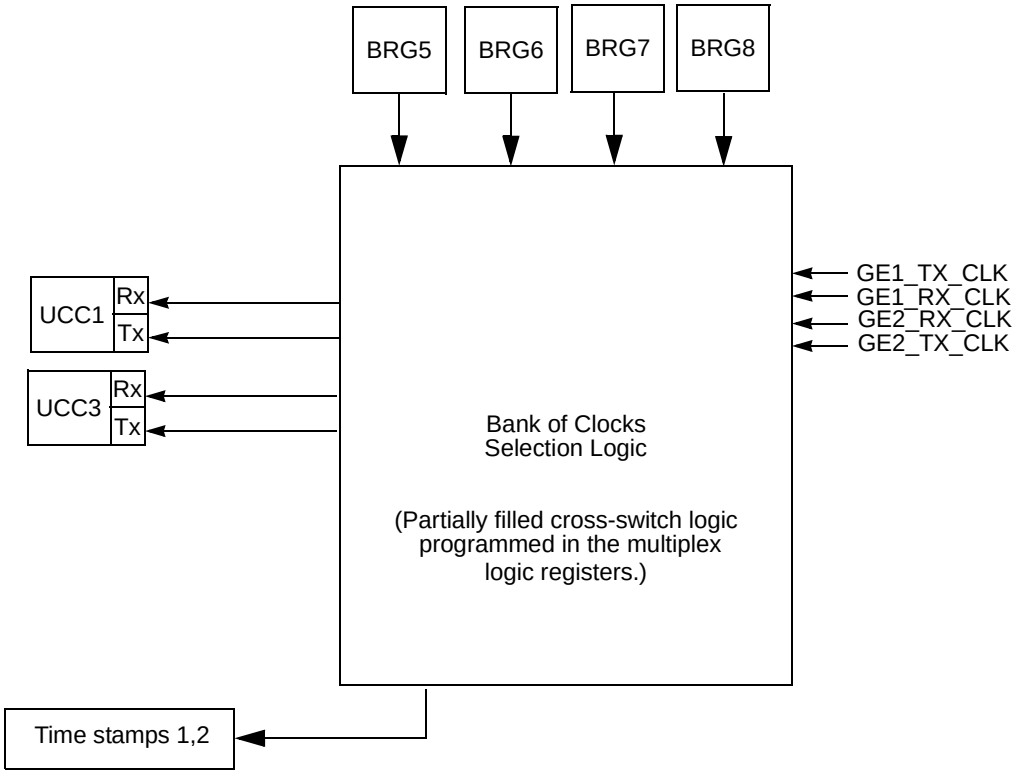


Figure 19-6. Bank of Clocks

Table 19-4. 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 19-5. 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

19.4.2 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 19-7** shows the block diagram for a BRG.

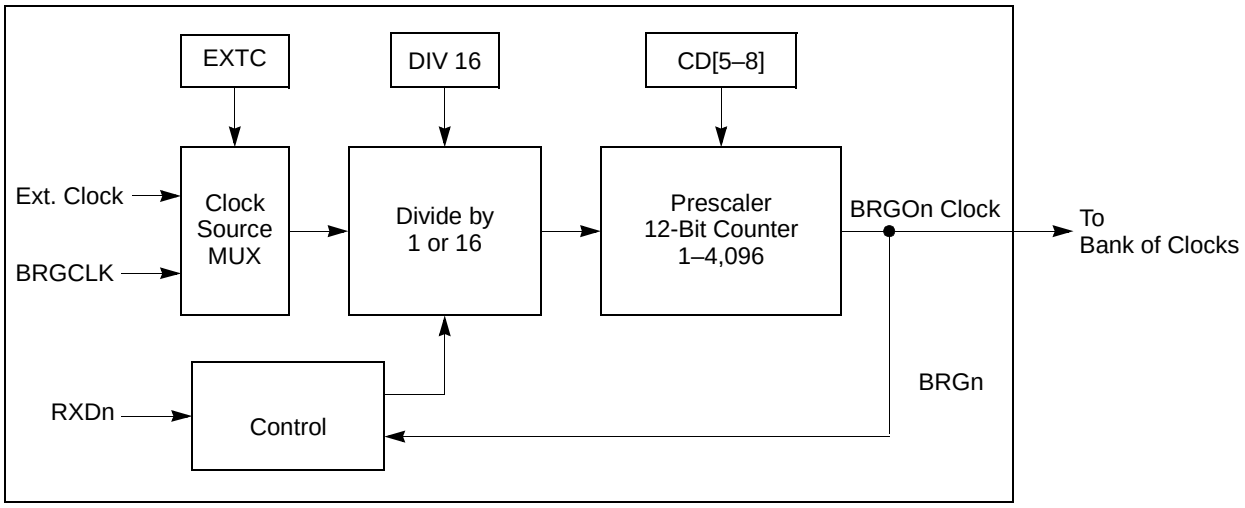


Figure 19-7. Baud-Rate Generator (BRG) Block Diagram

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.

19.5 Interrupt Controller

The QUICC Engine subsystem interrupt controller sends general interrupts to the DSP cores in the MSC8157E 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 MSC8157E uses a general configuration registers to expedite some interrupt responses:

- 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 8-35). 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 8-37).

Note: See the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM) for details about configuring the QUICC Engine interrupt system.

19.6 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.

19.6.1 UCC Functionality

The UCC block diagram is shown in **Figure 19-8**.

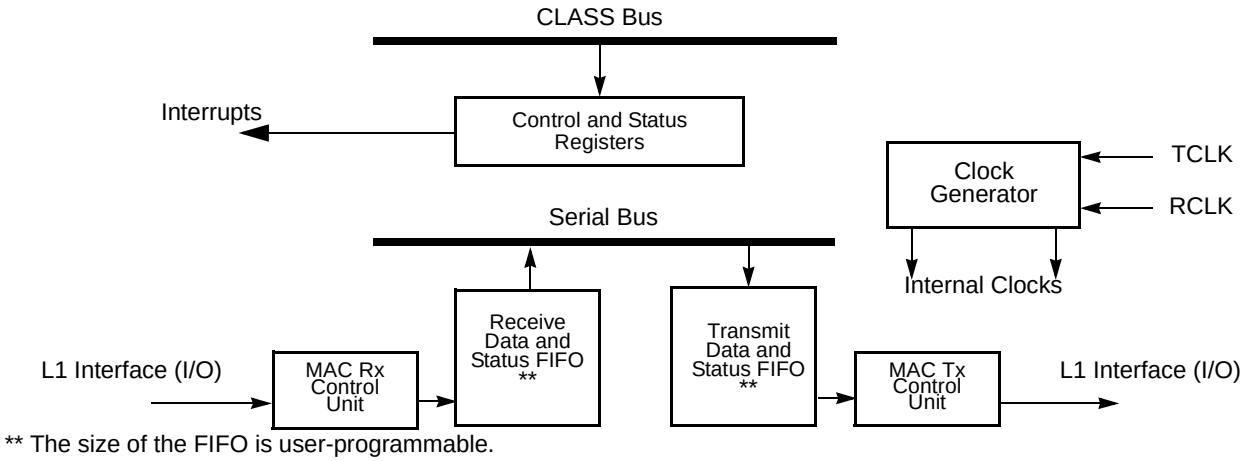


Figure 19-8. UCC Block Diagram

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 19-8** 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

19.6.2 UCC Programming Restrictions

The following sections describe programming restrictions for the UCCs.

19.6.2.1 Tx Virtual FIFO

Due to the structure of the Tx Virtual FIFO, it is possible for the Tx Virtual FIFO to contain part of a frame when there is no room for the remainder of the frame (that is, when the frame size is in the scale of magnitude of the Tx Virtual FIFO size). If the Tx Virtual FIFO contains a partial

frame, the transmission of the frame may start, even if there are fewer bytes of data than defined by the UCC Tx Virtual FIFO Threshold (UTFTT) value in the Tx Virtual FIFO. The only reason the frame might not begin to transmit is that it meets a specific sequence (which is rare, but possible) in which the UCC transmit-start condition is not met for the frame. The erroneous behavior is a result of incorrect recovery after the pausing data-retrieve phase from the BD buffer to the Tx Virtual FIFO in a certain internal stage. Reaching the UTFTT watermark in the Tx Virtual FIFO for each transmit frame prevents this sequence. UTFTT has an upper limit value that is less than the UCC Tx Virtual FIFO Size (UTFS) value. For each UTFS value, there is a maximum UTFTT value that corresponds to it. Configuring UTFTT higher than the suggested value can result in a UCC halt and Ethernet transmission stops.

Note: UTFTT refers to the payload of a single Ethernet frame. Transmission may start (this is the correct behavior) before surpassing the threshold if there is no space for additional data in the Tx Virtual FIFO

The UCC may stop transmitting when a large frame is partially stored in the Tx Virtual FIFO before the transmission of the frame starts. Note that the Tx BD ring buffer configuration (data allocation to the BD buffers) may adversely affect Tx Virtual FIFO usage, thus preventing the Tx Virtual FIFO from reaching the threshold for transmission in the Tx Virtual FIFO.

Use one of the following options to avoid this condition:

1. To prevent the failure, configure UTFS and UTFTT so that it is possible to store UTFTT data (payload) bytes in the Tx Virtual FIFO.

   For a single buffer per frame, the maximum allowable UTFTT values are:
   Ethernet Controller: Set UTFTT < [(0.9375 × UTFS) – 128]
   For example, for an Ethernet controller with UTFS = 1024, set UTFTT < 832.

   For multiple buffers per frame, the maximum allowable UTFTT values are:
   Ethernet Controller: Set UTFTT < [(UTFS × (M – 8)/M –128]
   For example, for an Ethernet controller with UTFS = 1024 and M = 64, set UTFFT < 768.

2. To recover from a possible failure, the application code should configure UTFTT to 0x40 and then restore it to its original value. The sequence triggers transmission from the point at which it stopped. Detection of such a UCC halt is done by monitoring Tx BD vacancy. A full Tx BD ring may indicate that the UCC has halted.

19.6.2.2 Multi-Threading Configuration

The Ethernet receiver might become stuck while in a multi-threaded configuration in case of heavy traffic. This condition is influenced by both the VFIFO size and the number of threads that are enabled.

Use at least one of the following measures to avoid the situation described above:

- Enable only one Rx thread.
- The Rx VFIFO size should be equal to 0.5 Kbytes.
- If the Rx VFIFO size is between 0.5 to 1.1 Kbytes, at least 4 threads must be enabled.
- If the Rx VFIFO size is between 1.1 to 1.6 Kbytes, at least 6 threads must be enabled.
- If the Rx VFIFO size is between 1.6 to 2.2 Kbytes, at least 8 threads must be enabled.
- If the Rx VFIFO size is between 2.2 to 4.5 Kbytes, the user must allocate 512 bytes (must be initialized to zero) in the Multiuser RAM. The base address for this area must be 512 bytes aligned. Immediately after the Ethernet Rx INIT command is ended (and before the UCC receiver is enabled), this address (24 bits) must be written to the Rx GPRAM in offsets 0x09–0x0b and 0x11–0x13. The user must write the value 0x3F to offsets 0x08 and 0x10. For example, if the base address for this new structure is 0x123400, the Rx GPRAM[0x08–0x0b and 0x10–0x13] will be equal to 0x3F123400. Note: In this case, there is no limitation to the number of threads that must be enabled.

19.7 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 MSC8157E 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 MSC8157E 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 19-9 illustrates the block diagram of the UCC Ethernet controller.

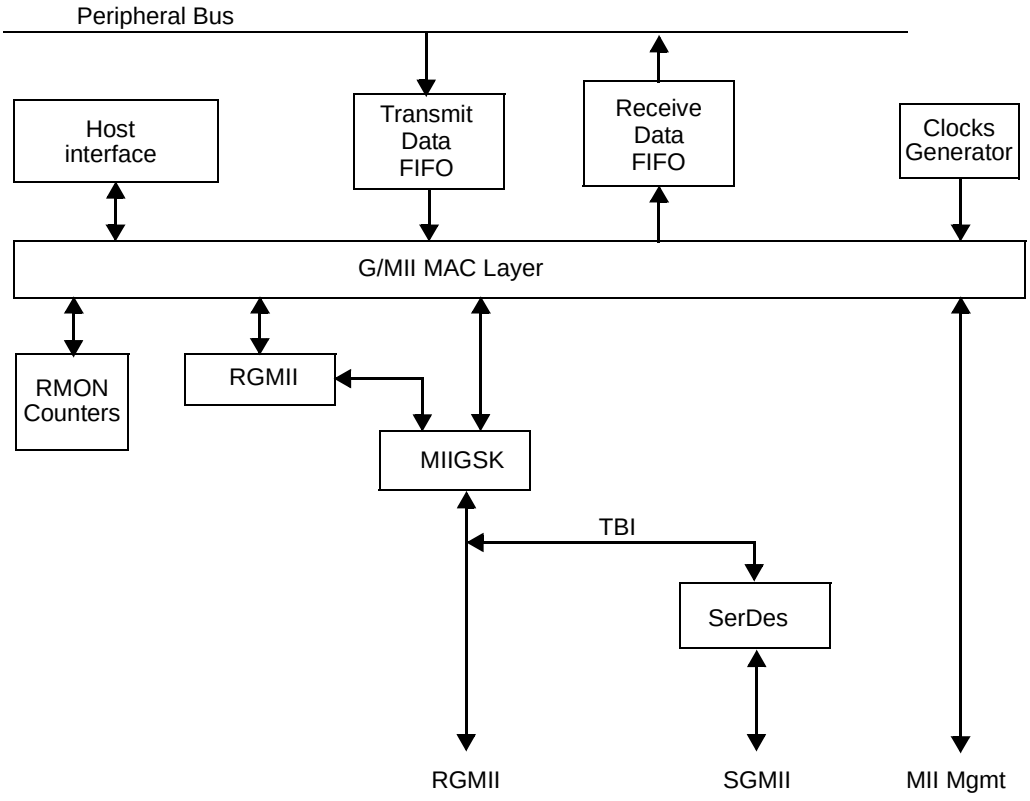


Figure 19-9. UCC Ethernet Controller Block Diagram

19.7.1 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).

19.7.1.1 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.

19.7.1.2 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 MSC8157E, 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.

19.7.2 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 19-6 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 19-6. 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.

19.7.2.1 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 19-10** shows the RGMII connections between the Ethernet controller and a PHY.

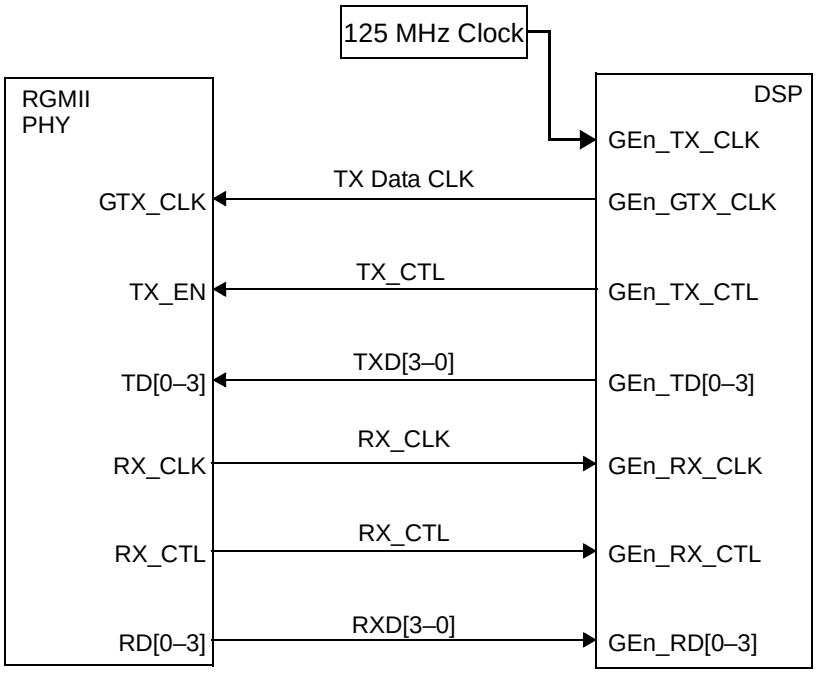


Figure 19-10. RGMII MAC-PHY Interface

19.7.2.1.1 RGMII Signals

Table 19-7 lists the RGMII signals.

Table 19-7. 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 19-7. 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	—

19.7.2.1.2 RGMII Signal Configuration

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 MSC8157E 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 MSC8157E data sheet. Guidelines for adjusting these numbers in individual designs is provided in *Using GCR4 to Adjust Ethernet Timing in MSC8157 DSPs* (AN4134).

19.7.2.2 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.

19.7.2.2.1 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 19-8** lists the SGMII signals.

Table 19-8. SGMII Signals

Signal Name	I/O	Size	Function	Reference Clock
SRIO_REF_CLK SRIO_REF_CLK	I I	2	**Reference Clock** 125 MHz differential pair.	—
SG1_TX SG1_TX	O O	2	**Transmit Data 1** Differential pair for Ethernet 1 controller.	SRIO_REF_CLK SRIO_REF_CLK
SG2_TX SG2_TX	O O	2	**Transmit Data 2** Differential pair for Ethernet 2 controller.	SRIO_REF_CLK SRIO_REF_CLK
SG1_RX SG1_RX	I I	2	**Receive Data 1** Differential pair for Ethernet 1 controller.	SRIO_REF_CLK SRIO_REF_CLK
SG2_RX SG2_RX	I I	2	**Receive Data 2** Differential pair for Ethernet 2 controller.	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).	—

19.7.2.2.2 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 5-16 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).

19.7.3 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.

19.7.4 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 19-9 lists the minimum steps required for register and parameter initialization

Table 19-9. 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	
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.

19.7.5 Ethernet Programming Restrictions

The Ethernet implementation in the MSC8157E has the programming restrictions defined in the following sections.

19.7.5.1 RMON Statistics

Because there are a number of conditions that can cause the statistics to be inaccurate, do not use the TX CRC counter values for statistics.

19.7.5.2 Unreported Overrun

An Overrun status for a received frame is a notification that is was not fully received because the Ethernet controller is temporarily overloaded and that the frame should be discarded. In rare cases, such as when two sequential frames that cause an Overrun are received, the second frame is discarded but not reported as causing an overrun. The frame loss can be detected by a higher level protocol (such as TCP/IP). Make sure that the software requests a retransmission of the frame if the higher level protocol reports a frame loss.

19.7.5.3 Broadcast Status after an In-Band CRS Event

In RGMII mode, the Ethernet can erroneously indicate a Broadcast status for the previous frame (wrong RxBD[BC]) is generated after an in-band CRS event. To make sure a real broadcast event has occurred, program the application to ignore the RxBD[BC] bit and check the Broadcast address instead.

19.7.5.4 Pause Frame End

A pause frame is used to restrict flow control when the Ethernet controller is being overloaded. To end the pause, the application sends an XON receive pause frame with a Pause Time Value (PTV) = 0, which should cause the transmitter to exit the pause frame immediately. However, in the MSC8157E, the controller decrements that counter after clearing it before performing the compare to zero. As a result, the XON actually causes the transmitter to continue in the pause

state for 65535 pause quantas (3353920 bit-times). To prevent this occurrence, use a pause frame with a PTV = 1 to force an exit from a pause state.

Note: In the case of connecting to another QUICC Engine device, the programmer may not use the QUICC Engine Automatic Flow Control (AUFC) because in this mode, the Ethernet Controller generates Pause Commands with Pause Time Value = 0 in order to exit pause state. The programmer must configure the QUICC Engine block to use either of the following flow control modes:

— Lossless Flow Control: The Ethernet controller always uses the value of the UEMPR[PT] register when generating pause frames. It never automatically generates a pause frame with a pause time value of 0 when the receiver recovers from being above the RxFIFO threshold or below the free RxBDs threshold.

— Host Enabled Flow Control: Use the following guidelines for the specified QUICC Engine commands:

- START FLOW CONTROL. Sends pause frames with the Pause Time Value in the UEMPR[PT] field. The programmer may configure UEMPR[PT] for pause time and use the START FLOW CONTROL command for sending pause frames For exiting pause state, the programmer may configure UEMPR[PT] to 1 and use the START FLOW CONTROL command.
- STOP FLOW CONTROL. Sends a Pause Time Value of 0 and should be avoided.

19.7.5.5 Pause Frame Time

If a transmit is in progress, the pause time may be shorter than specified. When the Ethernet controller receives a pause frame with PTV not equal to 0 and MACCFG1[Rx Flow] = 1, it completes transmitting any current frame in progress; then it should pause for PTV $\times$ 512 bit-times. The MAC, however, does not take the full transmission time of the current frame into account when calculating the Tx pause time, and it may pause for 1–2 pause quanta (512–1024 bit-times) less than the PTV value. This can cause the Ethernet controller to pause transmission for up to 1024 bit-times less than the pause frame request value. If the PTV does not contain at least 2 pause quanta worth of margin, it may allow receive buffer overflows to occur in the link partner. Because the transmit pause does not take effect until after the current frame completes transmitting, the link partner pause frame generator must already include the maximum frame size as margin when calculating the pause time value to use to prevent overflow of the receiver buffers. To prevent this condition, always add 2 pause quanta to the pause time value when generating pause frames to prevent buffer overflow.

19.7.5.6 Transmit During Pause Frame

After receiving a PAUSE frame, the UCC Ethernet controller may transmit a frame although it has entered the PAUSE state. As a consequence, the pause duration is shortened by the length of the transmitted frame. The UCCS[BPR] and UCCE[CBPR] bits reflect the PAUSE state (A

PAUSE frame with Pause Time Value of zero ends the PAUSE state). A frame may be transmitted on the line during this PAUSE window. If this transmitted frame does not consume the entire PAUSE window, the following frame is suspended until the pause duration is over and only then it will be transmitted. If the short PAUSE frame can impact system performance, consider increasing the PAUSE frame time. The increased Pause frame should large enough so that the minimum pause requirement is always met. Make sure that by enlarging the PAUSE frame, all the PAUSEs are affected, not just the short ones.

19.7.5.7 Magic Packet Handling

The Ethernet MAC should recognize Magic Packet sequences by identifying an Ethernet frame containing a valid Ethernet header (Destination and Source Addresses) and valid FCS (CRC-32), and whose payload includes the specific Magic Packet byte sequence at any offset from the start of data payload. The specific byte sequence comprises an unbroken stream of 102 bytes, the first 6 bytes of which are 0xFFs, followed by 16 copies of the MACs unique IEEE station address in the normal byte order for Ethernet addresses.

The following are example partial sequences followed by the start of a complete sequence for station address 01_02_03_04_05_06:

- FF_FF_FF_FF_FF_FF_FF_01_02_03_04_05_06_01...
  Seventh byte of 0xFF does not match next expected byte of Magic Packet sequence (01). Pattern search restarts looking for 6 bytes of FF at byte 01.
- FF_FF_FF_FF_FF_FF_01_FF_FF_FF_FF_FF_FF_01_02_03_04_05_06_01...
  First FF byte following 01 does not match Magic Packet sequence. Pattern search restarts looking for 6 bytes of FF at second byte of FF following 01.

The following is an example partial sequence followed by the start of a complete sequence which is erroneously not recognized for station address 01_02_03_04_FF_06:

- FF_FF_FF_FF_FF_FF_01_02_03_04_FF_FF_FF_FF_FF_FF_01_<complete sequence>
  11th byte (0xFF) is seen as the 11 byte of the partial pattern and is not recognized as the start of a complete sequence. Pattern search restarts looking for 6 bytes of 0xFF at 12th byte, but sees only 5.

19.7.5.7.1 Failure to Exit Magic Packet Mode

If a complete Magic Packet sequence (including 6 byes of 0xFF) immediately follows a partial Magic Packet sequence, however, the complete sequence is not recognized and the MAC does not exit Magic Packet mode.

To prevent this occurrence, place one byte of data that is not 0xFF and does not match any bytes of Destination Address before the start of the Magic Packet sequence in the frame. Because the Magic Packet sequence pattern search starts at the 3rd byte after Destination Address, the Magic

Packet sequence can be placed at the start of the data payload as long as the second byte of the length/type field follows the placement rule.

19.7.5.7.2 Malformed Magic Packet Mode Triggers Exit

The following describes these scenarios:

1. Any Ethernet frame containing a valid Ethernet header (Destination and Source Addresses) and valid FCS (CRC-32), and whose payload includes the specific Magic Packet byte sequence at any offset from the start of data payload. The specific byte sequence comprises an unbroken stream of 102 bytes, the first 6 bytes of which are 0xFFs, followed by 16 copies of the MACs unique IEEE station address in the normal byte order for Ethernet addresses.
2. Once the Ethernet MAC has recognized a valid Destination Address for one frame, it continues searching for valid 102-byte Magic Packet sequences through multiple frames without checking for a valid Destination Address on each frame. The only events that cause the MAC to go back to check for valid Destination Address before checking for a Magic Packet sequence on new frames are:
   - A frame containing a recognized full Magic Packet sequence (with valid or invalid FCS)
   - Software disable of Magic packet mode (MACCFG2[MPE] = 0)
   - Perform soft reset via MACCFG1[Rx_EN] and MACCFG1[Tx_EN].
3. The Ethernet controller may exit Magic Packet mode if it receives a frame with Destination Address not matching station address, or invalid unicast or broadcast address in a valid Magic Packet sequence for the device.

If an erroneous recognition of a Magic Packet can affect system functionality, avoid the use of Magic Packets.

19.7.5.8 Ethernet Transmit Scheduler

If the data in the buffer is all 00s (bmax) or FFs (bmin), it results in an incorrect offset/address result, causing the Ethernet scheduler to select an arbitrary queue. To avoid incorrect behavior of the Ethernet Tx scheduler (including a possible stuck condition), always us all even values for the Weight Factor in the scheduling parameters.

19.7.5.9 Initialization of SGMII Mode

A key function of the TBI block in the 1 Gbps MAC is link initialization. The link initialization function does not always properly detect when the link is synchronized. As a result, in some instances, the link initialization may not complete (Link Status remains 0).

Note: The MSC8157E does not support auto negotiation (AN) in SGMII mode.

Use the following sequence prior to the clearing of IEVENT to ensure correct link initialization (this assumes that the SerDes PLL has already completed its reset sequence and the SerDes clocks are running):

1. Perform an MII Mgmt write cycle to TBI Control Register.

   MIIMCON[0000_0000_0000_0000_0001_0000_0000_0000]

   Set AN enable. A change in value of AN enable performs a reset of the AN function of the TBI. Although the MSC8157E does not support AN functionality, the AN bit is used to reset the initialization state machine internally.

2. Check to see if MII Mgmt write is complete.

   Read MII Mgmt Indicator register and check for Busy = 0,
   MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000]

   This indicates that the write cycle was completed.

3. Perform an MII Mgmt read cycle of Control Register.

   Clear MIIMCOM[Read Cycle]

   Set MIIMCOM[Read Cycle]

4. Check to see if MII Mgmt read is complete.

   Read MII Mgmt Indicator register and check for Busy = 0,
   MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000]

   This indicates that the read cycle was completed.

5. Wait for 100 µs

6. Perform an MII Mgmt write cycle to TBI Control Register.

   Clear MIIMCOM[Read Cycle]

   MIIMCON[0000_0000_0000_0000_0000_0001_0100_0000]

   This tells the TBI to restart link initialization and start operating with auto-negotiation disabled.

7. Check to see if MII Mgmt write is complete.

   Read MII Mgmt Indicator register and check for Busy = 0,
   MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000]

   This indicates that the write cycle was completed.

8. Perform an MII Mgmt read cycle of Control Register.

   Clear MIIMCOM[Read Cycle]

   Set MIIMCOM[Read Cycle]

9. Check to see if MII Mgmt read is complete.

Read MII Mgmt Indicator register and check for Busy = 0,
MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000]

This indicates that the read cycle was completed.

10. Continue with the remainder of the SGMII initialization sequence.

19.8 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 19-11**, 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.

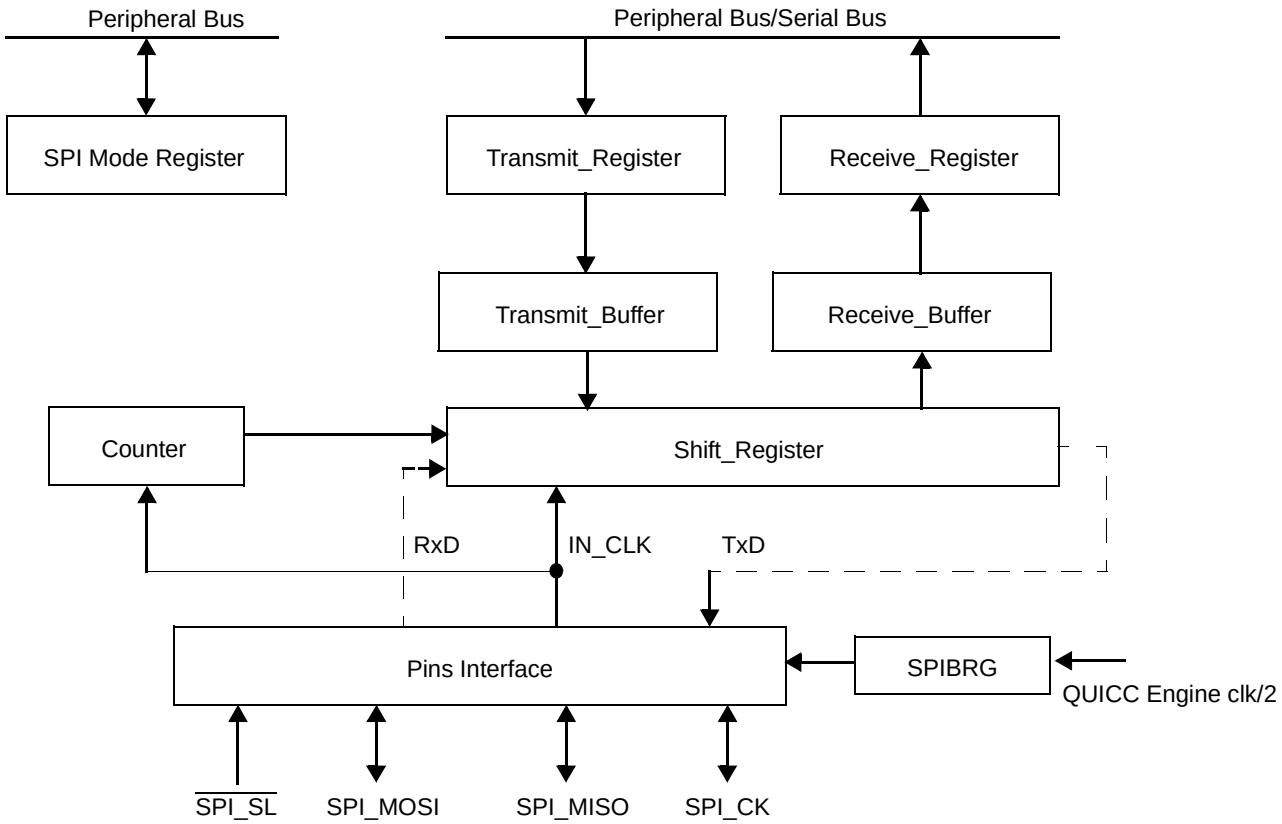


Figure 19-11. SPI Block Diagram

19.8.1 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.

19.8.1.1 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 19-12**. 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.

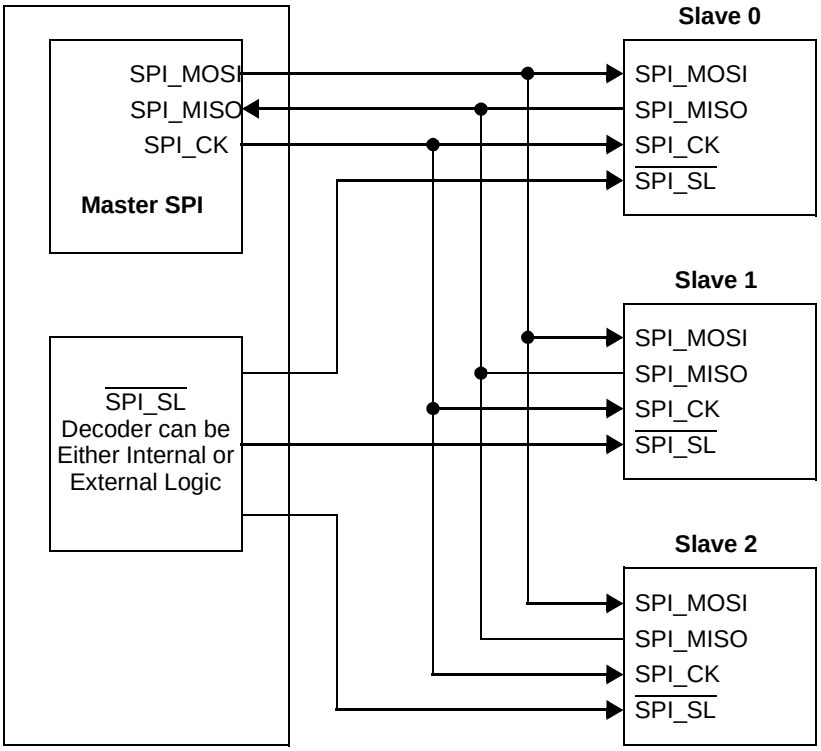


Figure 19-12. Single-Master/Multi-Slave Configuration

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.

19.8.1.2 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.

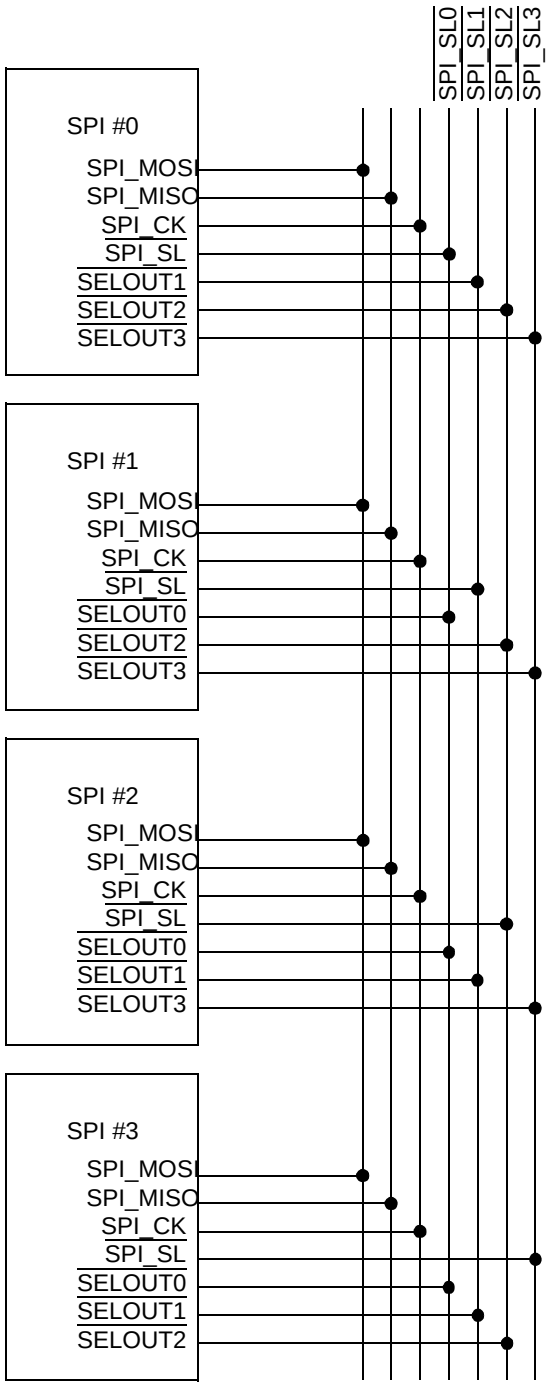
Transmission continues until no more data is available or $\overline{\text{SPI\_SL}}$ is deasserted. If it is deasserted before all data is sent, it stops but the TxBD stays open. Transmission continues once $\overline{\text{SPI\_SL}}$ is reasserted and SPI_CK begins toggling. After the characters in 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.

19.8.2 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 19-13**. 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 19-13. Multimaster Configuration

19.8.3 External Signal Configuration

The SPI supports a four-wire interface—transmit, receive, clock, and slave select. See **Table 3-1** 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 3-10** *SPI Signals* on page 3-18 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.

19.8.4 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.

19.9 Programming Model

This section provides a summary list of the MSC8157E QUICC Engine subsystem, Ethernet controller, and SPI registers with their offsets.

Note: The QUICC Engine registers use a base address of 0xFEE00000.

Table 19-10. MSC8157E 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 19-10. MSC8157E 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 19-10. MSC8157E 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 19-10. MSC8157E 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 19-10. MSC8157E 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

20 UART

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 21**, *GPIO* for details).

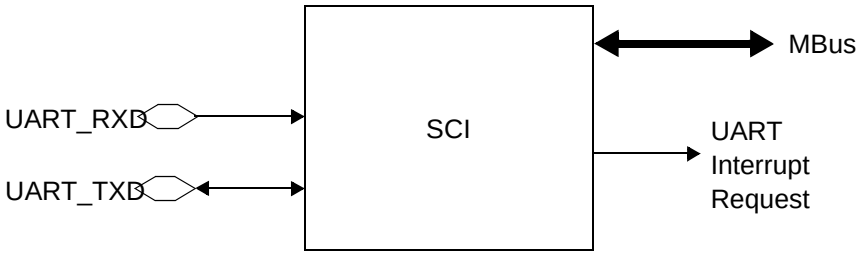


Figure 20-1. UART Interface

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 MSC8157E and remote devices, including other MSC8157E 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.

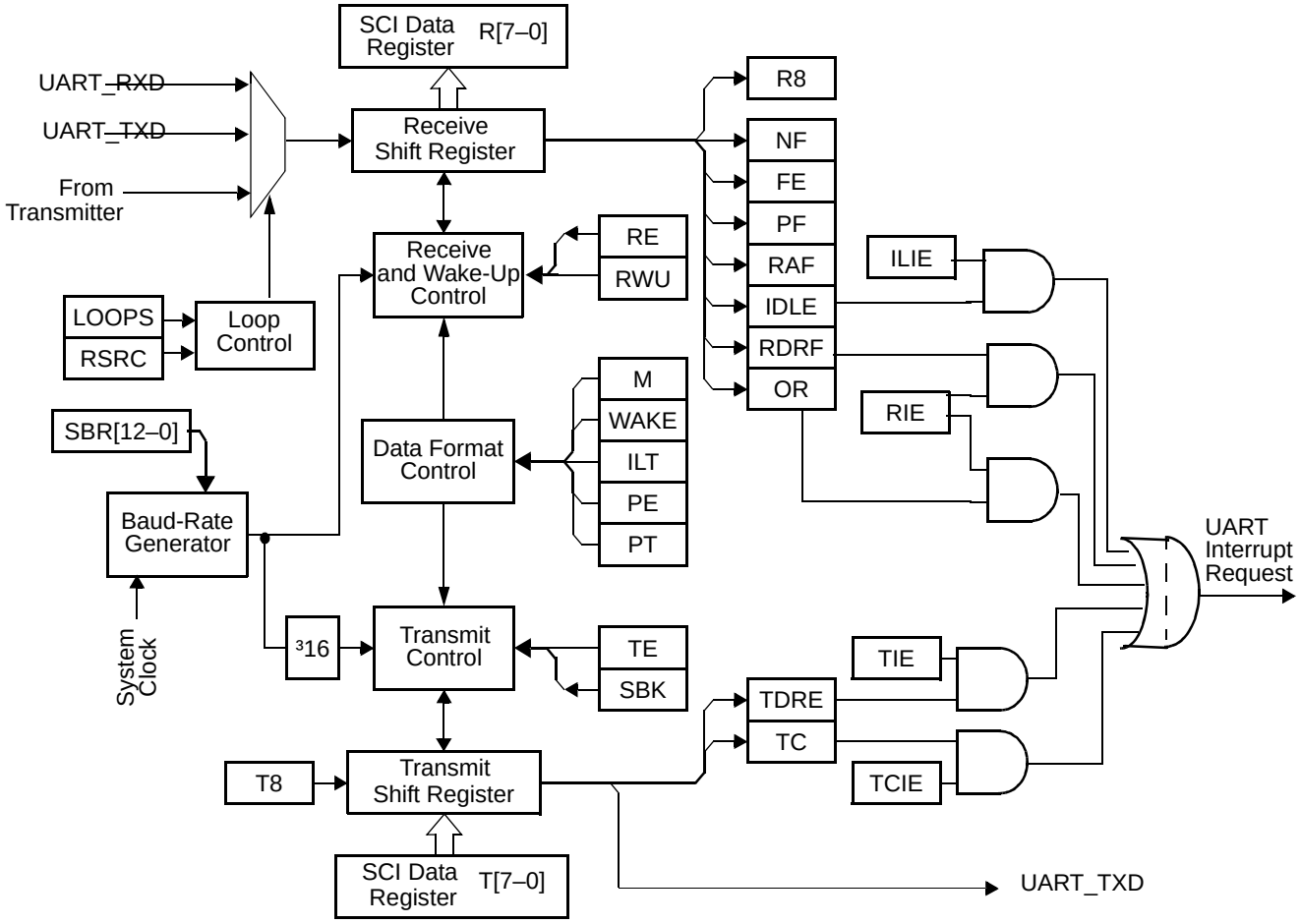
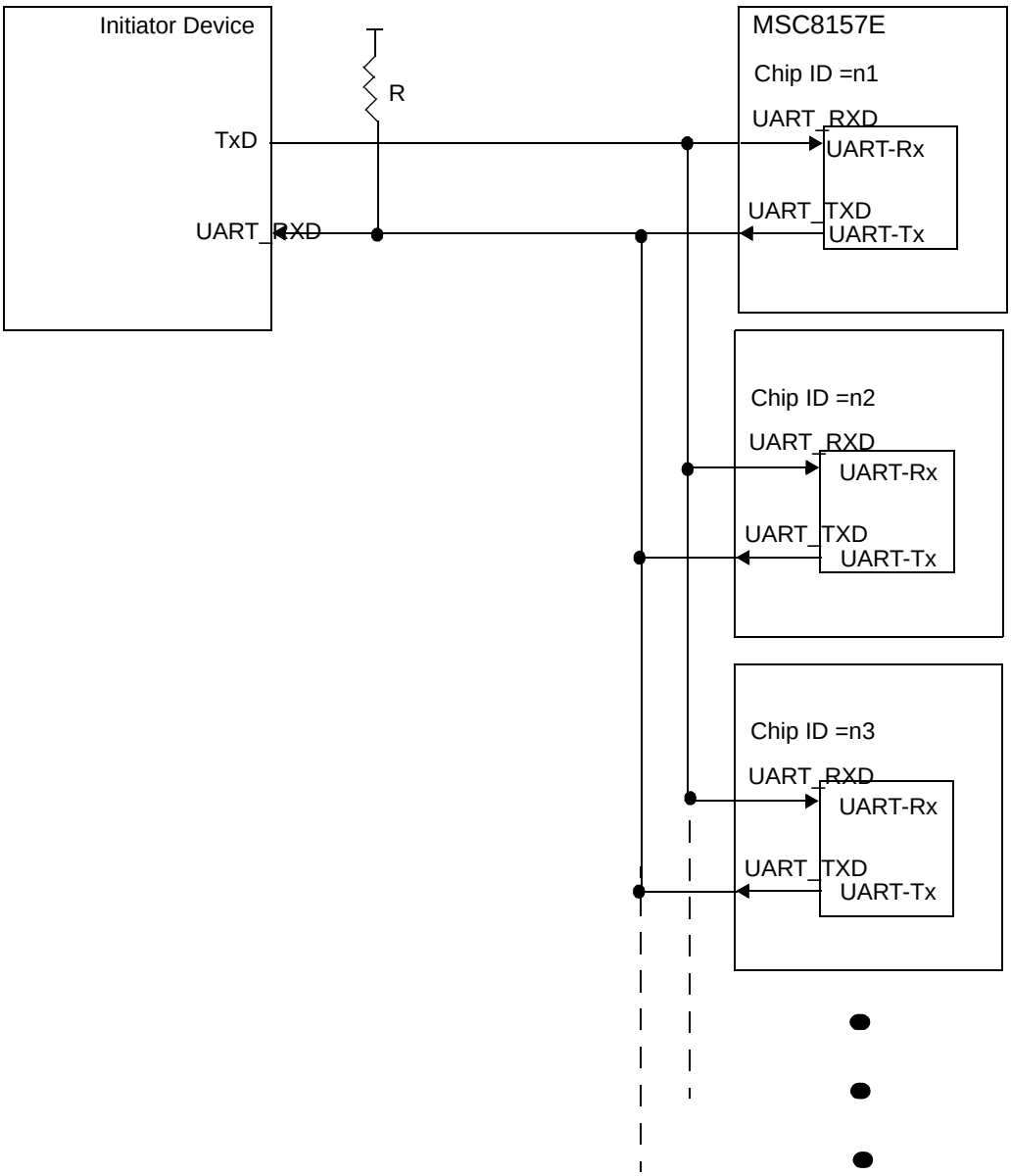


Figure 20-2. UART Block Diagram

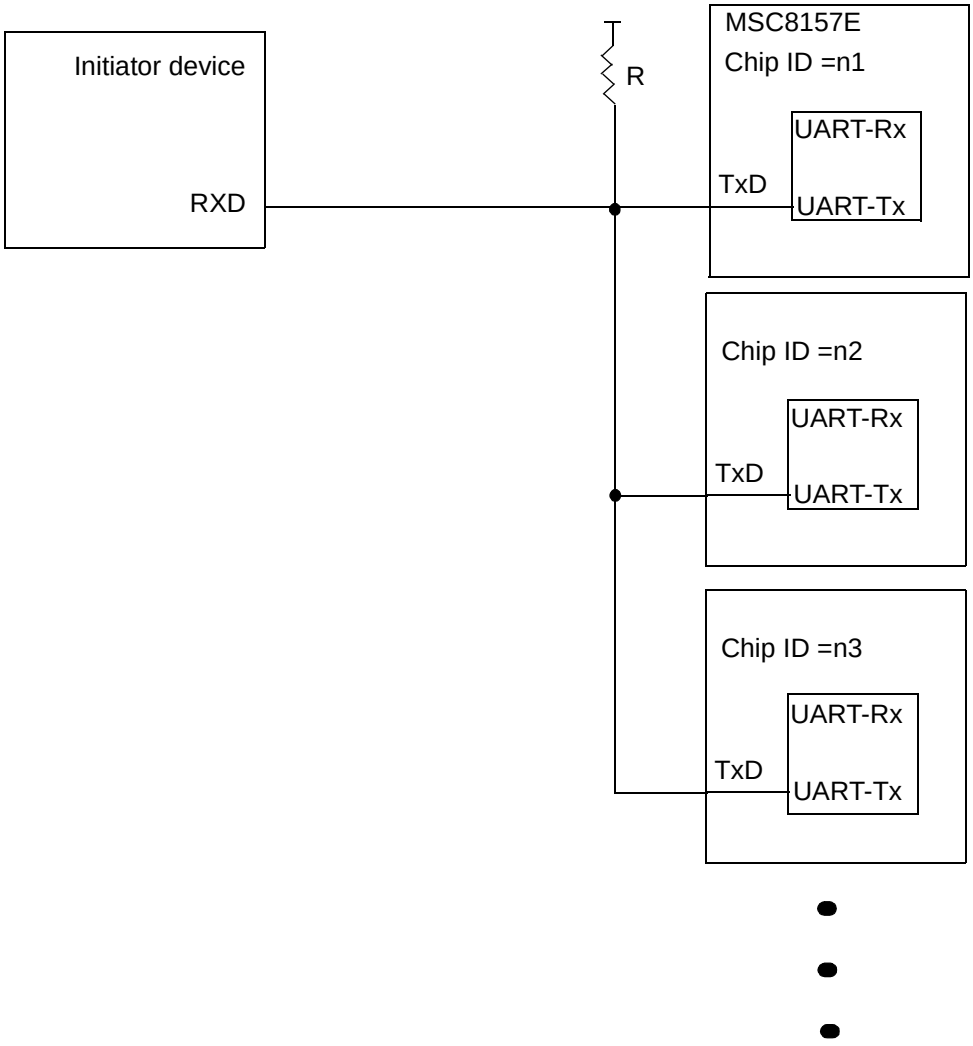
Figure 20-3 shows the full duplex UART system in which the MSC8157E UART transmits and receives simultaneously. A higher-level protocol should handle the full duplex communication to guarantee that no more than one target UART transmits to the UART_RXD signal of the initiator at a given time. Receiver 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 21**, *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 21**, *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**.

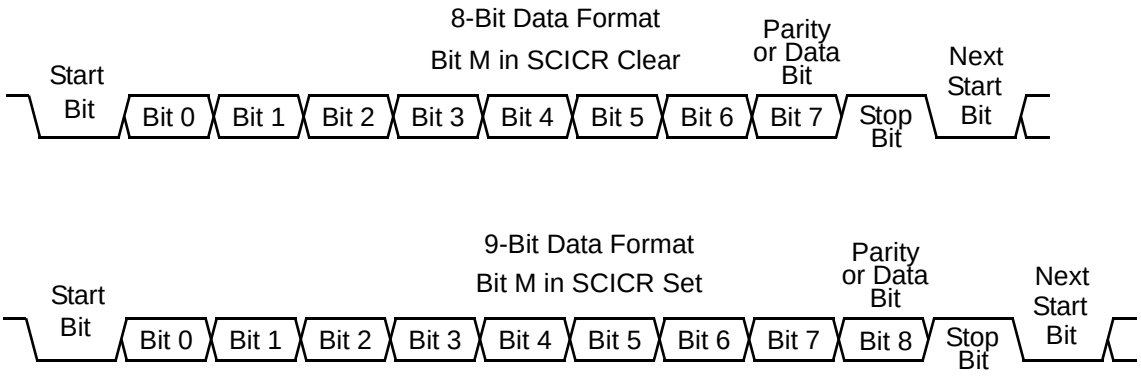


Figure 20-5. UART Data Formats

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 System Peripherals 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 System Peripherals 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 System Peripherals 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 (System Peripherals 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. These numbers are based on a CLASS frequency of 500 MHz, but the frequency can be configured 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).

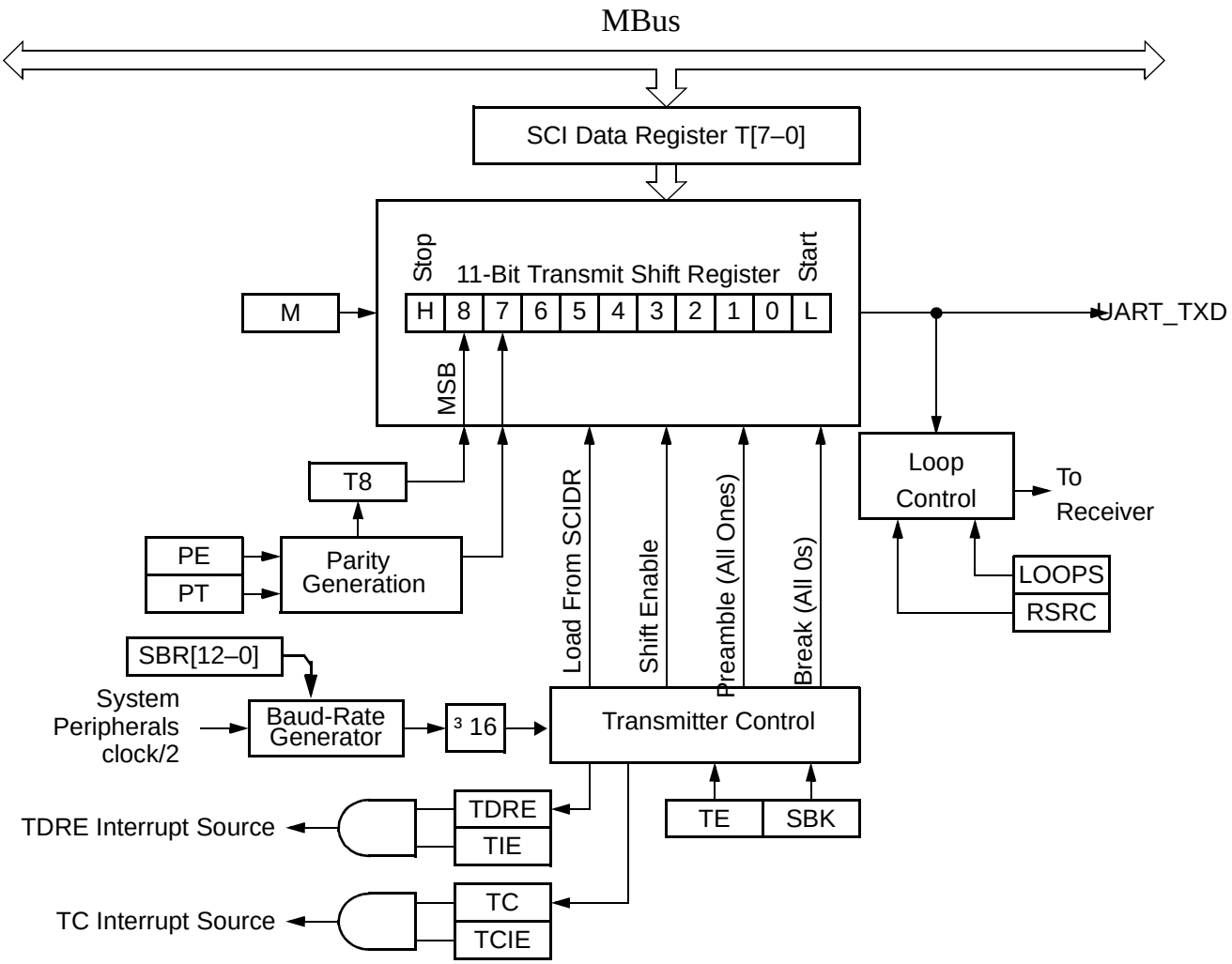


Figure 20-6. Transmitter Block Diagram

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 21**, *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 21**, *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*.

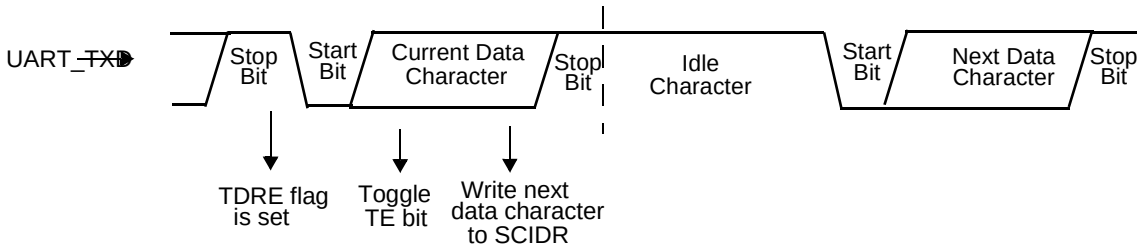


Figure 20-7. Queuing an Idle Character

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 21**, *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.

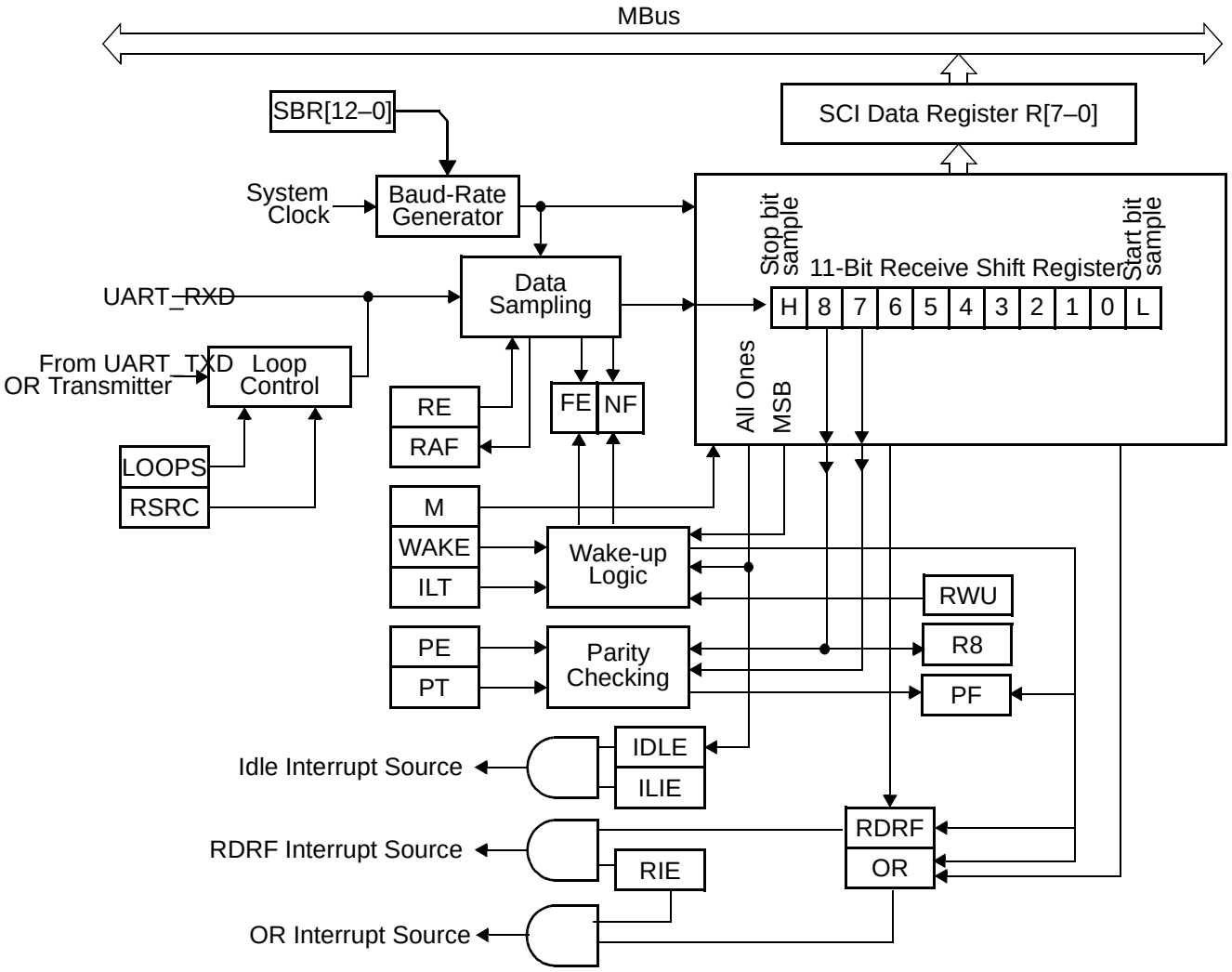


Figure 20-8. UART Receiver Block Diagram

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.

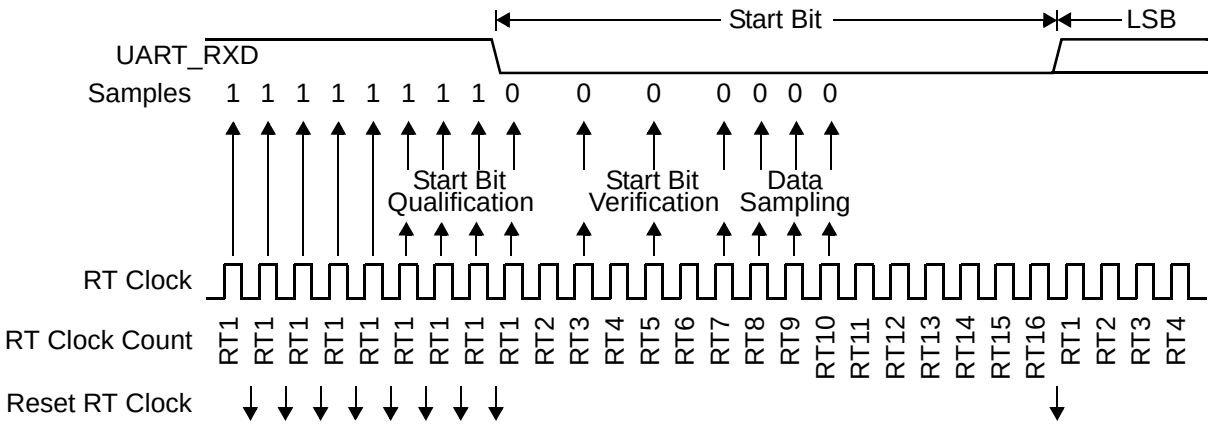


Figure 20-9. Receiver Data Sampling

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.

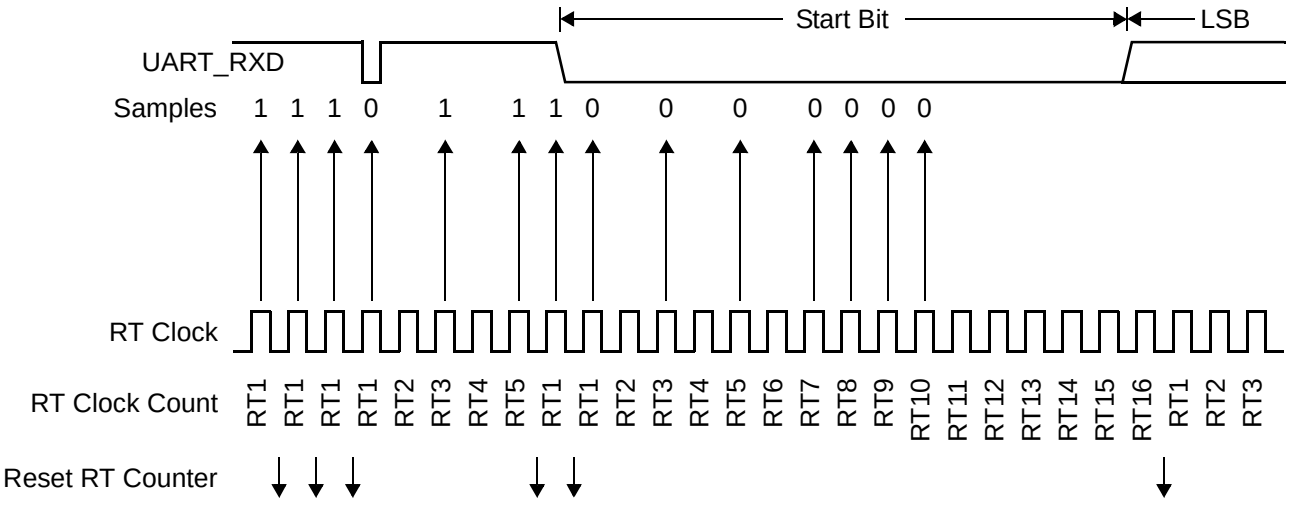


Figure 20-10. Start Bit Search Example 1

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.

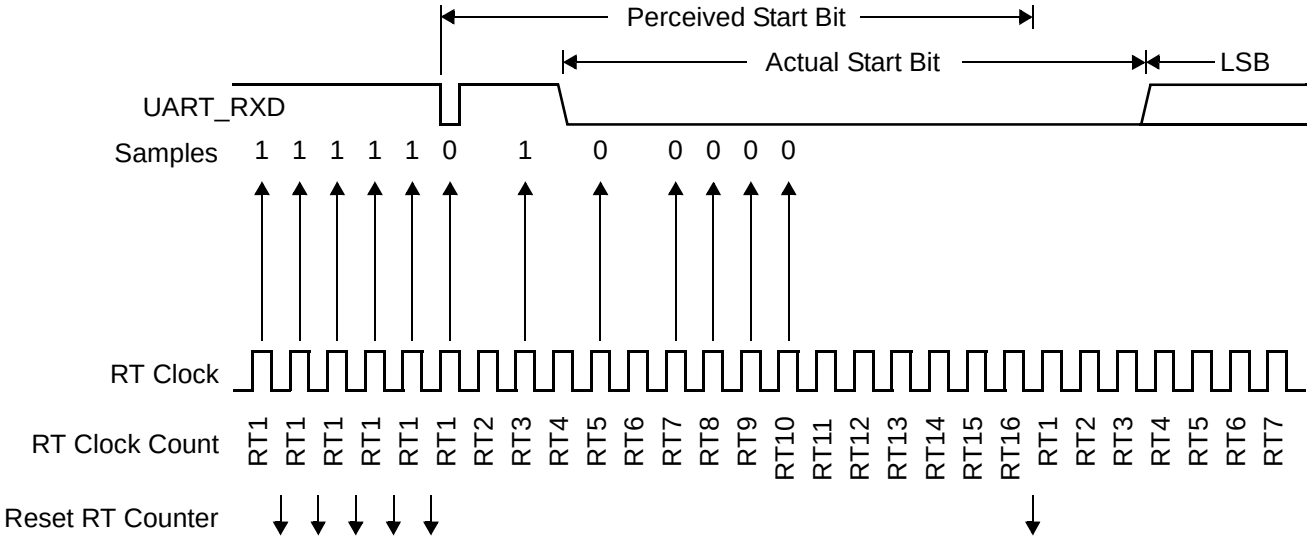


Figure 20-11. Start Bit Search Example 2

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.

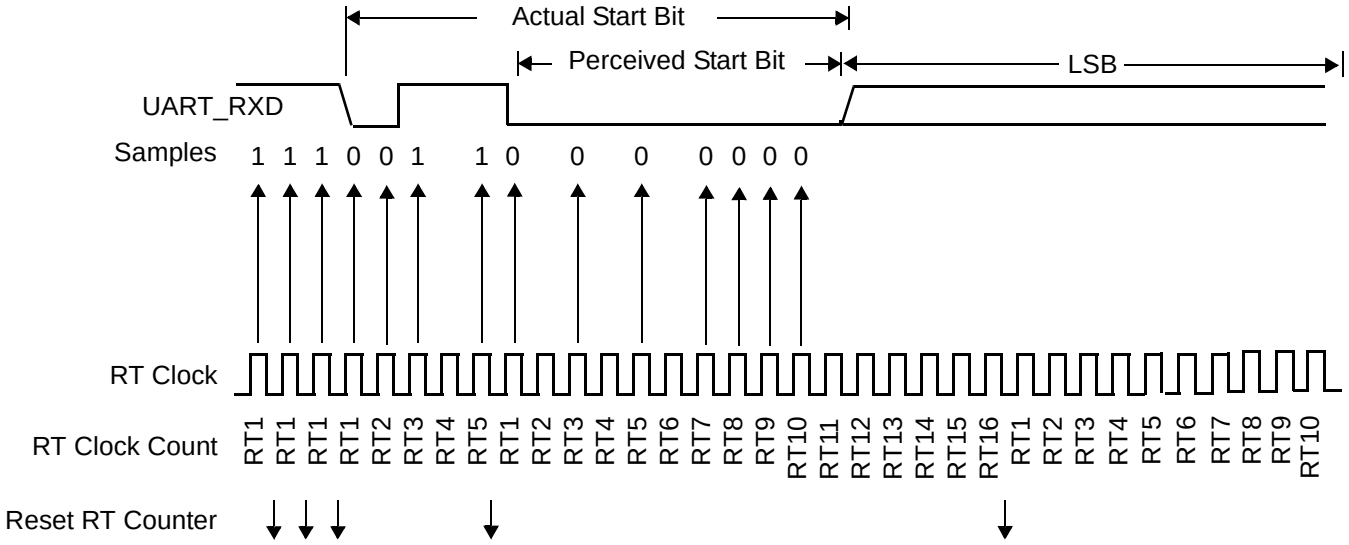


Figure 20-12. Start Bit Search Example 3

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.

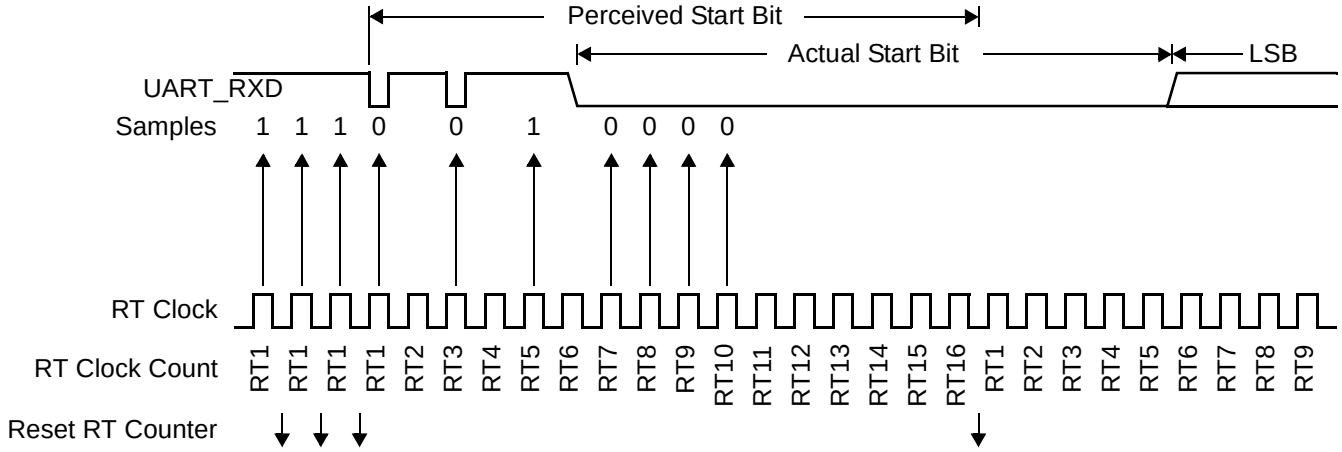


Figure 20-13. Start Bit Search Example 4

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.

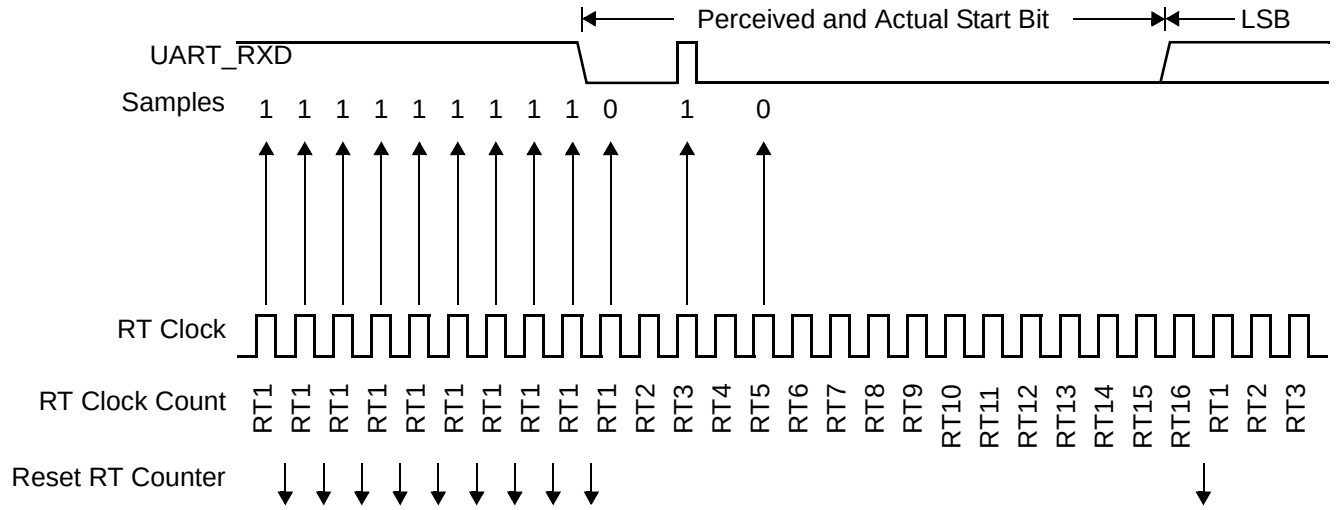


Figure 20-14. Start Bit Search Example 5

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.

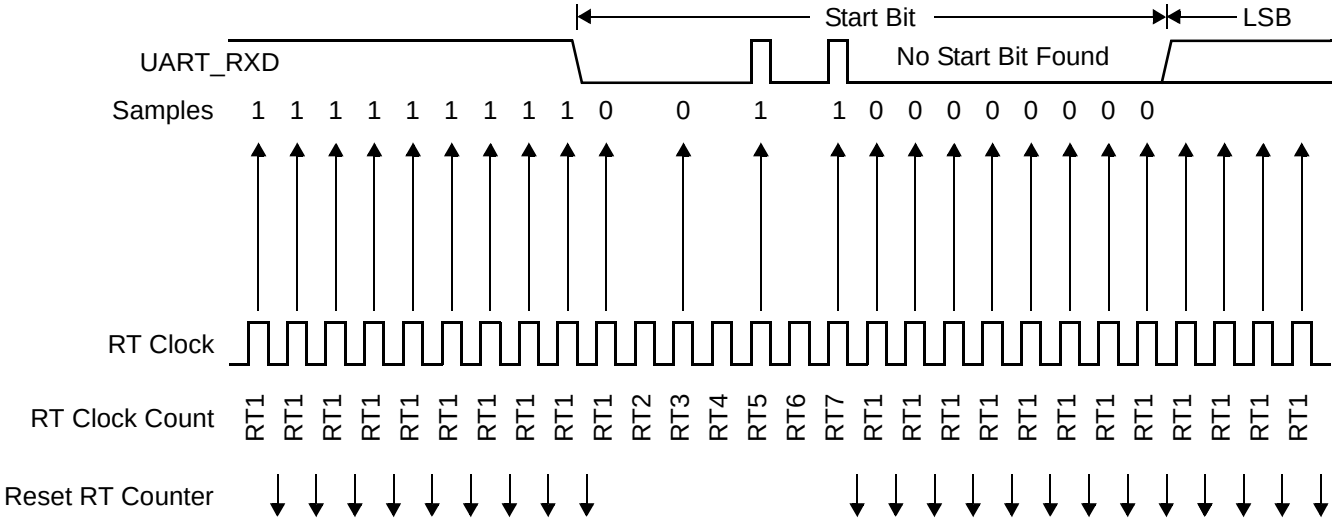


Figure 20-15. Start Bit Search Example 6

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.

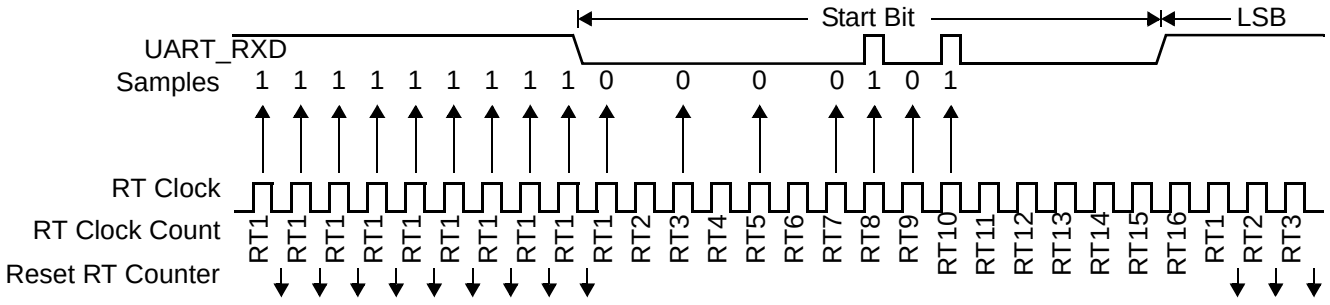


Figure 20-16. Start Bit Search Example 7

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.

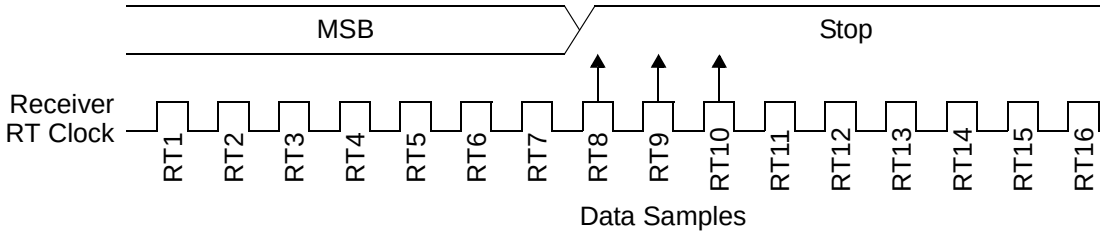


Figure 20-17. Slow Data

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.

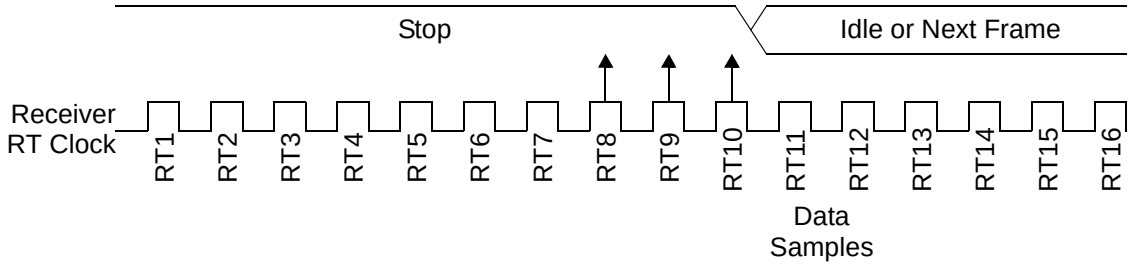


Figure 20-18. Fast Data

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 21**, *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.

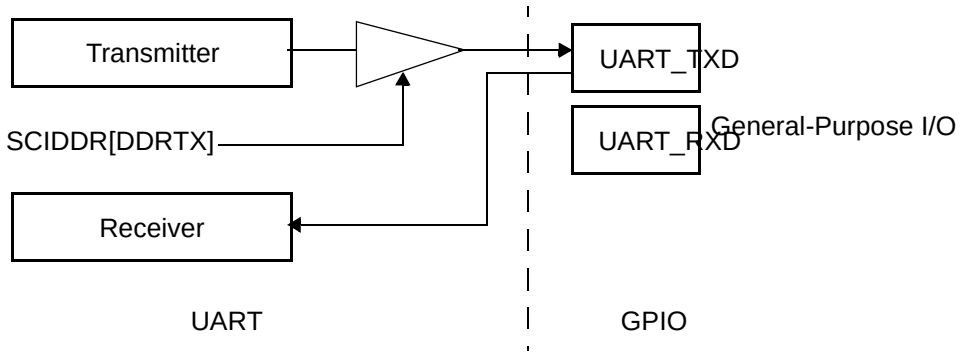


Figure 20-19. Single-Wire Operation (LOOPS = 1, RSRC = 1)

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 21**, *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.

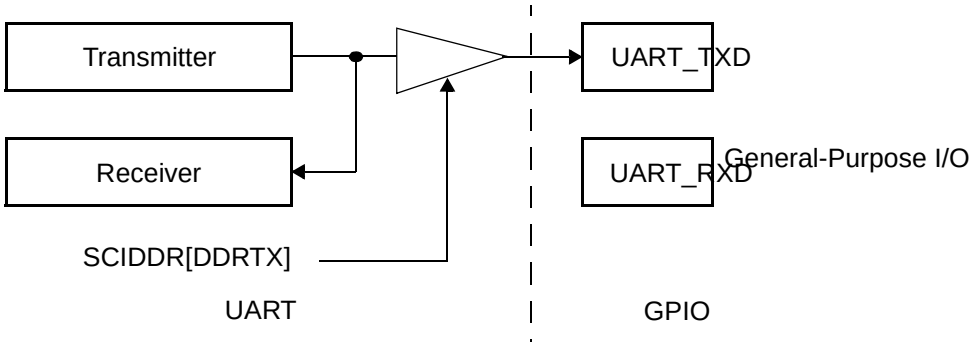


Figure 20-20. Loop Operation (LOOPS = 1, RSRC = 0)

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 8-3). 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 = (System Peripherals 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 21**, *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.			

21 Timers

The MSC8157E device includes 3 types of timers:

- *Device-level timers*. Each MSC8157E device contains sixteen 16-bit and eight 32-bit device timers. Each can be used by any of the DSP cores within MSC8157E 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 *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 MSC8157E device includes 8 software watchdog timers. Each of the software watchdog timers can be used by any of the cores within MSC8157E as well as by an external host. For details, see **Section 21.3**.

21.1 Device-Level Timers

There are four identical 16-bit quad timer modules and two identical 32-bit quad timers in the MSC8157E device. Each quad timer module contains four identical timer groups (16-bit or 32-bit, respectively) that serve as frequency dividers, clock generators, and event counters. Each 16-bit timer or 32-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 MSC8157E inputs/outputs as listed in **Table 21-1** and **Table 21-2**. 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 16-bit Timers Connectivity

Timer Module	Timer Channel	Input Signal Type	Input Signal	Output Type	Output
0	0	External	TMR0	—	—
	1	External	TMR1	External	TMR0
	2	External	CLKIN	—	—
	3	—	—	External	TMR1
1	0	External	TMR0	—	—
	1	External	TMR2	—	—
	2	External	CLKIN	—	—
	3	—	—	External	TMR2

Table 21-1. Device-Level 16-bit Timers Connectivity (Continued)

Timer Module	Timer Channel	Input Signal Type	Input Signal	Output Type	Output
2	0	External	TMR0	—	—
	1	External	TMR3	—	—
	2	External	CLKIN	—	—
	3	—	—	External	TMR3
3	0	External	TMR0	—	—
	1	External	TMR4	—	—
	2	External	CLKIN	—	—
	3	—	—	External	TMR4

Notes:
1. 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.
2. Inputs and outputs for the TMRn signal lines are multiplexed. See **Table 3-12*Timer Signals,*** on page 3-24 .
3. Device-level 16-bit timer IP Bus Clock is either the peripherals clock (250 MHz) or the SerDes PLL2 reference clock. See **Section 15.10.53**, *SRDS Bank 1–2 PLL Control Register 0 (SRDSB[1–2]PLLCR0)*, on page 15-96. GCR8[TIMERS_CLKMUX_SEL] determines the IP Bus Clock for all 16-bit device level timers (see **Section 8.2.19**, *General Control Register 8 (GCR8)*, on page 8-33.
4. The inputs chosen for each channel should not run at a frequency that is more than half of the frequency of the Timers internal clock.

Table 21-2. Device-Level 32-bit Timers Connectivity

Timer Module	Timer Channel	Input Signal Type	Input Signal	Output Type	Output
0	0	MUX	mux0	—	—
	1	MUX	mux1	—	—
	2	MUX	mux2	—	—
	3	MUX	mux3	External	TMR5
1	0	MUX	mux4	—	—
	1	MUX	mux5	—	—
	2	MUX	mux6	—	—
	3	MUX	mux7	External	TMR6

Notes:
1. 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 TMR32bnSCR[OEN] bit for the timer (see **Section 21.4.1.2**).
2. Inputs and outputs for the TMRn signal lines are multiplexed. See **Table 3-12*Timer Signals,*** on page 3-24 .
3. Device-level 32-bit timers IP Bus Clock is either the Peripherals Clock (250 MHz) or the SerDes PLL2 reference clock. See **Section 15.10.53**, *SRDS Bank 1–2 PLL Control Register 0 (SRDSB[1–2]PLLCR0)*, on page 15-96. GCR8[TIMERS_32B_CLKMUX_SEL] determines the IP Bus Clock for all 32-bit device level timers (see **Section 8.2.19**, *General Control Register 8 (GCR8)*, on page 8-33.
4. Input multiplexers (mux0 to mux7) are independent multiplexers determining the input per each timer. Each multiplexer has the same input options as can be seen in **Table 8-18*GCR7 Bit Descriptions,*** on page 8-30 . The 32-bit Timer0 inputs are determined by GCR7[TIMER_32B_0_MUX0_SEL to TIMER_32B_0_MUX3_SEL].
5. The inputs chosen for each channel should not run at a frequency that is more than half of the frequency of the Timers internal clock.

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 Timers clock rate when the timer input signals are not used.
- Maximum count rate is half the Timers clock when the timer input signals are used.
- Each counter has a separate prescaler.

21.1.2 Timer Module Architecture

Each quad timer module contains four timers. The block diagram of one 16-bit timer within a quad timer module is shown in **Figure 21-1**. The 32-bit timer block is similar. 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:
  - Timers clock
  - Timers clock 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 21-24) and then set the TMRxSCTL[FORC] bit (page 21-24).

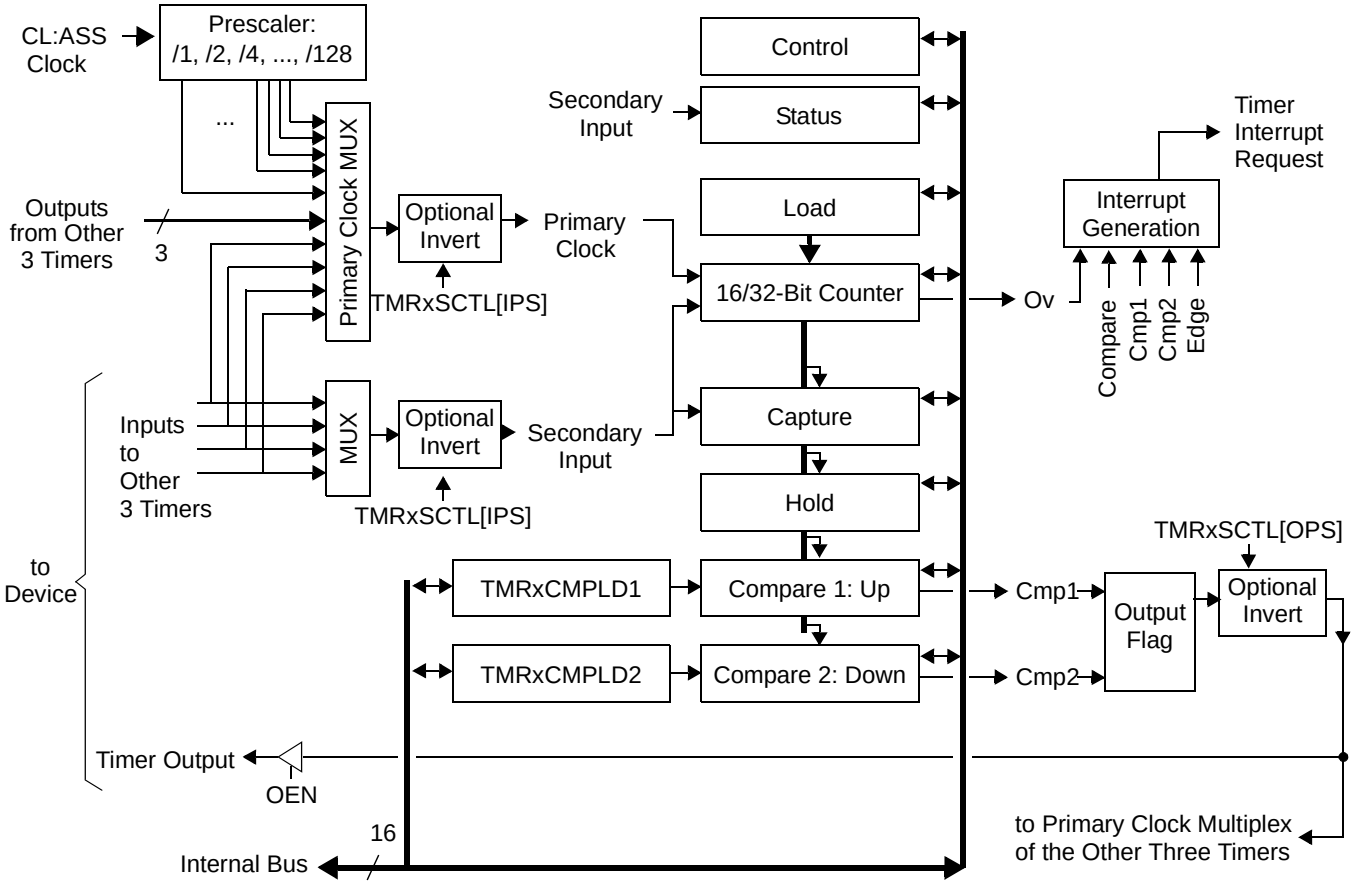


Figure 21-1. 16-Bit Timer Module Block Diagram — One of Four Timers

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 21-3** summarizes the cascading restrictions.

Table 21-3. 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 Timers clock or external events via the timer input using the primary clock.

- Count the Timers clock 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 21-22). 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 21-4** summarizes the counting modes.

Table 21-4. 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 TMRxSCTL[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	—
Gated Count	011	Counts primary clock edges while the secondary input is high (low if TMRxSCTL[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*

Table 21-4. Summary of Timer Counting Modes

Counting Mode	CM Bits	Description	Primary Clock	Secondary Clock
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 TMRxSCTL[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 TMRxSCTL[IPS] bit.				

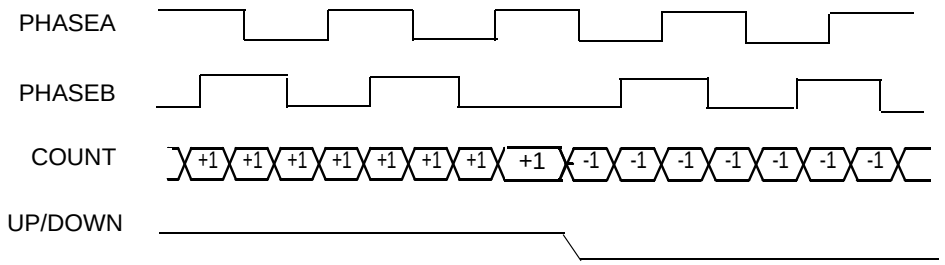


Figure 21-2. Quadrature Incremental Position Encoder

Other count modes derived as special cases of the modes described in **Table 21-4** 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 21-6**).

- 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 Timers clock/1) if the timer is set up as follows:

- TMRxCTL[CM] = 001 to count the rising edges of the primary source.
  (see **Table 21-6*TMR[0–3]SCTL[0–3] Bit Descriptions,*** on page 21-24).
- 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 21-6**).
- 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 [Timers clock (250 MHz)]/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.

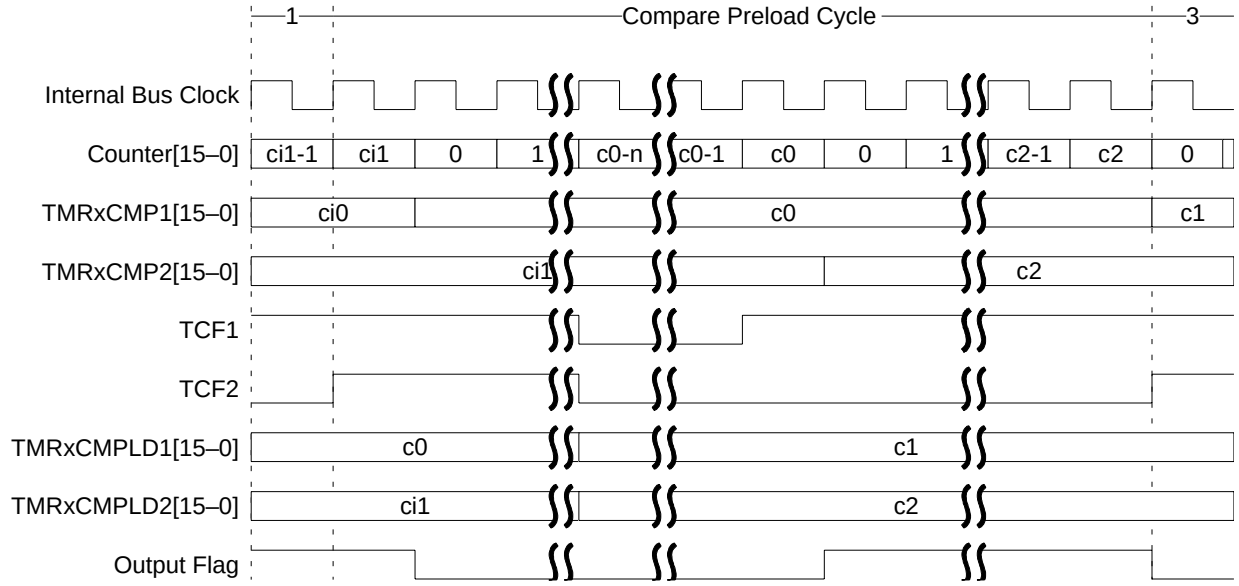


Figure 21-3. Compare Preload Timing

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.

Count until compare and then reinitialize is controlled by CTRL[LENGTH] bit. If set, the counter counts until the value stored at compare register (CMP1 or CMP2, depending on count direction) and at the next trigger the counter will reinitialize.

For counting until a variable value is required, use the following method:

1. Clear CTRL[LENGTH] bit.
2. Set COMSCR[CL1] to 0b01.
3. Set CMP1 to the first compare value (X).
4. Add to CMPLD1 the second compare value (Y), i.e. set CMPLD1 to X+Y.
5. At every compare event, the service routine should add to CMPLD1 the next compare value.
6. Overflow is supported by the counter, that is, when the counter value is 0xFFFF_FFFF the next trigger causes it to rollover to 0x0000_0000.

Note: This method is described is for count-up mode. For count-down mode, instead of COMSCR[CL1], CMP1 and CMPLD1, use COMSCR[CL2], CMP2 and CMPLD2.

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 21-9.

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. The feature is used for updating the CMP1 and CMP2 registers from CMPLD1 and CMPLD2 by hardware whenever a successful compare occurs. To ensure correct functionality, use the loading method described in this section.

Note: When using the Compare Preload feature, never use the Count to Compare and then Reinitialize feature.

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 21-9. 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**.

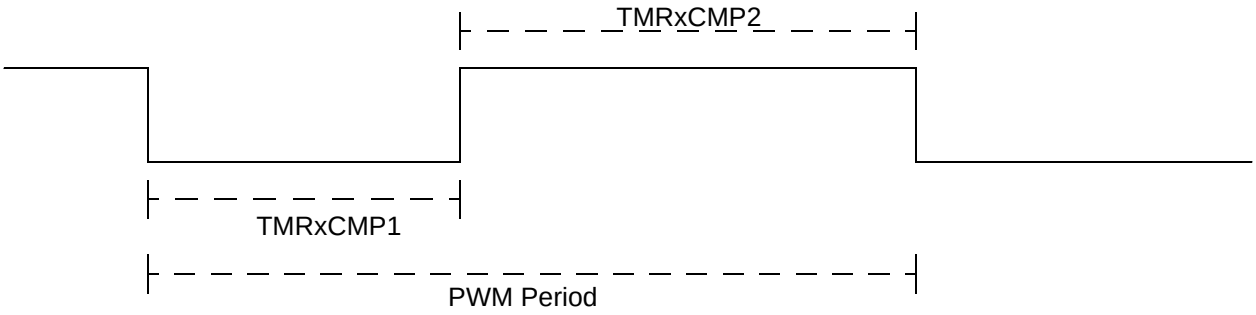


Figure 21-4. Variable PWM Waveform

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 21-28), 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 21-24), 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 21-24).

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 System Global Timer Register (32b timers only)

The System Global Timer and the System Global Timer Control Registers allow the user to read needed information from the four timer hold registers in one 32-bit read from the System Global

Timer Register. The System Global Timer Control Register permits a highly flexible selection of relevant sections from the four hold registers. The result is a 32-bit register that reflects user-configuration selections from the LSBs of timers 0 to 3. shows a functional diagram of the registers. See **Section 21.4.1.23**, *Timer_32b Global System Timer Register (TMR_32b_n_GLB)*, on page 21-39 and **Section 21.4.1.24**, *Timer_32b Global System Timer Control Register (TMR_32b_n_GLBCTL)*, on page 21-40 for programming details.

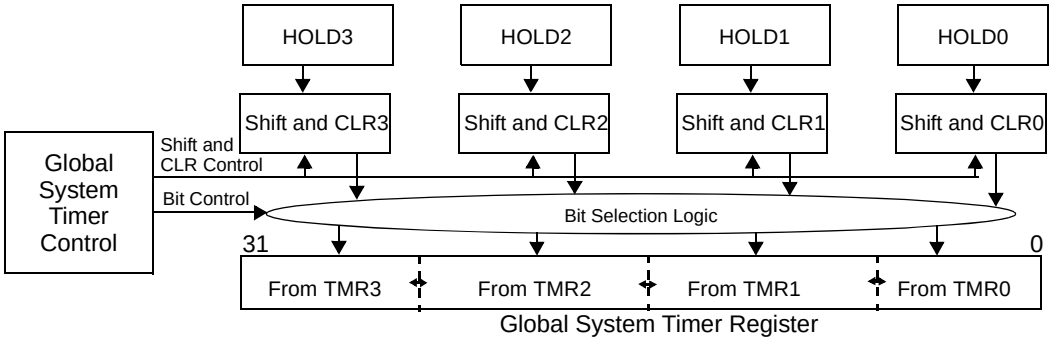


Figure 21-5. Global System Timer Diagram

21.1.7 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.

21.1.7.1 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.

21.1.7.2 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.

21.1.7.3 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.

21.1.8 Special CPRI Support

Both the 16-bit and 32-bit timers can be configured to support synchronous CPRI operation using a 122.88 MHz clock multiplexing option as a reference clock. Use the following steps to enable this special mode:

1. Configure the reference clock frequency by writing a “1” to the SRDSB2PLLCR0[RFCK_EN] field. See **Section 15.10.53**, *SRDS Bank 2 Reset Control Register (SRDSB2RSTCTL)*, on page 15-96.
2. For 16-bit timers, select the reference clock by writing a “1” to the GCR8[TIMERS_CLKMUX_SEL] field. See **Section 8.2.19**, *General Control Register 8 (GCR8)*, on page 8-33. To use them, the timers must be enabled by writing a 1 to the TMRnSCTL[OEN] bit (see **Section 21.4.1.2**, *Timer Channel Status and Control Registers (TMRnSCTLx)*, on page 21-24).
3. For 32-bit timers, select the reference clock by writing a “1” to the GCR8[TIMERS_32B_CLKMUX_SEL] field. See **Section 8.2.19**, *General Control Register 8 (GCR8)*, on page 8-33.

Note: See **Table 21-1*Device-Level 16-bit Timers Connectivity,*** on page 21-1 and **Table 21-2*Device-Level 32-bit Timers Connectivity,*** on page 21-2 for configuration details.

21.2 SC3850 DSP Core Subsystem Timers

For a detailed description of the core subsystem timers, see the *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 MSC8157E 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.

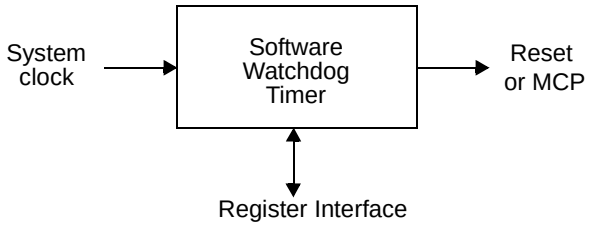


Figure 21-1. Software Watchdog Timer High Level Block Diagram

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 21-6** shows a state diagram for the watchdog timer.

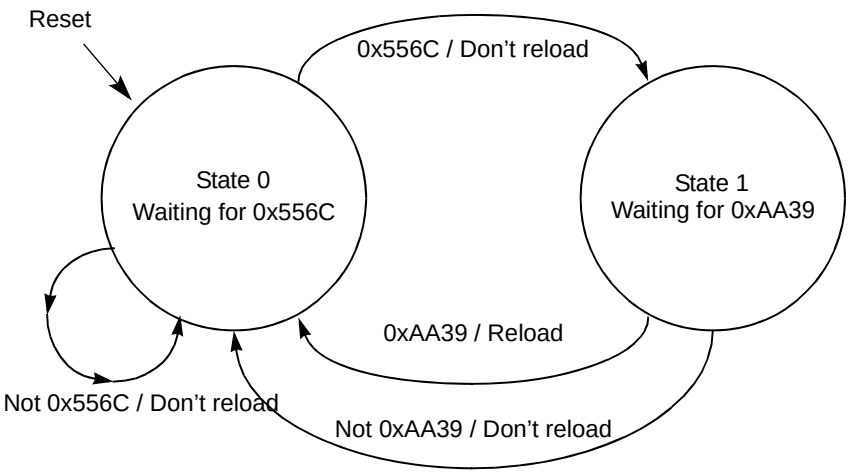


Figure 21-6. Software Watchdog Timer Service State Diagram

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 21-7** shows how to handle this need.

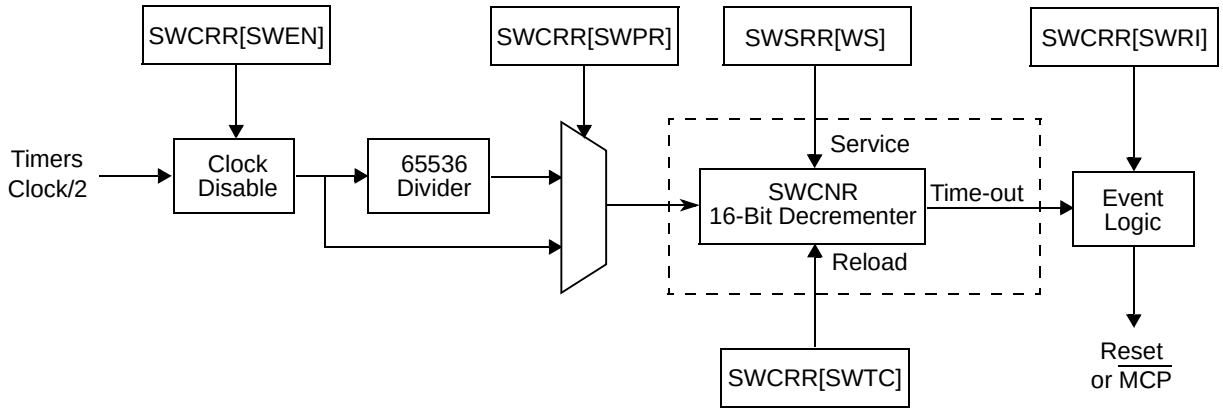


Figure 21-7. Software Watchdog Timer Functional Block Diagram

In **Figure 21-7**, the range is determined by SWCRR[SWTC]. The value in SWTC is then loaded into a 16-bit decrementer clocked by the Timers clock. 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 16-bit 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 21-22.
- Timer Channel Status and Control Registers (TMR[0–3]SCTL[0–3]), page 21-24.
- Timer Channel Compare Register 1 (TMR[0–3]CMP1[0–3]), page 21-26.
- Timer Channel Compare Register 2 (TMR[0–3]CMP2[0–3]), page 21-26.
- Timer Channel Compare Load Register 1 (TMR[0–3]CMPLD1[0–3]), page 21-26.
- Timer Channel Compare Load Register 2 (TMR[0–3]CMPLD2[0–3]), page 21-26.
- Timer Channel Comparator Status and Control Registers (TMR[0–3]COMSC[0–3]), page 21-27.
- Timer Channel Capture Register (TMR[0–3]CAP[0–3]), page 21-27.
- Timer Channel Load Register (TMR[0–3]LOAD[0–3]), page 21-28.
- Timer Channel Hold Registers (TMR[0–3]HOLD[0–3]), page 21-28.
- Timer Channel Counter Register (TMR[0–3]CNTR[0–3]), page 21-28.

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

The 32-bit device-level timer registers are listed as follows, along with the pages on which the registers are discussed:

- Timer_32b Channel Compare Register 1 (TMR_32b_[0–1]_CMP1[0–3]), page 21-29.
- Timer_32b Channel Compare Register 2 (TMR_32b_[0–1]_CMP2[0–3]), page 21-29.
- Timer_32b Channel Capture Register (TMR_32b_[0–1]_CAP[0–3]), page 21-30.
- Timer_32b Channel Load Register (TMR_32b_[0–1]_LOAD[0–3]), page 21-30.
- Timer_32b Channel Hold Registers (TMR_32b_[0–1]_HOLD[0–3]), page 21-31.
- Timer_32b Channel Counter Register (TMR_32b_[0–1]_CNTR[0–3]), page 21-31.
- Timer_32b Channel Control Registers (TMR_32b_[0–1]_CTL[0–3], page 21-32.
- Timer_32b Channel Status and Control Registers (TMR_32b_[0–1]_SCR[0–3]), page 21-35.
- Timer_32b Channel Compare Load Register 1 (TMR_32b_[0–1]_CMPLD1[0–3]), page 21-37.
- Timer_32b Channel Compare Load Register 2 (TMR_32b_[0–1]_CMPLD2[0–3]), page 21-37.
- Timer_32b Channel Comparator Status and Control Registers (TMR_32b_[0–1]_COMSC[0–3]), page 21-38.
- Timer_32b Global System Timer Register (TMR_32b_[0–1]_GLB), page 21-39
- Timer_32b Global System Timer Control Register (TMR_32b_GLBCTL), page 21-40
- Timer_32b Timer Set and Forget Timer (TMR_32b_[0–1]_SAF), page 21-42
- Timer_32b Timer Clear Lock Register (TMR_32b_[0–1]_CLRL), page 21-43

Note: The base addresses for the device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400
- Timer_32b_1 = 0xFFF26600

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 21-5. 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 TMRxSCTL[IPS] = 0. Falling edges are counted only when TMRxSCTL[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 No operation. Disabled. 001 Count rising edges of the primary source. 010 Count rising and falling edges of the primary source. 011 Count rising edges of the primary source while the secondary input is high active. 100 Quadrature count mode, uses primary clock and secondary input. 101 Count rising edges of the primary clock; secondary input specifies direction (1 = minus). 110 Edge of the secondary input triggers primary count until a compare occurs. 111 Cascaded timer mode (up/down).
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 (Timers clock/1). 1001 Prescaler (Timers clock/2). 1010 Prescaler (Timers clock/4). 1011 Prescaler (Timers clock/ 8). 1100 Prescaler (Timers clock/16). 1101 Prescaler (Timers clock/32). 1110 Prescaler (Timers clock/64). 1111 Prescaler (Timers clock/128).

Table 21-5. TMR[0–3]CTL[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
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.
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 21-13.	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 21-6. 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.
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 (TMRxSCTL[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** This bit reflects the current state of the external input pin selected via the secondary count source after application of the IPS bit. This bit is read-only.	0 Low. 1 High.

Table 21-6. TMR[0–3]SCTL[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
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 21-13.	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 21-13.	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.	
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 21-7. 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.
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
	COUNTER															
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.

21.4.1.12 Timer_32b Channel x Compare 1 Registers (TMR_32b_n_CMP1_x)

TMR_32b_[0–1]_CMP1_[0–3] Timer_32b Channel Compare 1 Registers — Offset x*0x40

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
	CV																															
Type	R/W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_32b_[0–1]_CMP1_[0–3] store the comparison value (CV) for comparison with the timer value.

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400
- Timer_32b_1 = 0xFFF26600

21.4.1.13 Timer_32b Channel x Compare 2 Registers (TMR_32b_n_CMP2_x)

TMR_32b_[0–1]_CMP2_[0–3] Timer_32b Channel Compare 2 Registers — Offset 0x04 + x*0x40

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
	CV																															
Type	R/W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_32b_[0–1]_CMP2_[0–3] store the comparison value (CV) for comparison with the timer value.

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400
- Timer_32b_1 = 0xFFF26600

21.4.1.14 Timer_32b Channel Capture Registers (TMR_32b_nCAPx)

TMR_32b_[0–1]_CAP[0–3] Timer_32b Channel Capture Registers Offset 0x08 + x*0x40

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
	CAPV																															
Type	R/W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_32b_[0–1]_CAP[0–3] store the values captured from the timers.

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400
- Timer_32b_1 = 0xFFF26600

21.4.1.15 Timer_32b Channel Load Registers (TMR_32b_n_LOADx)

TMR_32b_[0–1]_LOAD[0–3]Timer_32b Channel Load Registers Offset 0x0C + x*0x40

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
	LDV																															
Type	R/W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_32b_[0–1]_LOAD[0–3] store the value used to load the timer.

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400
- Timer_32b_1 = 0xFFF26600

21.4.1.16 Timer_32b Channel Hold Registers (TMR_32b_n_HOLDx)

TMR_32b_[0–1]_HOLD[0–3] Timer_32b Channel Hold Registers Offset 0x10 + x*0x40

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
	HDV																															
Type	R/W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_32b_[0–1]_HOLD[0–3] store the value whenever any timer is read.

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400
- Timer_32b_1 = 0xFFF26600

21.4.1.17 Timer_32b Channel Counter Registers (TMR_32b_n_CNTRx)

TMR_32b_[0–1]_CNTR[0–3] Timer_32b Channel Counter Registers Offset 0x14 + x*0x40

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
	COUNTER																															
Type	R/W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_32b_[0–1]_CNTR[0–3] are counters.

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400
- Timer_32b_1 = 0xFFF26600

21.4.1.18 Timer_32b Channel Control Registers (TMR_32b_n_CTLx)

TMR_32b_[0–1]_CTL[0–3] Timer_32b Control Registers Offset 0x18 + x*0x40

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
	CM			PCS				SC		ONCE	LEN	DIR	CO INIT	OM		
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-8. TMR_32b_[0–1]_CTL[0–3] Bit Descriptions

Name	Reset	Description	Settings
— 31–16	0	Reserved. Write to zero for future compatibility.	
CM 15–13	0	**Count Mode** Control the basic counting behavior of the counter. Rising edges are counted only when TMR_32b_SCR[IPS] = 0. Falling edges are counted only when TMR_32b_SCR[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 No operation. Disabled. 001 Count rising edges of the primary source. 010 Count rising and falling edges of the primary source. 011 Count rising edges of the primary source while the secondary input is high active. 100 Quad count mode, uses primary and secondary sources. 101 Count rising edges of the primary clock; secondary input specifies direction (1 = minus). 110 Edge of the secondary input triggers primary count until a compare occurs. 111 Cascaded timer mode (up/down).

Table 21-8. TMR_32b_[0–1]_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. 0101 Timer 1 output. 0110 Timer 2 output. 0111 Timer 3 output. 1000 Prescaler (Timers clock/1). 1001 Prescaler (Timers clock/2). 1010 Prescaler (Timers clock/4). 1011 Prescaler (Timers clock/ 8). 1100 Prescaler (Timers clock/16). 1101 Prescaler (Timers clock/32). 1110 Prescaler (Timers clock/64). 1111 Prescaler (Timers clock/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 TMR_32b_x_CMP1 value. If counting down, a successful compare occurs when a timer reaches TMR_32b_x_-CMP2 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 TMR_32b_x_CMP1 value. If counting down, a successful compare occurs when the timer reaches the TMR_32b_x_CMP2 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.
COINIT 3	0	**Co-channel Initialization** Enables another timer within the same module to force the reinitialization of this timer when the other timer has an active compare event.	0 Co-channel timers can not force a reinitialization of this timer. 1 Co-channel timers may force a reinitialization of this timer.

Table 21-8. TMR_32b_[0–1]_CTL[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
OM 2–0	0	**Output Mode** Determine the mode of operation for the OFLAG 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 OFLAG output on successful compare. 010 Set OFLAG output on a successful compare. 011 Toggle OFLAG output flag when a successful compare occurs. 100 Toggle OFLAG 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.

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400
- Timer_32b_1 = 0xFFF26600

21.4.1.19 Timer_32b Channel Status and Control Registers (TMR_32b_n_SCRx)

TMR_32b_[0–1]_SCR[0–3] Timer_32b Channel Status and Control Register — Offset 0x1C + x*0x40

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
	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 21-9. TMR_32b_[0–1]SCR[0–3] Bit Descriptions

Name	Reset	Description	Settings	
— 31–16	0	Reserved. Write to zero for future compatibility.		
TCF 15	0	**Timer Compare Flag** Set when a successful compare occurs. Clear the bit by writing 0 to it.	0 1	No successful compare. Successful compare.
TCFIE 14	0	**Timer Compare Flag Interrupt Enable** Enables interrupts when the TCF bit is set.	0 1	No interrupt. Interrupt.
TOF 13	0	**Timer Overflow Flag** Set when the timer rolls over its maximum value 0xFFFFFFFF or 0x00000000, depending on count direction. Clear the bit by writing 0 to it.	0 1	No overflow. 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 1	No interrupt. Interrupt.
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 (TMR_32b_SCR[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 1	No action. Positive input transition while timer enabled.
IEFIE 10	0	**Input Edge Flag Interrupt Enable** Enables interrupts when the IEF bit is set.	0 1	No action. Interrupts enabled.
IPS 9	0	**Input Polarity Select** Inverts the input signal polarity.	0 1	No action. Invert signal polarity.

Table 21-9. TMR_32b_[0–1]SCR[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
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 21-13.	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 21-13.	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.	
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.

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400

- Timer_32b_1 = 0xFFF26600

21.4.1.20 Timer_32b Channel Compare Load 1 Registers (TMR_32b_n_CMPLD1x)

TMR_32b_[0–1]_CMPLD1[0–3] Timer_32b Channel Compare Load 1 Registers — Offset 0x20 + x*0x40

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
	CLV1																															
Type	R/W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_32b_[0–1]CMPLD1[0–3] store the preload value for the TMR_32b_[0–1]CMP1[0–3].

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400
- Timer_32b_1 = 0xFFF26600

21.4.1.21 Timer_32b Channel Compare Load 2 Registers (TMR_32b_n_CMPLD2x)

TMR_32b_[0–1]_CMPLD2[0–3] Timer_32b Channel Compare Load 2 Registers — Offset 0x24 + x*0x40

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
	CLV2																															
Type	R/W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR_32b_[0–1]CMPLD2_[0–3] store the preload value for the TMR_32b_[0–1]CMP2[0–3].

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400
- Timer_32b_1 = 0xFFF26600

21.4.1.22 Timer_32b Channel Comparator Status and Control Registers (TMR_32b_n_COMSCx)

TMR_32b_[0–1]_COMSC[0–3] Timer_32b Channel Comparator Status and Control Registers — Offset 0x28 + x*0x40

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
	—								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_32b_[0–1]_COMSC[0–3] store the preload values used for the TMR_32b_[0–]CMP2[0–3] registers.

Table 21-10. TMR_32b_[0–1]_COMSC[0–3] Bit Descriptions

Name	Reset	Description	Settings
— 31–16	0	Reserved. Write to zero for future compatibility.	
— 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.
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.

Table 21-10. TMR_32b_[0–1]_COMSC[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
CL2 3–2	0	**Compare Load Control 2** Control when TMR_32b_x_CMP2 is preloaded with the value from TMR_32b_x_CMPLD2.	00 Never preload. 01 Load upon successful compare with the value in TMR_32b_x_CMP1. 10 Load upon successful compare with the value in TMR_32b_x_CMP2. 11 Reserved.
CL1 1–0	0	**Compare Load Control 1** Control when TMR_32b_x_MP1 is preloaded with the value from TMR_32b_x_CMPLD1.	00 Never preload. 01 Load upon successful compare with the value in TMR_32b_x_CMP1. 10 Load upon successful compare with the value in TMR_32b_x_CMP2. 11 Reserved.

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400
- Timer_32b_1 = 0xFFF26600

21.4.1.23 Timer_32b Global System Timer Register (TMR_32b_n_GLB)

TMR_32b_[0–1]_GLB Timer_32b Global System Timer Register Offset 0x100

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
	GLOBAL SYSTEM TIMER VALUE																															
Type	R																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This read register is the Global System Timer Register that reflects the four timers hold registers.

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400
- Timer_32b_1 = 0xFFF26600

21.4.1.24 Timer_32b Global System Timer Control Register (TMR_32b_n_GLBCTL)

TMR_32b_[0–1]_GLBCTL Timer_32b Global System Timer ‘Control Register — Offset 0x104

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DHR3	DHR2	DHR1	DHR0	—							HR2				
Type	W															
Reset	1	1	1	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**
	—			HR1					—			HR0				
Type	W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-11. TMR_32b_[0–1]_GSDWCTRL Bit Descriptions

Name	Reset	Description	Settings
DHR3 31	1	**Disable Timer 3 Hold Reflection** Enable/disable Timer 3 hold reflection	0 Enable Timer 3 hold reflection. 1 Disable Timer 3 hold reflection
DHR2 30	1	**Disable Timer 2 Hold Reflection** Enable/disable Timer 2 hold reflection	0 Enable Timer 2 hold reflection. 1 Disable Timer 2 hold reflection
DHR1 29	1	**Disable Timer 1 Hold Reflection** Enable/disable Timer 1 hold reflection	0 Enable Timer 1 hold reflection. 1 Disable Timer 1 hold reflection
DHR0 28	1	**Disable Timer 0 Hold Reflection** Enable/disable Timer 0 hold reflection	0 Enable Timer 0 hold reflection. 1 Disable Timer 0 hold reflection
— 27–21	0	Reserved. Write to zero for future compatibility.	
HR2 20–16	0	**Timer 2 Hold Reflection** This field chooses which bits in the Timer 2 HOLD register to reflect in the TMR_32b_GLB. The number entered in this field is reduced by the number of bits selected by HR0 and HR1 and the result indicates how many Timer 2 HOLD low bits to display in the TMR_32b_GLB.	
— 15–13	0	Reserved. Write to zero for future compatibility.	
HR1 12–8	0	**Timer 1 Hold Reflection** This field chooses which bits in the Timer 1 HOLD register to reflect in the TMR_32b_GLB register. The number entered in this field is reduced by the number specified by HR0 and the result indicates how many Timer 1 HOLD low bits to display in TMR_32b_GLB.	
— 7–5	0	Reserved. Write to zero for future compatibility.	
HR0 4–0	0	**Timer 0 Hold Reflection** This field chooses which bits in the Timer 0 HOLD register to reflect in the TMR_32b_GLB register. The number entered in this field chooses the highest bit to reflect in TMR_32b_GLB, which displays this bit down to the least significant bit in the Timer 0 HOLD register.	

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400

- Timer_32b_1 = 0xFFF26600

This register controls which bit from the 4 timer/counters in the timer module are reflected in the TMR_32b_GLB. HR0 takes precedent. If it is configured as 31, then no other timer can have bits displayed. HR1 is next and if it is larger thank HR0, the difference specifies the number of low bits from the Timer 1 HOLD register to display. HR2 is next and if it is larger than HR1, the difference specifies the number of Timer 2 HOLD register low bits to display. Any remaining bits are reflected from the Timer 3 HOLD register low bits. **Figure 21-8** shows an example configuration and its effects on the display in the TMR_32b_GLB.

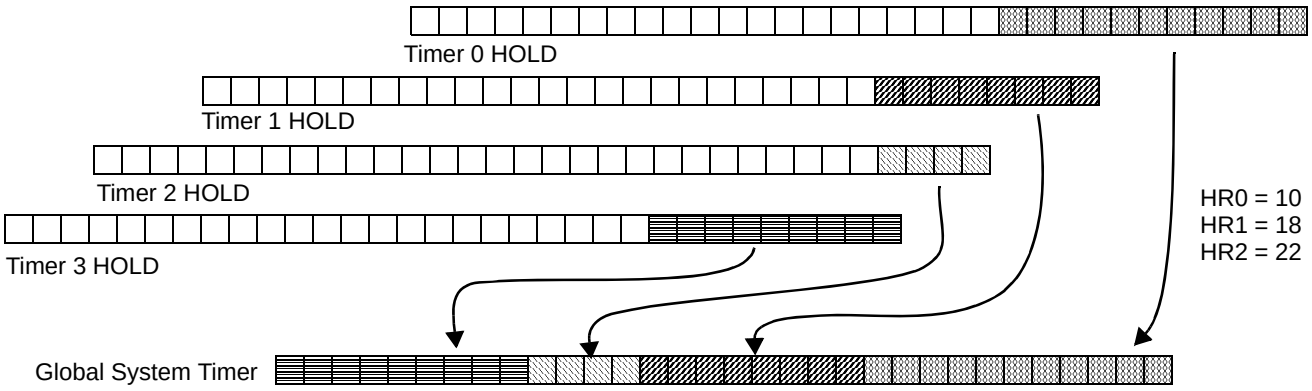


Figure 21-8. Hold Reflection Example Showing SWR Content

21.4.1.25 Timer_32b Timer Set and Forget Register (TMR_32b_n_SAF)

TMR_32b_[0–1]_SAF Timer_32b Set and Forget Register Offset 0x108

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**
	—												SAF_TMR3	SAF_TMR2	SAF_TMR1	SAF_TMR0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-12. TMR_32b_[0–1]_SAF Bit Descriptions

Name	Reset	Description	Settings
— 31–4	0	Reserved. Write to zero for future compatibility.	
SAF_TMR3 3	0	**Set and Forget Timer Input 3** When set (1), timer input 3 is in lock mode in which any signal change is ignored by the Quad Timers until released by the TIMR_CLRL[CL3] bit. When this bit is cleared (0), the Quad Timer is sensitive to any signal change according to the channel CTRL configuration.	0 Quad timer sensitive to changes in timer input 3. 1 Quad timer is locked and ignores any changes in timer input 3.
SAF_TMR2 2	0	**Set and Forget Timer Input 2** When set (1), timer input 2 is in lock mode in which any signal change is ignored by the Quad Timers until released by the TIMR_CLRL[CL2] bit. When this bit is cleared (0), the Quad Timer is sensitive to any signal change according to the channel CTRL configuration.	0 Quad timer sensitive to changes in timer input 2. 1 Quad timer is locked and ignores any changes in timer input 2.
SAF_TMR1 1	0	**Set and Forget Timer Input 13** When set (1), timer input 1 is in lock mode in which any signal change is ignored by the Quad Timers until released by the TIMR_CLRL[CL1] bit. When this bit is cleared (0), the Quad Timer is sensitive to any signal change according to the channel CTRL configuration.	0 Quad timer sensitive to changes in timer input 1. 1 Quad timer is locked and ignores any changes in timer input 1.
SAF_TMR0 0	0	**Set and Forget Timer Input 0** When set (1), timer input 0 is in lock mode in which any signal change is ignored by the Quad Timers until released by the relevant TIMR_CLRL[CL0] bit. When this bit is cleared (0), the Quad Timer is sensitive to any signal change according to the channel CTRL configuration.	0 Quad timer sensitive to changes in timer input 0. 1 Quad timer is locked and ignores any changes in timer input 0.

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400
- Timer_32b_1 = 0xFFF26600

21.4.1.26 Timer_32b Timer Clear Lock Register (TMR_32b_n_CLRL)

TMR_32b_[0–1]_CLRL Timer_32b Clear Lock Register Offset 0x10C

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**
	—												CL3	CL2	CL1	CL0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-13. TMR_32b_[0–1]_CLRL Bit Descriptions

Name	Reset	Description	Settings
— 31–4	0	Reserved. Write to zero for future compatibility.	
CL3 3	0	**Clear Lock for Timer Input 3** When set (1), the timer input 3 lock is released. This bit automatically clears after the lock release cycle.	0 No action. 1 Clear lock for timer input 3.
CL2 2	0	**Clear Lock for Timer Input 2** When set (1), the timer input 2 lock is released. This bit automatically clears after the lock release cycle.	0 No action. 1 Clear lock for timer input 2.
CL1 1	0	**Clear Lock for Timer Input 1** When set (1), the timer input 1 lock is released. This bit automatically clears after the lock release cycle.	0 No action. 1 Clear lock for timer input 1.
CL0 0	0	**Clear Lock for Timer Input 0** When set (1), the timer input 0 lock is released. This bit automatically clears after the lock release cycle.	0 No action. 1 Clear lock for timer input 0.

Note: The base addresses for the 32-bit device level timer modules are as follows:

- Timer_32b_0 = 0xFFF26400
- Timer_32b_1 = 0xFFF26600

21.4.2 SC3850 DSP Core Subsystem Timers

For a detailed information about programming the core subsystem timers, see the *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 21-45.
- System Watchdog Count Register 0–7 (SWCNR[0–7]), see page 21-46.
- System Watchdog Service Register 0–7 (SWSRR[0–7]), see page 21-46.

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 21-14** defines the SWCRR[0–7] bit fields.

Table 21-14. 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 21-15. 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 21-16** defines the bit fields of SWSRR.

Table 21-16. 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.

22 GPIO

The MSC8157E device has 32 general-purpose I/O (GPIO) ports, 16 of which can be configured as external maskable interrupt inputs, and 25 can be configured as dedicated peripheral interface ports. As GPIOs, each port is configured as an input or output. The dedicated peripheral functions multiplexed with the GPIO ports are grouped to maximize the usefulness of the ports in the greatest number of applications.

Configuration as an input or output port is done using a register for data outputs 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.

Note: To understand the port assignment capability described in this chapter, you must also understand the operation of the CPRI, SPI, timers, UART, I^2C, and DMA peripherals.

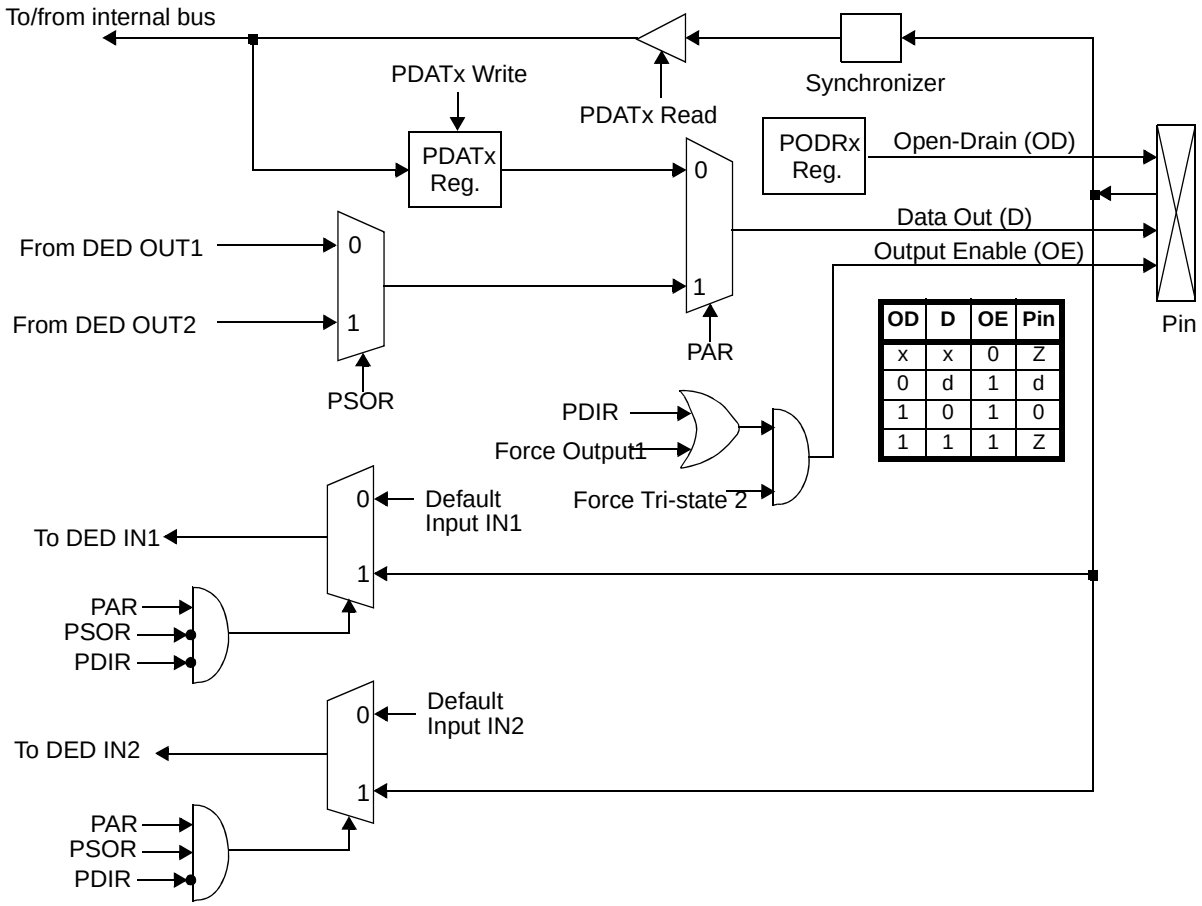
22.1 Features

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.
- Seven 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.
- Twelve of the GPIOs can be configured as CPRI signals (two per controller).

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	CP_SYNC1	CP_SYNC1	0
1	—	$\overline{\text{IRQ1}}$	1	CP_SYNC2	CP_SYNC2	0
2	—	$\overline{\text{IRQ2}}$	1	CP_SYNC3	CP_SYNC3	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	CP_SYNC4	CP_SYNC4	0
6	—	$\overline{\text{IRQ6}}$	1	CP_SYNC5	CP_SYNC5	0
7	—	$\overline{\text{IRQ7}}$	1	CP_SYNC6	CP_SYNC6	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	TMR5	TMR5	0
17	—	CPRI_LOS3	0	SPI_SCK	SPI_SCK	0
18	—	CPRI_LOS4	0	SPI_MOSI	SPI_MOSI	0
19	—	CPRI_LOS5	0	SPI_MISO	SPI_MISO	0
20	—	CPRI_LOS6	0	$\overline{\text{SPI\_SL}}$	$\overline{\text{SPI\_SL}}$	1
21	—	—	0	TMR6	TMR6	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	—	CPRI_LOS1	0	UART_RXD	UART_RXD	1
29	—	CPRI_LOS2	0	UART_TXD	UART_TXD	0
30	—	—	0	I2C_SCL	I2C_SCL	1
31	—	—	0	I2C_SDA	I2C_SDA	1

Note: You must enable the port to use it as an input. See **Section 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.

23 Hardware Semaphores

The MSC8157E 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).

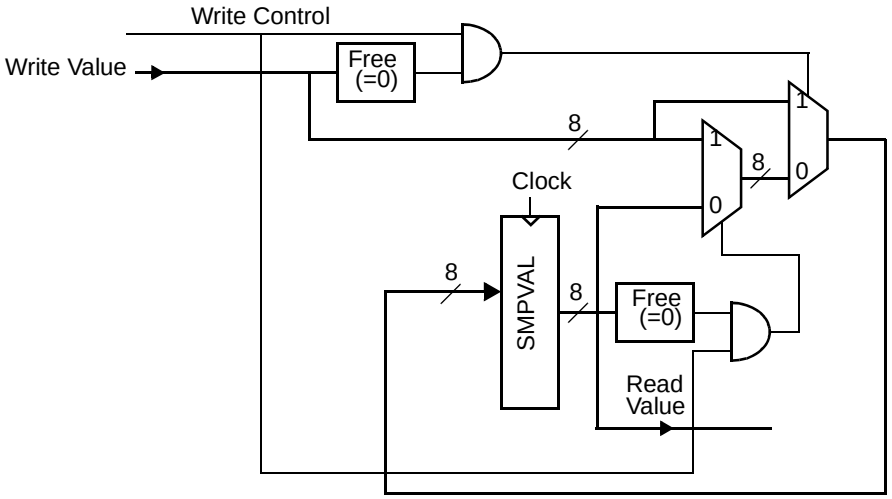


Figure 23-1. Hardware Semaphore Block Diagram

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.

24 I²C

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 MSC8157E device, the maximum I2C_SCL frequency is 400 kHz. **Figure 24-1** is a block diagram of the I²C block.

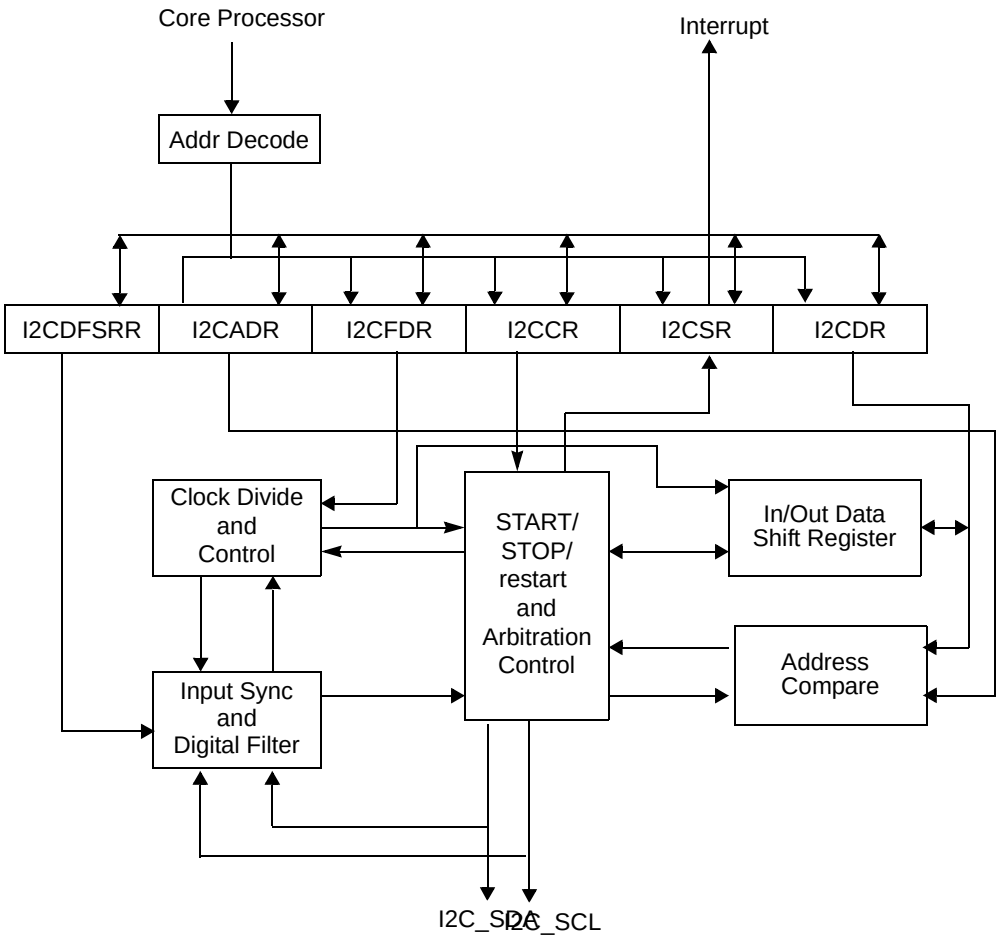


Figure 24-1. I²C Controller Block Diagram

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 [I2C Input Clock] 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.

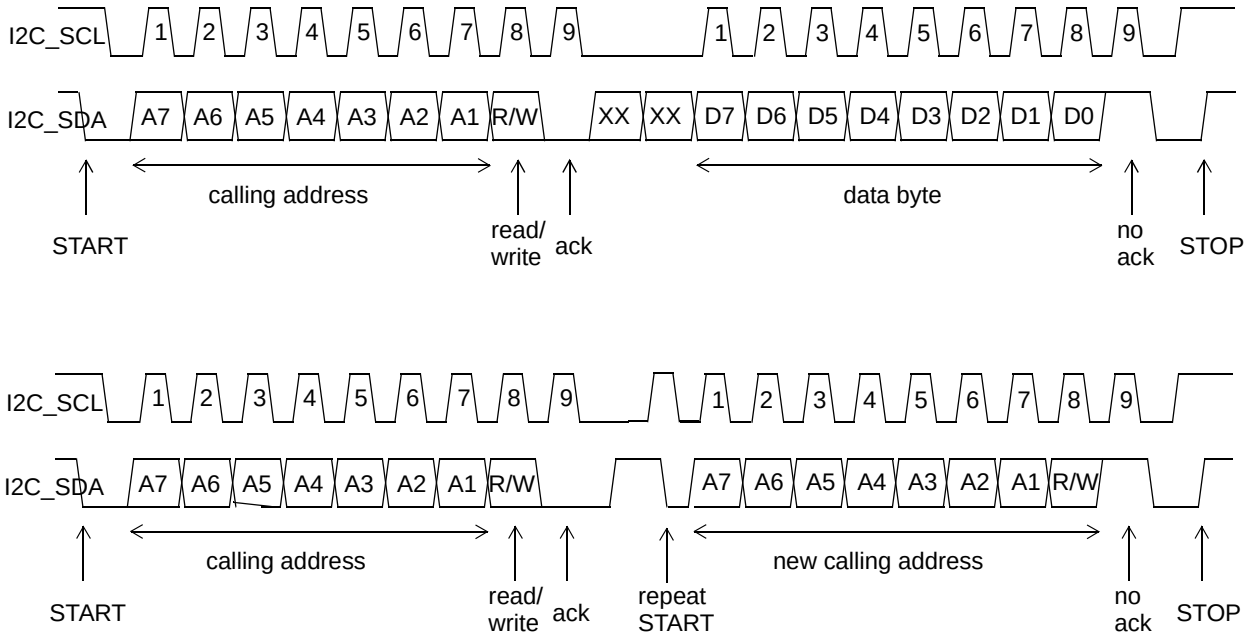


Figure 24-2. I^2C Interface Transaction Protocol

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 signalled 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 21**, *GPIO* for details.
- Update I2CFDR[FDR] and select the required division ratio to obtain the I2C_SCL frequency from the [I^2C Input Clock] 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.

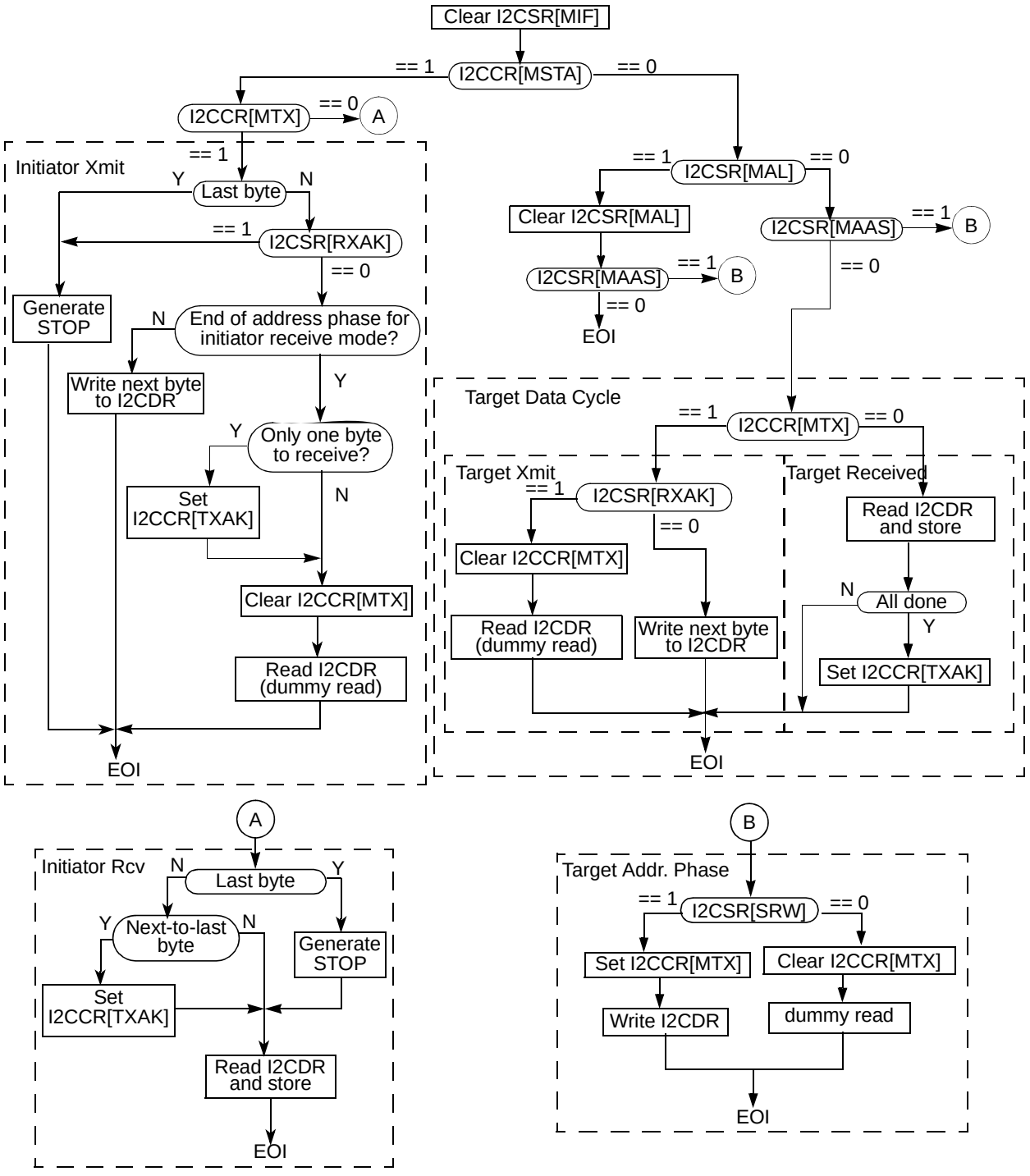


Figure 24-3. Example I^{2}C Interrupt Service Routine Flowchart

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 [I²C Input Clock] 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].

25 Debugging, Profiling, and Performance Monitoring

The MSC8157E 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. 1149.6 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 MSC8157E supports circuit-board test strategies based on this standard. This section covers aspects of JTAG that are specific to the MSC8157E. It includes the items that the standard requires to be defined, with additional information specific to the MSC8157E device. For details on the standard, refer to the **IEEE** Std. 1149.6 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 MSC8157E. For details on the OCE module functionality, see the *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 MSC8157E 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 MSC8157E JTAG can do the following:

- Perform boundary scan operations to check circuit-board electrical continuity.
- Bypass the MSC8157E for a given circuit-board test by effectively reducing the Boundary Scan Register (BSR) to a single cell.
- Sample the MSC8157E 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 MSC8157E-based device.
- Force test data onto the outputs of an MSC8157E-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. 1149.6-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 MSC8157E 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 MSC8157E 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.

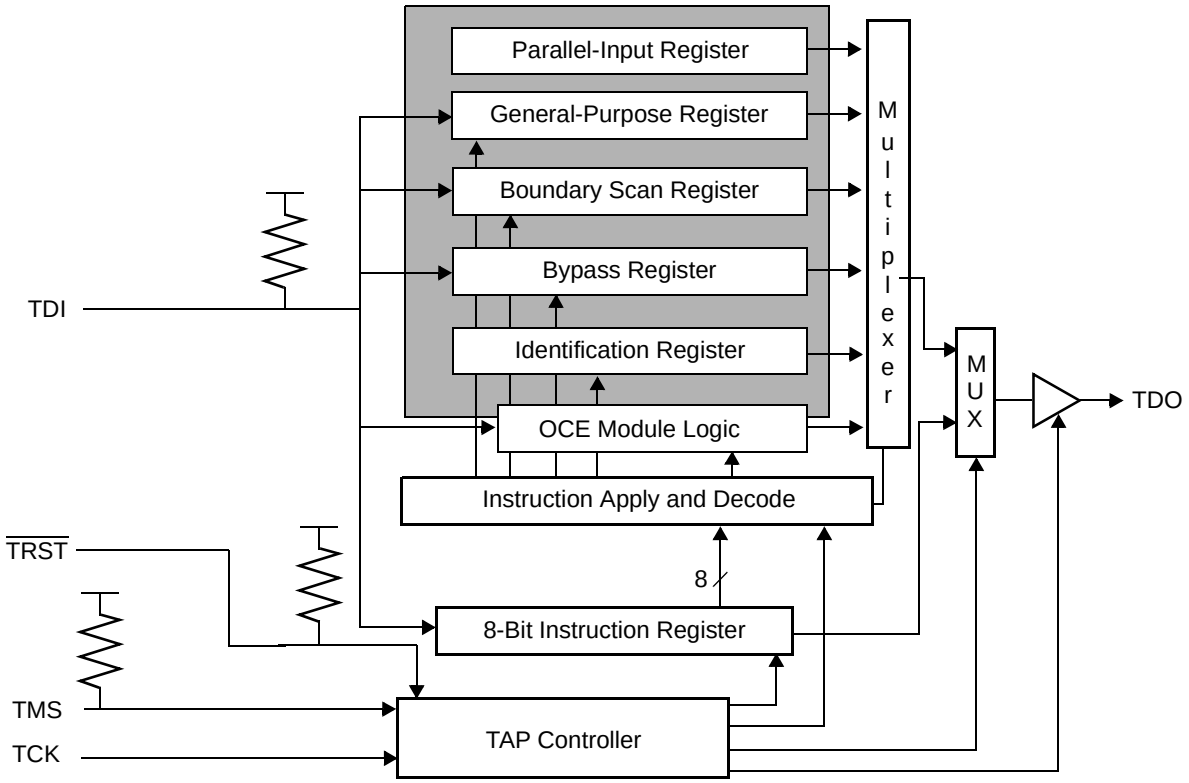


Figure 25-1. Test Logic Block Diagram

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. 1149.6 documentation.

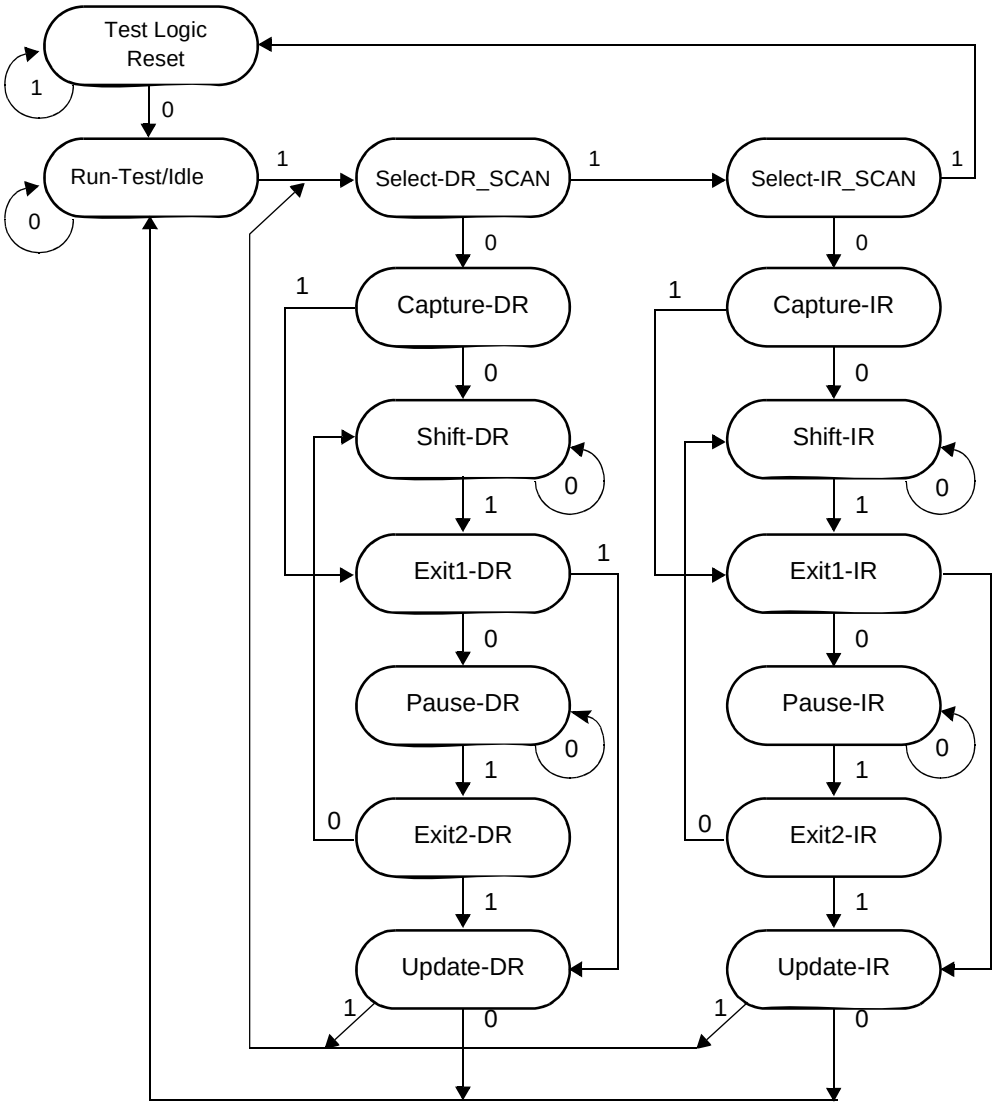


Figure 25-2. TAP Controller State Machine

25.1.3 Instruction Decoding

The MSC8157E includes the three mandatory public instructions EXTEST, SAMPLE/PRELOAD, and BYPASS and also supports the optional CLAMP and HIGHZ instructions defined by **IEEE Std.** 1149.6. 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 MSC8157E 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 MSC8157E 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 HRESET 1 All active MBISTs are complete
— 4–3	Reserved. Always 0.	

Table 25-2. Instruction Register Capture and SC3850 Core Status Values

Name/bits	Description	Settings
clock_end_ count 2	**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
— 1–0	Contains value (01) required by the JTAG standard	Read-only

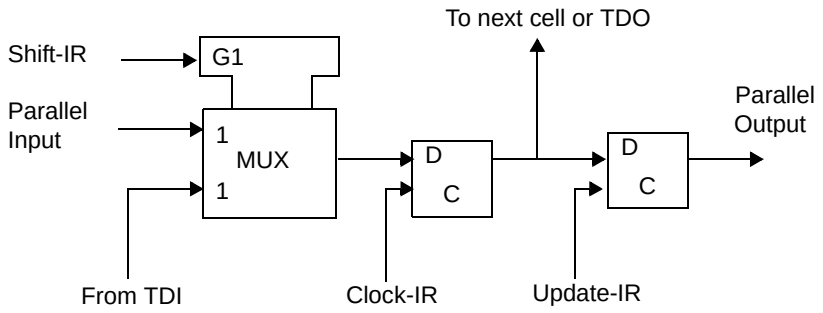


Figure 25-3. Instruction Register (IR) Configuration

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 MSC8157E 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. 1149.6 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. 1149.6. 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 MSC8157E system logic to force a predictable internal state while external boundary scan operations are performed.
0xF2	HIGHZ	Yes	Optional in the **IEEE** Std. 1149.6. 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 MSC8157E 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. 1149.6, 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 MSC8157E 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 MSC8157E-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. 1149.6. 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. 1149.6. This public instruction allows you to generate a debug request signal to the MSC8157E. 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 MSC8157E 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. 1149.6. 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 MSC8157E uses JTAG TAP for standard defined testing compatibilities and for multi-core OCE module control and OCE module interconnection control. The MSC8157E 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 MSC8157E 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 *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 MSC8157E. 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.

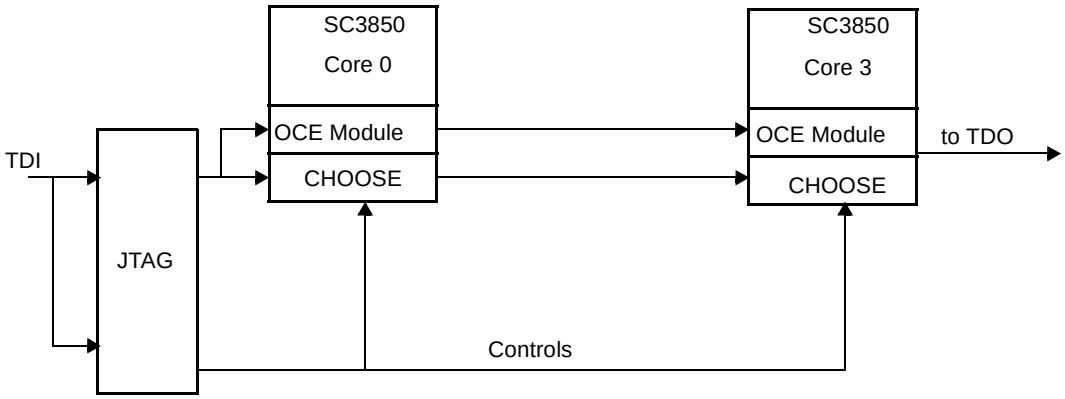


Figure 25-4. Cascading Multiple OCE Modules

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 MSC8157E, 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.

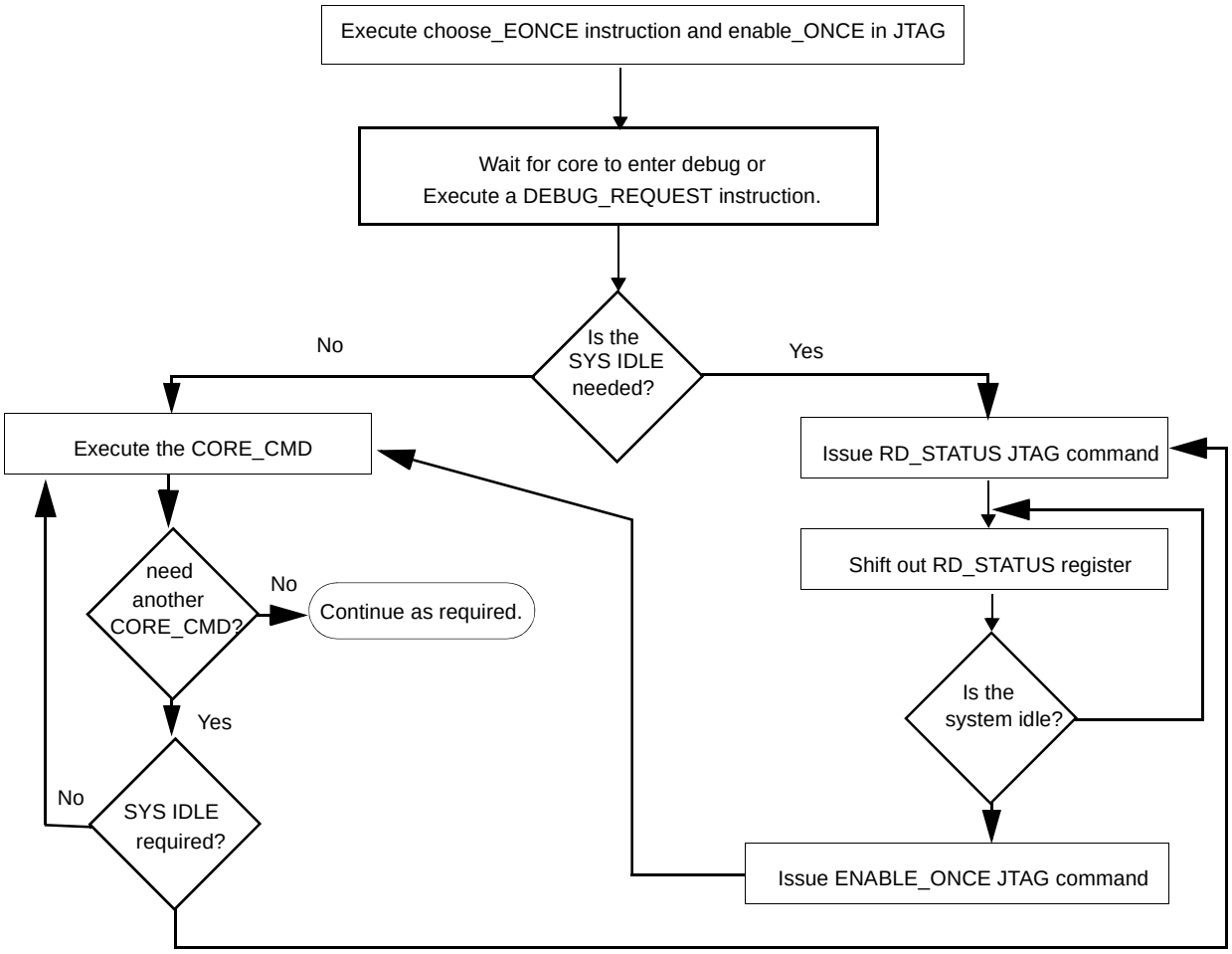


Figure 25-5. CORE_CMD and RD_STATUS Flow

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**.

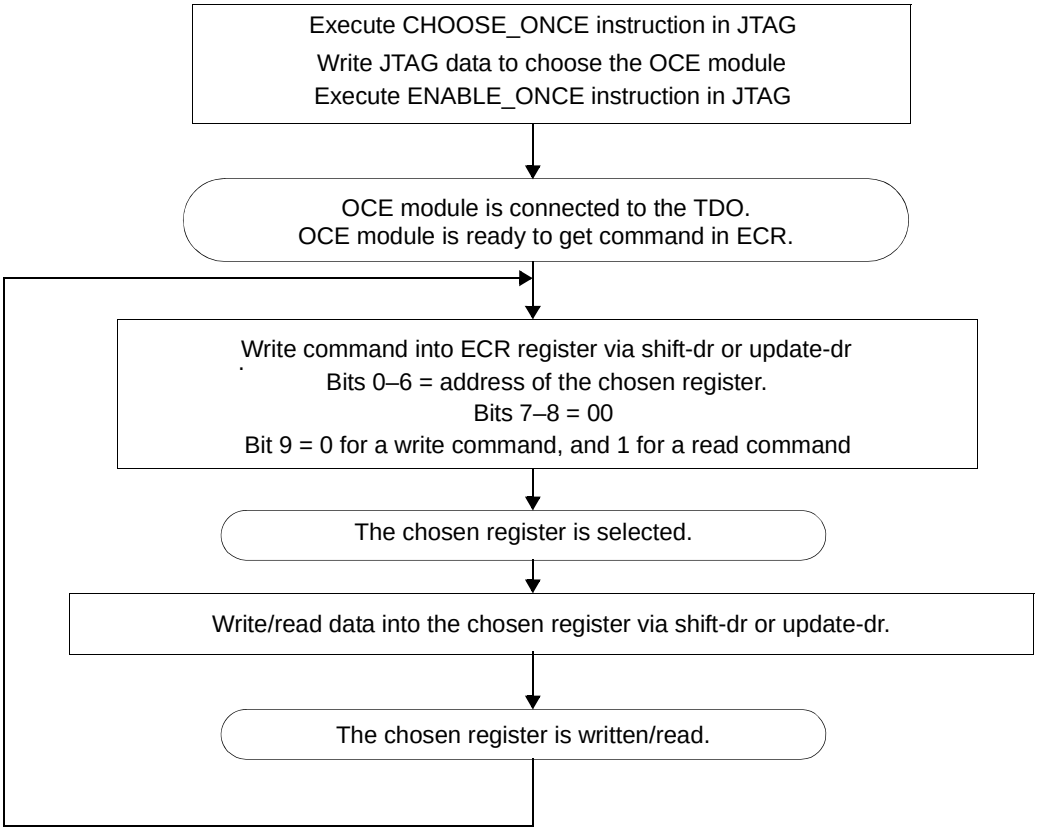


Figure 25-6. Reading and Writing OCE Registers Via the JTAG TAP

25.1.9 Signalling a Debug Request

The EE[0–1] signals connect to each of the MSC8157E 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 MSC8157E

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 trigger 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 MSC8157E 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 *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.

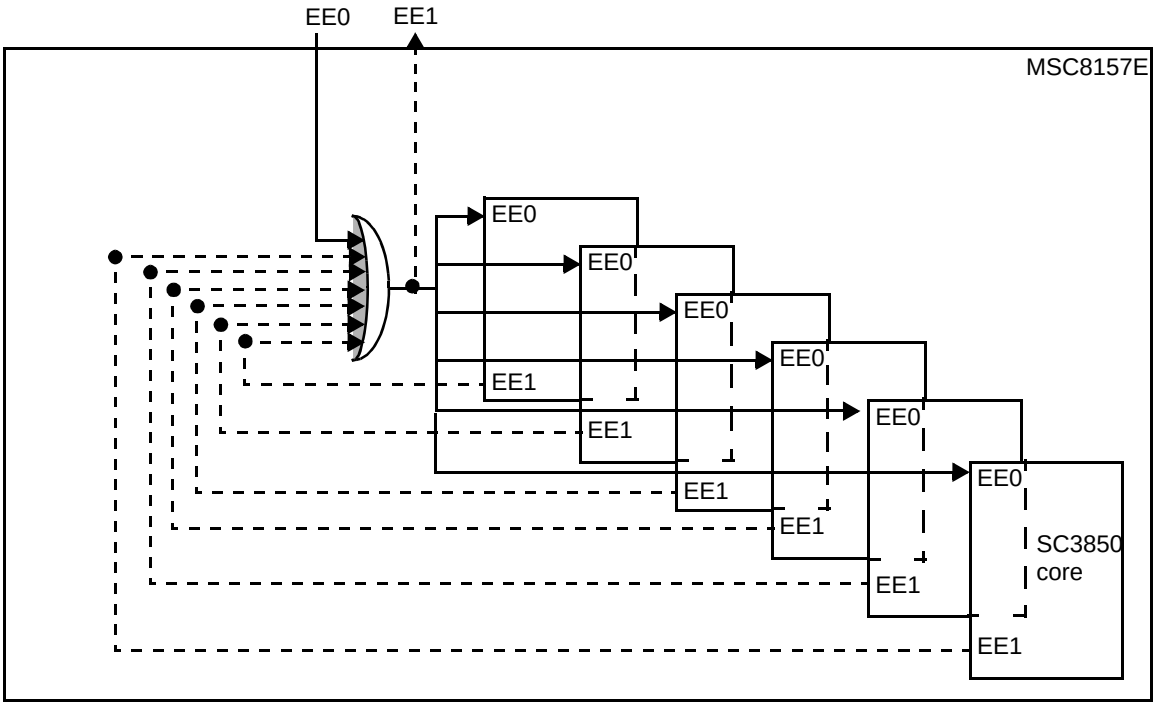


Figure 25-7. Selected SC3850 Core Issues a Debug Request to All Other SC3850 Cores

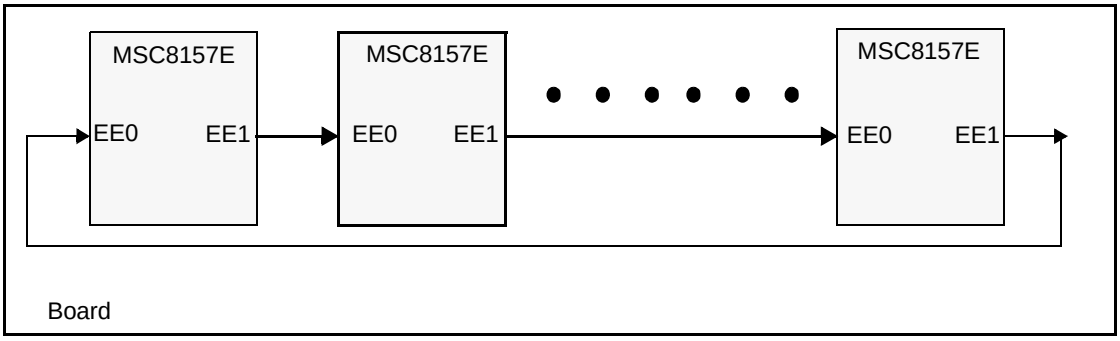


Figure 25-8. Board EE Signal Interconnectivity

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 MSC8157E 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 MSC8157E 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. 1149.6. 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	0	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 MSC8157E first mask set is 0b0000.
- Design Center number is 0b000110.
- Sequence Number for the MSC8157E is 0b0010010100.
- Manufacture identity is 0b00000001110.
- The final 1 is required by the **IEEE** Std. 1149.6.

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 MSC8157E BSR contains bits for most device signals and control signals. All MSC8157E 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 MSC8157E and are described by the BSDL file on the product website. **Figure 25-9** through **Figure 25-12** show various BSR cell types.

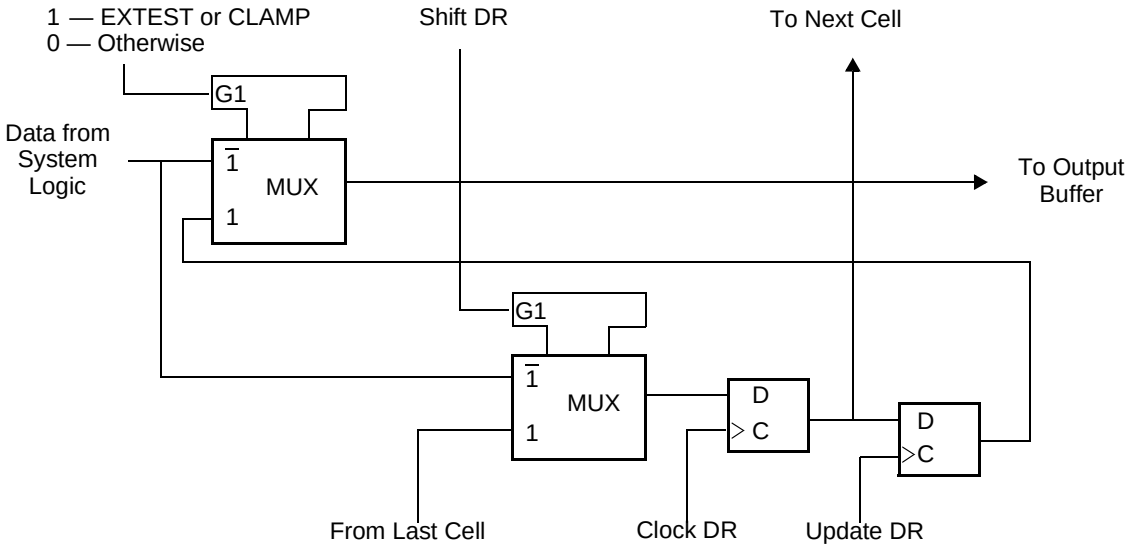


Figure 25-9. Output Signal Cell (O.PIN)

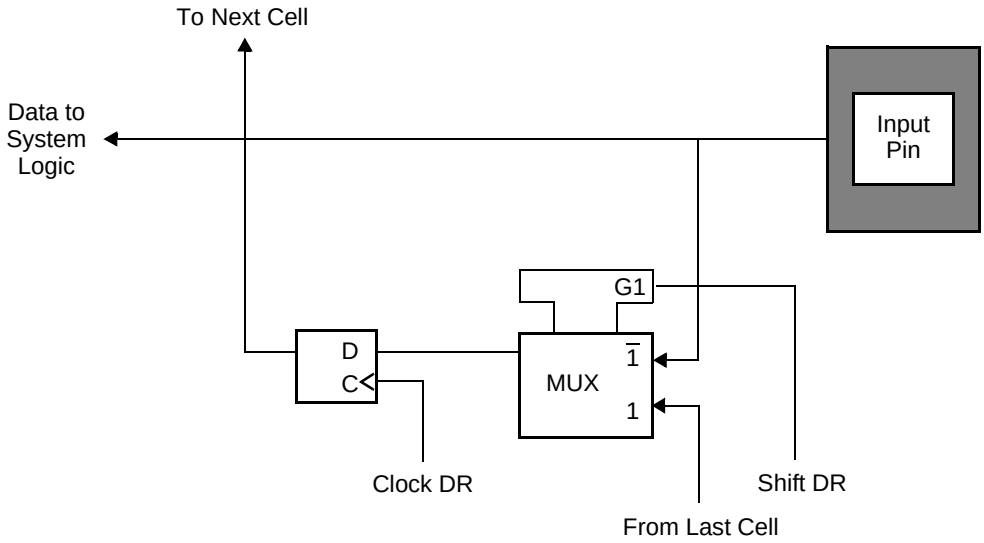


Figure 25-10. Observe-Only Input Signal Cell (I.OBS)

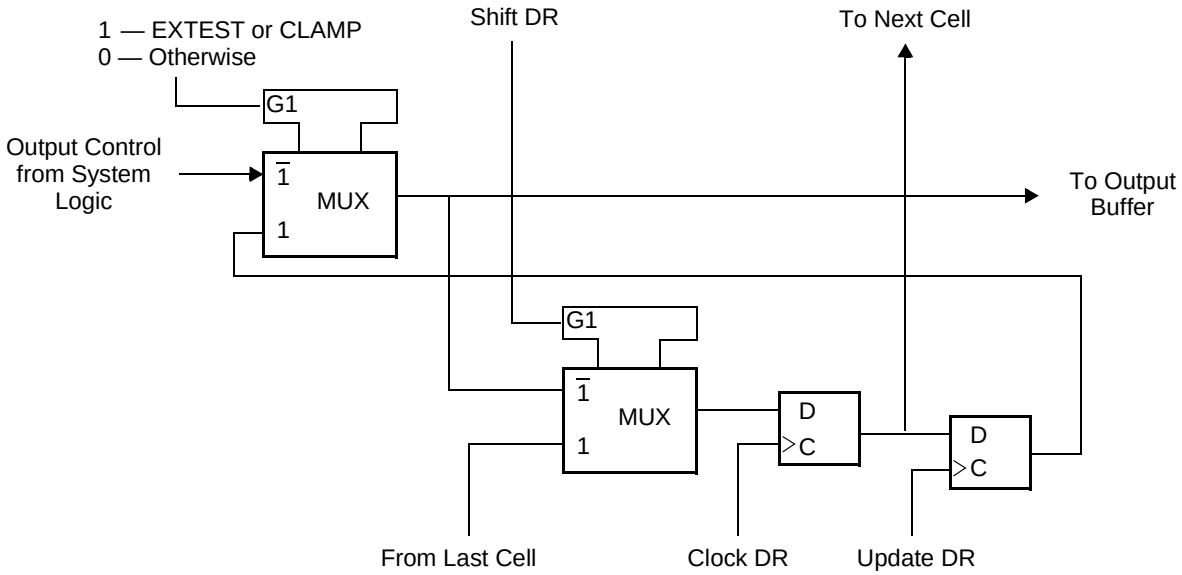


Figure 25-11. Output Control Cell (IO.CTL)

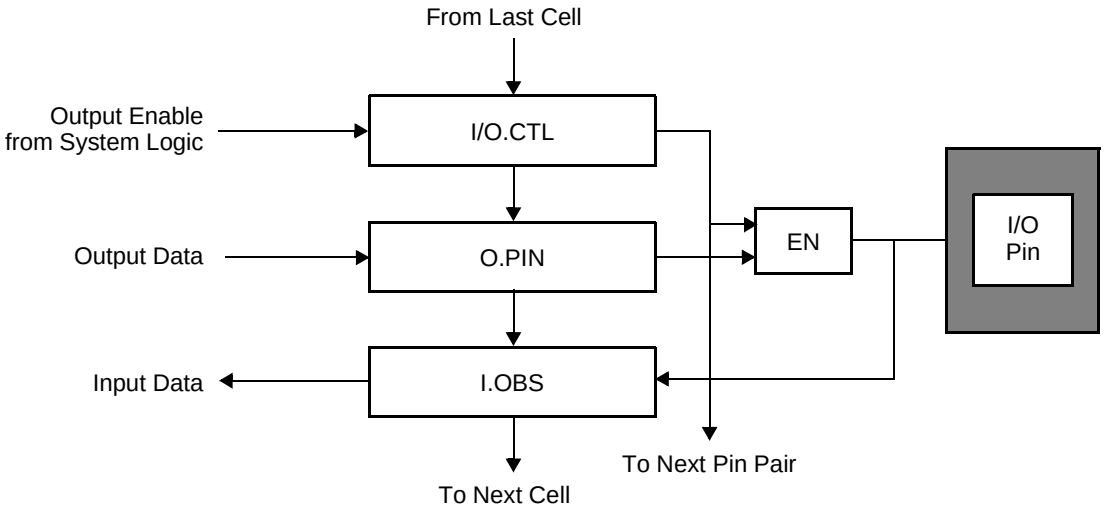


Figure 25-12. General Arrangement of Bidirectional Signal Cells

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 MSC8157E 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.

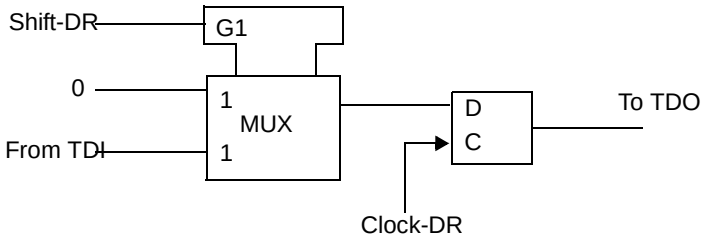


Figure 25-13. Bypass Register Configuration

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**.

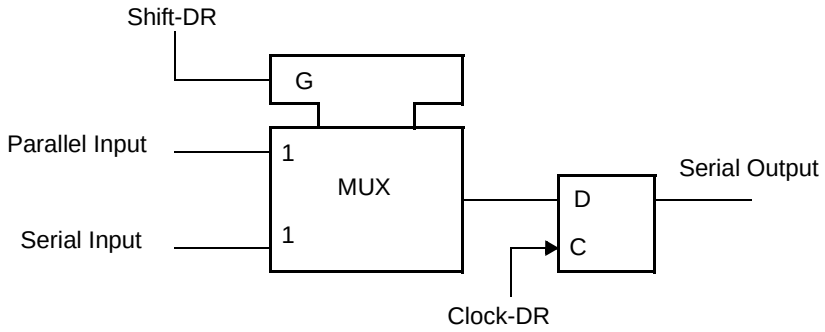


Figure 25-14. Identification Register Configuration (ID)

25.2 Debug and Profiling

The main debugging and profiling purposes are:

- Parallel counting of different events characterizing the operation of the MSC8157E device.
- Support for entering Debug mode upon detecting a predefined state of the MSC8157E 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 MSC8157E registers and memories in debug mode by a host processor.

25.2.1 Features

Table 25-6 describes MSC8157E device debug and profiling features

Table 25-6. MSC8157E Debug and Profiling Features

Block Name	Number of Blocks in MSC8157E	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	+	+	—	+

Table 25-6. MSC8157E Debug and Profiling Features

Block Name	Number of Blocks in MSC8157E	Supports Internal Debug	Supports Internal Profiling	Supports Profiling by PM Block	Supports Profiling and Debug by a CLASS Module
Class	2	+	+	—	+
PCI	1	+	—	—	+
RapidIO complex	1	+	—	+	+
eMSG complex	1	+	—	+	+
CPRI	1	+	—	+	+
L2/M2 memory	6	—	—	—	+
M3 memory	1	—	—	—	+
DDR memory	1	—	—	—	+
MAPLE-B2	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 MSC8157E provides access for all peripherals (QUICC Engine subsystem, RapidIO, PCI Express, 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 *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 MSC8157E 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 *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 *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 MSC8157E 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

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 19.2**, *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 the *QUICC Engine Block Reference Manual with Interworking Protocol* for details.

All serial interfaces (SGMII, Serial RapidIO, PCI Express, and CPRI) support digital loopback mode. See **Section 15.10.59** for details.

25.2.9 Serial RapidIO and eMSG Debug and Profiling

The SRIO/eMSG complex debugging system includes three error interrupts and the ability to connect to the profiling unit.

25.2.9.1 Debug Errors

The SRIO/eMSG complex asserts the error interrupts listed in **Table 25-9** when a system error is detected.

Table 25-9. eMSG Debug Interrupt

DSP core subsystem Interrupt Number	Description of Interrupt
86	ORed Serial RapidIO eMSG Queue Manager Receive Interrupt
87	ORed Serial RapidIO eMSG Queue Manager Transmit Interrupt
88	ORed Serial RapidIO eMSG Buffer Manager Interrupt

See **Chapter 16**, *Serial RapidIO Controller and Enhanced Message Complex* for details.

In addition, there are 27 eMSG-related interrupts that come out of the HSSI complex and are ORed in GIR7 and enabled for each core by GIER7_0 to GIER7_5.

25.2.9.2 Profiling Unit

All RapidIO/eMSG events can connect to Performance monitor (PM) block. See **Section 25.3**, *Performance Monitor* for details. The RapidIO PM uses the RapidIO clock.

25.2.10 CPRI Debug and Profiling

The CPRI system debugging system includes two error interrupts and the ability to connect to the profiling unit.

25.2.10.1 Debug Errors

The CPRI system asserts the error interrupt listed in **Table 25-10** when a system error is detected.

Table 25-10. CPRI Debug Interrupt

DSP core subsystem Interrupt Number	Description of Interrupt
89	ORed CPRI Receive Control Interrupt
90	ORed CPRI Transmit Control Interrupt

See **Section 17.3.10**, *Interrupts*, on page 17-54 for details.

25.2.11 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.

25.2.12 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. See the *SC3850 DSP Core Subsystem Reference Manual* DPU chapter for details.

25.3 Performance Monitor

The MSC8157E 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 MSC8157E 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. 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.

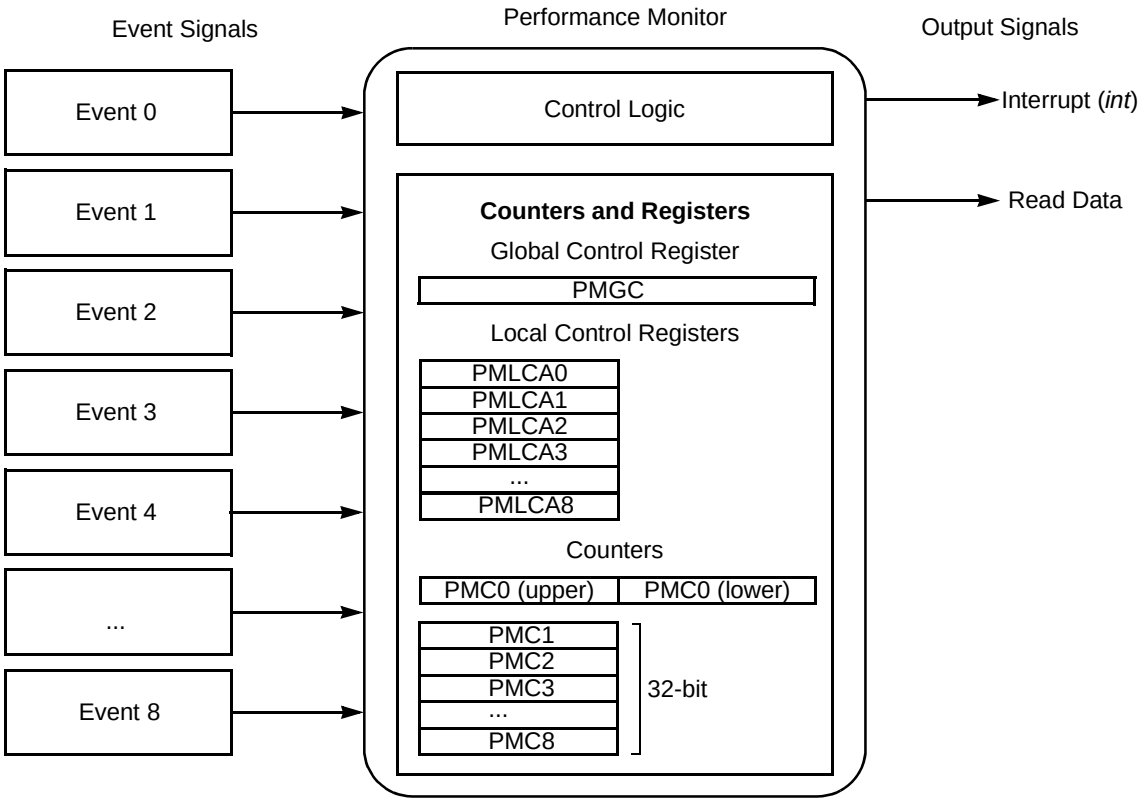


Figure 25-15. Performance Monitor Block Diagram

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 MSC8157E 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.
- Triggering and 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

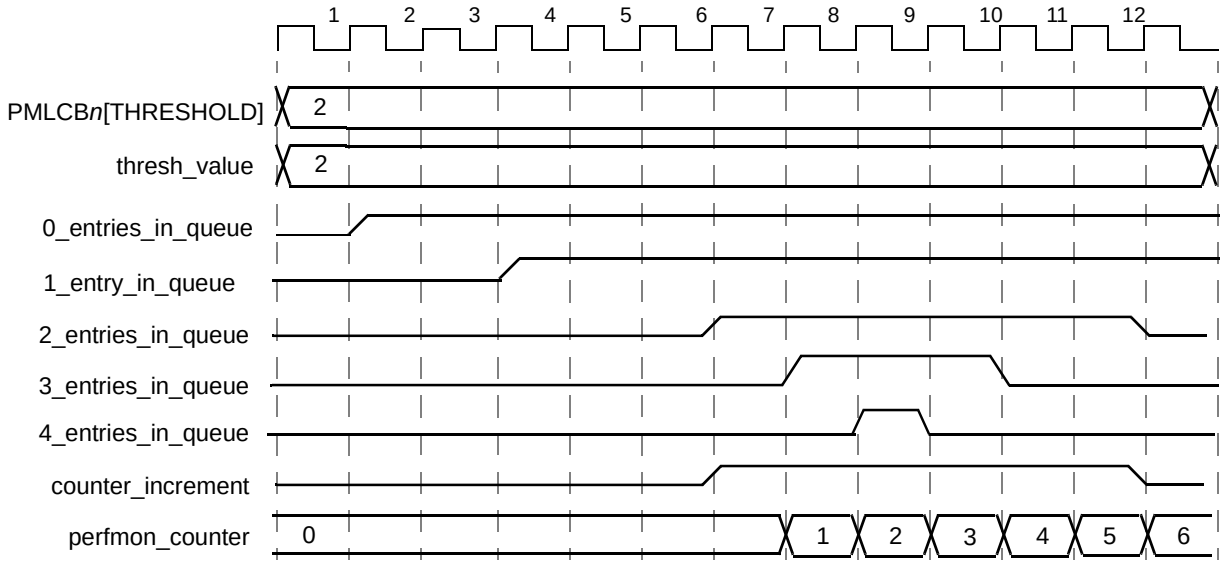
Using the control registers described in **Section 25.3.2**, the nine PMCs can count the occurrences of specific events. The 64-bit PMC0 is design to count only clock cycles. However, to provide flexibility, a total of 64 reference events can be counted on any of the 32-bit PMCs (PMC1–PMC8). Additionally, 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 25-11**, 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 25-11 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 25-11** 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 25-11. 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
BMLite Events		
.Write to external memory has completed for buffer pool 0 to 5.	C1:5	
free list write access has completed.	C2:5	
software portal (release ring buffer)	C3:5	
software portal (PI/CI update)	C4:5	
Read access from external Memory has completed for buffer pool 0 to 15.	C5:5	
free list read access completed.	C6:5	
software portal (acquire ring buffer) Read access completed	C7:5	
eMSG Events		
Event 31	C1:47	
Event 30	C2:47	
Event 29	C3:47	
Event 28	C4:47	
Event 27	C5:47	
Event 26	C6:47	
Event 25	C7:47	
Event 24	C8:47	
Event 23	C1:48	
Event 22	C2:48	
Event 21	C3:48	
Event 20	C4:48	
Event 19	C5:48	

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Event 18	C6:48	
Event 17	C7:48	
Event 16	C8:48	
Event 15	C1:49	
Event 14	C2:49	
Event 13	C3:49	
Event 12	C4:49	
Event 11	C5:49	
Event 10	C6:49	
Event 9	C7:49	
Event 8	C8:49	
Event 7	C1:50	
Event 6	C2:50	
Event 5	C3:50	
Event 4	C4:50	
Event 3	C5:50	
Event 28	C6:50	
Event 1	C7:50	
Event 0	C8:50	
HSSI General Event		
Consolidated ECC error signal from rio_top. (It includes the ECC error from srio(0,1),eMSG and DMA)	Ref:14	
SRIO1 Events		
A received packet has been sent to OCN for passthrough	C2:2	
OCN reorder occurred.	C3:2	
OCN reorder occurred, priority 0.	C4:7	
OCN reorder occurred, priority 1.	C5:7	
OCN reorder occurred, priority 2.	C6:7	
OCN reorder occurred priority 3.	C7:7	
Clock cycle occurred in which an OCN tag is unavailable, any priority. Event asserted for as many clock cycles as this is true.	C6:18	
Clock cycle occurred in which an OCN tag is unavailable, priority 0. Event asserted for as many clock cycles as this is true.	C7:18	
Clock cycle occurred in which an OCN tag is unavailable, priority 1. Event asserted for as many clock cycles as this is true.	C8:18	

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Clock cycle occurred in which an OCN tag is unavailable, priority 2. Event asserted for as many clock cycles as this is true.	C1:18	
Clock cycle occurred in which an OCN tag is unavailable, priority 3. Event asserted for as many clock cycles as this is true.	N/A	N/A
Inbound buffer 0 utilization	C1:20	
Inbound buffer 1 utilization	C2:20	
Inbound buffer 2 utilization	C3:20	
Inbound buffer 3 utilization	C4:20	
Inbound buffer 4 utilization	C5:20	
Inbound buffer 5 utilization	C6:20	
Inbound buffer 6 utilization	C7:20	
Inbound buffer 7 utilization	C8:20	
Inbound buffer 0, priority 0 utilization.	C1:21	
Inbound buffer 1, priority 0 utilization.	C2:21	
Inbound buffer 2, priority 0 utilization.	C3:21	
Inbound buffer 3, priority 0 utilization.	C4:21	
Inbound buffer 4, priority 0 utilization.	C5:21	
Inbound buffer 0, priority 1 utilization.	C1:22	
Inbound buffer 1, priority 1 utilization.	C2:22	
Inbound buffer 2, priority 1 utilization.	C3:22	
Inbound buffer 3, priority 1 utilization.	C4:22	
Inbound buffer 4, priority 1 utilization.	C5:22	
Inbound buffer 0, priority 2 utilization.	C1:23	
Inbound buffer 1, priority 2 utilization.	C2:23	
Inbound buffer 2, priority 2 utilization.	C3:23	
Inbound buffer 3, priority 2 utilization.	C4:23	
Inbound buffer 4, priority 2 utilization.	C5:23	
Inbound buffer 0, priority 3 utilization.	C1:24	
Inbound buffer 1, priority 3 utilization.	C2:24	
Inbound buffer 2, priority 3 utilization.	C3:24	
Inbound buffer 3, priority 3 utilization.	C4:24	
Inbound buffer 4, priority 3 utilization.	C5:24	
Packet sent to RapidIO	C4:2	
Packet was misaligned	C1:2	
Packet was retried	C6:2	
Packet was reordered	C8:2	

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
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	
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	
Clock cycle occurred in which a RapidIO tag is unavailable, any priority. Event asserted for as many clock cycles as this is true.	C1:55	
Clock cycle occurred in which a RapidIO tag is unavailable, priority 0. Event asserted for as many clock cycles as this is true.	C2:55	
Clock cycle occurred in which a RapidIO tag is unavailable, priority 1. Event asserted for as many clock cycles as this is true.	C3:55	
Clock cycle occurred in which a RapidIO tag is unavailable, priority 2. Event asserted for as many clock cycles as this is true.	C4:55	
Clock cycle occurred in which a misaligned RapidIO tag is unavailable, any priority. Event asserted for as many clock cycles as this is true.	C3:57	
Clock cycle occurred in which a misaligned RapidIO tag is unavailable, priority 0. Event asserted for as many clock cycles as this is true.	C4:57	
Clock cycle occurred in which a misaligned RapidIO tag is unavailable, priority 1. Event asserted for as many clock cycles as this is true.	C5:57	
Clock cycle occurred in which a misaligned RapidIO tag is unavailable, priority 2. Event asserted for as many clock cycles as this is true.	C6:57	
Outbound buffer 0 utilization	C1:25	
Outbound buffer 1 utilization	C2:25	
Outbound buffer 2 utilization	C3:25	
Outbound buffer 3 utilization	C4:25	

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Outbound buffer 4 utilization	C5:25	
Outbound buffer 5 utilization	C6:25	
Outbound buffer 6 utilization	C7:25	
Outbound buffer 7 utilization	C8:25	
Outbound buffer 0, priority 0 utilization	C1:26	
Outbound buffer 1, priority 0 utilization	C2:26	
Outbound buffer 2, priority 0 utilization	C3:26	
Outbound buffer 3, priority 0 utilization	C4:26	
Outbound buffer 4, priority 0 utilization	C5:26	
Outbound buffer 0, priority 1 utilization	C1:27	
Outbound buffer 1, priority 1 utilization	C2:27	
Outbound buffer 2, priority 1 utilization	C3:27	
Outbound buffer 3, priority 1 utilization	C4:27	
Outbound buffer 4, priority 1 utilization	C5:27	
Outbound buffer 0, priority 2 utilization	C1:28	
Outbound buffer 1, priority 2 utilization	C2:28	
Outbound buffer 2, priority 2 utilization	C3:28	
Outbound buffer 3, priority 2 utilization	C4:28	
Outbound buffer 4, priority 2 utilization	C5:28	
Outbound buffer 0, priority 3 utilization	C1:29	
Outbound buffer 1, priority 3 utilization	C2:29	
Outbound buffer 2, priority 3 utilization	C3:29	
Outbound buffer 3, priority 3 utilization	C4:29	
Outbound buffer 4, priority 3 utilization	C5:29	
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	
Non idles received. This can be used to determine the RapidIO link utilization. This is actually 1/2 of the actual count.	C4:3	
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	

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Clock cycle occurred in which the inbound buffer is full to any priority (measured from when EOP received to EOP transferred on OCN). 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 (measured from when EOP received to EOP transferred on OCN). Event asserted for as many clock cycles as this is true.	C3:13	
Clock cycle occurred in which the inbound buffer is full to priority 1 (measured from when EOP received to EOP transferred on OCN). 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 (measured from when EOP received to EOP transferred on OCN). 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 (measured from when EOP received to EOP transferred on OCN). Event asserted for as many clock cycles as this is true.	C6:13	
Non idles transmitted. This can be used to determine the RapidIO link utilization. This is actually 1/2 of the actual count.	C3:3	
ACK History Queue utilization. When this event is selected, each bit corresponds to the number of ackIDs that have not been acknowledged. If all 8 bits are asserted, then no ackID is available and a packet has been stalled.	Ref:33	
OCN DMA0 Events		
DMA Channel 0 Read request	C1:0	
DMA Channel 1 Read request	C2:0	
DMA Channel 2 Read request	C3:0	
DMA Channel 3 Read request	C4:0	
DMA Channel 0 Write request	C6:53	
DMA Channel 1 Write request	C7:53	
DMA Channel 2 Write request	C8:53	
DMA Channel 3 Write request	C5:53	
DMA Channel 0 Dsec request	C7:55	
DMA Channel 1 Dsec request	C8:54	
DMA Channel 2 Dsec request	C5:54	
DMA Channel 3 Dsec request	C6:54	
Channel 0 Read DW	C8:61	
Channel 1 Read DW	C5:61	
Channel 2 Read DW	C6:61	
Channel 3 Read DW	C7:61	
Channel 0 Write DW	C2:62	
Channel 1 Write DW	C3:62	
Channel 2 Write DW	C4:62	

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Channel 3 Write DW	C1:62	
SRIO2 Events		
A received packet has been sent to OCN for passthrough	C3:1	
OCN reorder occurred.	C6:1	
OCN reorder occurred, priority 0.	C7:10	
OCN reorder occurred, priority 1.	C8:10	
OCN reorder occurred, priority 2.	C1:10	
OCN reorder occurred priority 3.	C2:10	
Clock cycle occurred in which an OCN tag is unavailable, any priority. Event asserted for as many clock cycles as this is true.	C7:19	
Clock cycle occurred in which an OCN tag is unavailable, priority 0. Event asserted for as many clock cycles as this is true.	C8:19	
Clock cycle occurred in which an OCN tag is unavailable, priority 1. Event asserted for as many clock cycles as this is true.	C1:19	
Clock cycle occurred in which an OCN tag is unavailable, priority 2. Event asserted for as many clock cycles as this is true.	C2:19	
Clock cycle occurred in which an OCN tag is unavailable, priority 3. Event asserted for as many clock cycles as this is true.	c3:19	
Clock cycle occurred in which an OCN tag is unavailable, priority 3. Event asserted for as many clock cycles as this is true.	N/A	N/A
Inbound buffer 0 utilization	C1:33	
Inbound buffer 1 utilization	C2:33	
Inbound buffer 2 utilization	C3:33	
Inbound buffer 3 utilization	C4:33	
Inbound buffer 4 utilization	C5:33	
Inbound buffer 5 utilization	C6:33	
Inbound buffer 6 utilization	C7:33	
Inbound buffer 7 utilization	C8:33	
Inbound buffer 0, priority 0 utilization.	C1:34	
Inbound buffer 1, priority 0 utilization.	C2:34	
Inbound buffer 2, priority 0 utilization.	C3:34	
Inbound buffer 3, priority 0 utilization.	C4:34	
Inbound buffer 4, priority 0 utilization.	C5:34	
Inbound buffer 0, priority 1 utilization.	C1:35	
Inbound buffer 1, priority 1 utilization.	C2:35	

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Inbound buffer 2, priority 1 utilization.	C3:35	
Inbound buffer 3, priority 1 utilization.	C4:35	
Inbound buffer 4, priority 1 utilization.	C5:35	
Inbound buffer 0, priority 2 utilization.	C1:36	
Inbound buffer 1, priority 2 utilization.	C2:36	
Inbound buffer 2, priority 2 utilization.	C3:36	
Inbound buffer 3, priority 2 utilization.	C4:36	
Inbound buffer 4, priority 2 utilization.	C5:36	
Inbound buffer 0, priority 3 utilization.	C1:37	
Inbound buffer 1, priority 3 utilization.	C2:37	
Inbound buffer 2, priority 3 utilization.	C3:37	
Inbound buffer 3, priority 3 utilization.	C4:37	
Inbound buffer 4, priority 3 utilization.	C5:37	
Packet sent to RapidIO	C7:1	
Packet was misaligned	C2:1	
Packet was retried	C4:1	
Packet was reordered	C1:1	
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	
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	
Clock cycle occurred in which a RapidIO tag is unavailable, any priority. Event asserted for as many clock cycles as this is true.	C2:56	
Clock cycle occurred in which a RapidIO tag is unavailable, priority 0. Event asserted for as many clock cycles as this is true.	C3:56	

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Clock cycle occurred in which a RapidIO tag is unavailable, priority 1. Event asserted for as many clock cycles as this is true.	C4:56	
Clock cycle occurred in which a RapidIO tag is unavailable, priority 2. Event asserted for as many clock cycles as this is true.	C5:56	
Clock cycle occurred in which a misaligned RapidIO tag is unavailable, any priority. Event asserted for as many clock cycles as this is true.	C4:58	
Clock cycle occurred in which a misaligned RapidIO tag is unavailable, priority 0. Event asserted for as many clock cycles as this is true.	C5:58	
Clock cycle occurred in which a misaligned RapidIO tag is unavailable, priority 1. Event asserted for as many clock cycles as this is true.	C6:58	
Clock cycle occurred in which a misaligned RapidIO tag is unavailable, priority 2. Event asserted for as many clock cycles as this is true.	C7:58	
Outbound buffer 0 utilization	C1:38	
Outbound buffer 1 utilization	C2:38	
Outbound buffer 2 utilization	C3:38	
Outbound buffer 3 utilization	C4:38	
Outbound buffer 4 utilization	C5:38	
Outbound buffer 5 utilization	C6:38	
Outbound buffer 6 utilization	C7:38	
Outbound buffer 7 utilization	C8:38	
Outbound buffer 0, priority 0 utilization	C1:39	
Outbound buffer 1, priority 0 utilization	C2:39	
Outbound buffer 2, priority 0 utilization	C3:39	
Outbound buffer 3, priority 0 utilization	C4:39	
Outbound buffer 4, priority 0 utilization	C5:39	
Outbound buffer 0, priority 1 utilization	C1:40	
Outbound buffer 1, priority 1 utilization	C2:40	
Outbound buffer 2, priority 1 utilization	C3:40	
Outbound buffer 3, priority 1 utilization	C4:40	
Outbound buffer 4, priority 1 utilization	C5:40	
Outbound buffer 0, priority 2 utilization	C1:41	
Outbound buffer 1, priority 2 utilization	C2:41	
Outbound buffer 2, priority 2 utilization	C3:41	
Outbound buffer 3, priority 2 utilization	C4:41	
Outbound buffer 4, priority 2 utilization	c5:41	

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Outbound buffer 0, priority 3 utilization	c1:52	
Outbound buffer 1, priority 3 utilization	C2:52	
Outbound buffer 2, priority 3 utilization	C3:52	
Outbound buffer 3, priority 3 utilization	C4:52	
Outbound buffer 4, priority 3 utilization	C5:52	
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	
Non idles received. This can be used to determine the RapidIO link utilization. This is actually 1/2 of the actual count.	C2:3	
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 (measured from when EOP received to EOP transferred on OCN). 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 (measured from when EOP received to EOP transferred on OCN). 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 (measured from when EOP received to EOP transferred on OCN). 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 (measured from when EOP received to EOP transferred on OCN). Event asserted for as many clock cycles as this is true.	C6:14	
Clock cycle occurred in which the inbound buffer is full to priority 3 (measured from when EOP received to EOP transferred on OCN). Event asserted for as many clock cycles as this is true.	C7:14	
Non idles transmitted. This can be used to determine the RapidIO link utilization. This is actually 1/2 of the actual count.	C1:3	
ACK History Queue utilization. When this event is selected, each bit corresponds to the number of ackIDs that have not been acknowledged. If all 8 bits are asserted, then no ackID is available and a packet has been stalled.	ref:34	
OCN DMA1 Events		
DMA Channel 0 Read request	C5:0	

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
DMA Channel 1 Read request	C6:0	
DMA Channel 2 Read request	C7:0	
DMA Channel 3 Read request	C8:0	
DMA Channel 0 Write request	C2:53	
DMA Channel 1 Write request	C3:53	
DMA Channel 2 Write request	C4:53	
DMA Channel 3 Write request	C1:53	
DMA Channel 0 Dsec request	C3:54	
DMA Channel 1 Dsec request	C4:54	
DMA Channel 2 Dsec request	C1:54	
DMA Channel 3 Dsec request	C2:54	
Channel 0 Read DW	C4:61	
Channel 1 Read DW	C1:61	
Channel 2 Read DW	C2:61	
Channel 3 Read DW	C3:61	
Channel 0 Write DW	C6:62	
Channel 1 Write DW	C7:62	
Channel 2 Write DW	C8:62	
Channel 3 Write DW	C5:62	
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 IOQ 1/4 full. When asserted indicates that the IOQ is 1/4 full.	C3:32	
PM IOQ 1/2 full. When asserted indicates that the IOQ is 1/2 full.	C4:32	
PM IOQ 3/4 full. When asserted indicates that the IOQ is 3/4 full.	C7:30	
PM IOQ full. When asserted indicates that the IOQ is full.	C8:30	
PM IOQ entry 0 or 6 start. When asserted indicates that this entry is now occupied.	Ref:29	
PM IOQ entry 0 or 6 stop. When asserted indicates that this entry is now unoccupied.	Ref:29	
PM IG2PI read. When asserted indicates that the controller has received an inbound PEX read packet.	C1:30	

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
PM IG2PI write. When asserted indicates that the controller has received an inbound PEX write packet.	C2:30	
PM IG2PI data valid. When asserted indicates that the controller is receiving inbound PEX data.	C3:30	
PM outbound OCN read request. When asserted indicates that the bridge has received an OCN read packet.	C8:32	
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	
PM OOQ 1/4 full. When asserted indicates that the OOQ is 1/4 full.	C5:31	
PM OOQ 1/2 full. When asserted indicates that the OOQ is 1/2 full.	C6:31	
PM OOQ 3/4 full. When asserted indicates that the OOQ is 3/4 full.	C1:31	
PM OOQ full. When asserted indicates that the OOQ is full.	C2:31	
PM OOQ entry 0 start. When asserted indicates that this entry is now occupied.	Ref:30	
PM OOQ entry 0 stop. When asserted indicates that this entry is now unoccupied.	Ref:30	
PM OG2PI read. When asserted indicates that the bridge is sending an outbound PCI Express read packet.	C7:31	
PM OG2PI write. When asserted indicates that the bridge is sending an outbound PCI Express write packet.	C8:31	
PM OG2PI data valid. When asserted indicates that the bridge is sending an outbound PCI Express data.	c5:32	
CPRI Events		
Write priority 2	c1:46	
Write EOD	c1:57	
Read idle	c1:58	
Write EOT	c2:57	
Read request pending	c2:58	
Write priority 3	c2:59	
Read EOD	c3:58	
Write priority 1	c4:19	
Write request qual	c4:30	
Read priority 0	c4:43	
Read EOF	c5:60	
Write data ACK	c6:34	

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Write priority 0	c6:40	
Read priority 1	c6:41	
Read data ACK	c7:34	
Write pending request	c7:40	
Read priority 2	c7:41	
Read request qual	c8:34	
Write idle	c8:40	
Read priority 3	c8:41	
CPRI1 Read request qual	c1:6	
CPRI1 Read data ACK	c2:6	
CPRI1 Read EOD	c3:6	
CPRI1 Read EOT	c4:6	
CPRI1 Write request qual	c5:6	
CPRI1 Write data ACK	c6:6	
CPRI1 Write EOD	c7:6	
CPRI1 Write EOT	c8:6	
CPRI2 Read request qual	c1:17	
CPRI2 Read data ACK	c2:17	
CPRI2 Read EOD	c3:17	
CPRI2 Read EOT	c4:17	
CPRI2 Write request qual	c5:17	
CPRI2 Write data ACK	c6:17	
CPRI2 Write EOD	c7:17	
CPRI2 Write EOT	c8:17	
CPRI3 Read request qual	c1:7	
CPRI3 Read data ACK	c2:8	
CPRI3 Read EOD	c3:9	
CPRI3 Read EOT	c4:10	
CPRI3 Write request qual	c5:11	
CPRI3 Write data ACK	c6:12	
CPRI3 Write EOD	c7:13	
CPRI3 Write EOT	c8:14	
CPRI4 Read request qual	c1:51	
CPRI4 Read data ACK	c2:51	
CPRI4 Read EOD	c3:51	
CPRI4 Read EOT	c4:51	
CPRI4 Write request qual	c5:51	

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
CPRI4 Write data ACK	c6:51	
CPRI4 Write EOD	c7:51	
CPRI4 Write EOT	c8:51	
CPRI5 Read request qual	c1:63	
CPRI5 Read data ACK	c2:63	
CPRI5 Read EOD	c3:63	
CPRI5 Read EOT	c4:63	
CPRI5 Write request qual	c5:63	
CPRI5 Write data ACK	c6:63	
CPRI5 Write EOD	c7:63	
CPRI5 Write EOT	c8:63	
CPRI6 Read request qual	c1:8	
CPRI6 Read data ACK	c2:9	
CPRI6 Read EOD	c3:10	
CPRI6 Read EOT	c4:11	
CPRI6 Write request qual	c5:12	
CPRI6 Write data ACK	c6:11	
CPRI6 Write EOD	c7:11	
CPRI6 Write EOT	c8:12	
DDR Events		
Number of pipelined reads that missed in the row open table	c1:13	edge_detect
Total number of row open table hits for reads and writes from core	c1:14	edge_detect
Total number of row open table hits for reads and writes from ethernet(0,1,2,3)	c2:14	edge_detect
Number of pipelined reads or writes that missed in the row open table	c2:15	edge_detect
Total number of row open table hits	c2:16	edge_detect
Total number of reads or writes from core	c2:4	edge_detect
Total number of row open table hits for reads and writes from PCI (0,1)	c3:16	edge_detect
Total number of reads or writes from ethernet (0,1,2,3).	c3:4	edge_detect
Total number of force page closings	c3:7	edge_detect
Number of nonpipelined reads that missed in the row open table	c3:8	edge_detect
Total number of row open table hits for reads and writes from high speed interface(hyper transport,PCI-Express,SRIO)	c4:16	edge_detect
Number of read modify write transactions due to ECC	c4:18	edge_detect
Total number of reads or wites from PCI(1,2)	c4:4	edge_detect

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Number of non pipelined reads or writes that missed in the row open table	c4:9	edge_detect
Number of cycles a read is returning data from DRAM.	c5:10	edge_detect
Total number of row open table hits for reads and writes from DMA.	c5:19	edge_detect
Total number of reads or writes from hign speed interfaces(Hyper transport, PCI-Express,SRIO).	c5:4	edge_detect
Number of pipelined reads that hit in the row open table	c5:9	edge_detect
Number of pipelined reads or writes that hit in the row open table	c6:10	edge_detect
Total number of row open table hits for reads and writes from security	c6:19	edge_detect
Number of cycles a write is sending data to DRAM	c6:3	edge_detect
Total number of reads or writes from DMA	c6:4	edge_detect
Number of non pipelined reads that hit in the row open table	c7:12	edge_detect
Input queue is full and ipm_req is asserted	c4:59	edge_detect
Total number of row open table misses	c7:3	edge_detect
Total number of reads or writes from security	c7:4	edge_detect
Number of non pipelined reads or writes that hit in the row open table	c8:13	edge_detect
Number of forced page closings not caused by a refresh	c8:3	edge_detect
All counters are valid	c8:4	edge_detect
Forced page closings due to collision with bank and sub bank.	c5:18	edge_detect
MBus Plus DDR Events		
	ref:32	edge_detect
There is at least 1 pending request	c3:15	sync
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.

Table 25-11. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
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 25-12 contains sample register settings for the three supported modes.

- Simple event performance monitoring example
- Threshold event performance monitoring example

The settings in **Table 25-12** are identical for all three examples.

Table 25-12. 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 25-13. 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

Table 25-13. Register Settings for Counting Examples (Continued)

Register	Register Field	Simple Event	Threshold
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 25-48**.
- Performance Monitor Local Control Register A0 (PMLCA0), see **page 25-49**.
- Performance Monitor Local Control Register A[1–8] (PMLCA[1–8]), see **page 25-50**.
- Performance Monitor Local Control Register B0 (PMLCB0), see **page 25-51**.
- Performance Monitor Local Control Register B[1–8] (PMLCB[1–8]), see **page 25-52**.
- Performance Monitor Counter 0 (PMC0), see **page 25-51**.
- Performance Monitor Counter 1–8 (PMC[1–8]), see **page 25-52**.

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 25-14** defines the PMGC bit fields.

Table 25-14. 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, along with PMLCB0, configures the PMC0 operation. **Table 25-15** defines the PMLCA0 bit fields.

Table 25-15. 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. 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 25-16** defines the PMLCA[1–8] bit fields.

Table 25-16. 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 25-11**.
— 15–0	0	Reserved. Write to zero for future compatibility.	

25.3.2.4 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 and PMLCB0. **Table 25-17** defines the PMC0 bit fields.

Table 25-17. PMC0 Bit Descriptions

Name	Reset	Description	Settings
PMC0 63–0	0	**Performance Monitor Counter 0** Contains the current PMC0 count.	

25.3.2.5 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 25-18** defines the PMC[1–8] bit fields.

Table 25-18. PMC[1–8] Bit Descriptions

Name	Reset	Description	Settings
PMCn 31–0	0	**Performance Monitor Counter n** Contains the current PMCn count.	

26 Multi Accelerator Platform Engine, Baseband 2 (MAPLE-B2)

The MAPLE-B2 is a multi-standard baseband algorithm accelerator for Channel Decoding/Encoding, Fourier Transforms, UMTS chip rate processing, OFDMA and SC-FDMA equalization and CRC algorithms. The MAPLE-B2 consists of second generation Programmable-System-Interface (PSIF2), which is a programmable controller with DMA capabilities, and the following ten accelerators:

- *Enhanced Turbo/Viterbi Processing-Element (eTVPE).* Accelerates Turbo Decoding, Rate-De-Matching and HARQ combining.
- *Three enhanced FFT/DFT Processing-Elements (eFTPEs).* Accelerate various sizes of Fourier transforms and various pre/post transform processing.
- *Turbo Encoder Processing-Element (DEPE).* Accelerates Turbo Encoding and Rate Matching.
- *Equalization Processing-Element (EQPE).* Accelerates MMSE and MLD types of equalization algorithms and Matrix Inversion.
- Chip Rate Processing-Element (CRPE). Accelerates Downlink and Uplink UMTS chip rate processing.
- CRC Processing-Element (CRCPE). Accelerates CRC attachment or check.
- Code Generation Processing Element (CGPE). Accelerates UMTS code generation.
- Convolution and Filtering Virtual Processing Element (CONVPE). Accelerates convolutions and filtering in frequency domain by combining the eFTPE and EQPE.

In addition to the the accelerators, MAPLE-B2 provides options of internal chaining of multiple PE's using single job descriptor:

- Turbo Decoding and Re-encoding including rate matching uses a combination of eTVPE and DEPE.
- MMSE equalization combined with DFT and/or IDFT functions uses eFTPE and EQPE.

The PSIF2 integrates four 128-bit wide MBus master ports used by the MAPLE-B2 to transfer input and output data to and from system memory. It also integrates two 128-bit MBus Slave ports used for job descriptors access by Host and antenna data access by antenna interfaces. See **Section 26.4.3.1**, *MAPLE-B2 Second Generation Programmable System Interface (PSIF2)*, on page 26-23 for details.

26.1 Information Organization

Because the MAPLE-B2 is a complex module, this section provides a detailed guide as to the organization of the information in this chapter. The following sections contain the described information:

- MAPLE-B2 features (see **Section 26.2**). Includes a table that correlates broadband protocols with their respective MAPLE-B2 processing elements (PEs).
- Modes of operation (see **Section 26.3**). Defines the MAPLE-B2 operating modes.
- Function description (see **Section 26.4**). Includes the following information:
  - Initialization (see **Section 26.4.1**). Describes the use of the MAPLE-B2 application programmer interface (API).
  - Configuration and Control (see **Section 26.4.2**). Describes RAM-based parameters, descriptors (includes buffer descriptor (BD) rings, BDs, task and other descriptors), and registers used to configure, control, and monitor MAPLE-B2 and internal module operation. This section does not include detailed structure and programming information, described in detail in **Section 26.5**, *Programming Model*, on page 26-315.
  - Internal modules. Describes the functional information, including general operation and flow, input and output data and control structures (organized by specific modes) for each of the following modules:
    - PSIF2. Includes quad-RISC processor for system interface, handshake protocols and low-level control of PE's. See **Section 26.4.3.1**, *MAPLE-B2 Second Generation Programmable System Interface (PSIF2)*, on page 26-23.
    - eTVPE. See **Section 26.4.3.2**, *eTVPE HARQ, Rate De-Matching, and Turbo/Viterbi Decoding Flow,* on page 26-28.
    - eFTPE. See **Section 26.4.3.3**, *eFTPE FFT/iFFT/DFT/iDFT Operation*, on page 26-78.
    - DEPE. See **Section 26.4.3.4**, *DEPE Downlink Turbo Encoding Operation*, on page 26-117.
    - EQPE. See **Section 26.4.3.5**, *EQPE Equalization Processing Operation*, on page 26-144.
    - CRPE. See **Section 26.4.3.6**, *CRPE Uplink/Downlink UMTS Chip Rate Processing Operation*, on page 26-197. The functional description is divided into three parts:
      - **Section 26.4.3.6.1**, *Chip Rate Uplink Batch Processing Element*, on page 26-198
      - **Section 26.4.3.6.2**, *Chip Rate Uplink Fast Operation*, on page 26-234
      - **Section 26.4.3.6.3**, *Chip Rate Downlink Processing Operation*, on page 26-266
    - CRCPE. See **Section 26.4.3.7**, *CRC Operation*, on page 26-285
    - CGPE. See **Section 26.4.3.8**, *Code Generation Operation*, on page 26-290

- CONVPE. See **Section 26.4.3.9**, *CONVPE Convolution and Correlation Processing Operation*, on page 26-298

— External Master support using Serial RapidIO doorbells. See **Section 26.4.4**, *External Masters Support Using Serial RapidIO Doorbell*, on page 26-308
— MAPLE-B2 internal task control. See **Section 26.4.5**, *MAPLE-B2 Internal Task Control*, on page 26-312.
— MAPLE-B2 dynamic power gating. See **Section 26.4.6**, *MAPLE-B2 Power Gating Scheme*, on page 26-313
— MAPLE-B2 reset and initialization. See **Section 26.4.7**, *Reset*, on page 26-315.

- Programming model (see **Section 26.5**). Provides detailed descriptions of RAM-based configuration, control, and monitoring structures, including:
  — Initialization parameters (performed by the MAPLE-API)
  — Configuration and control parameters (configured by the host)
  — Descriptors (configured by the host)
  — Registers (configured via the SBus)

26.2 MAPLE-B2 Features

Table 26-1 lists the MAPLE-B2 support for specific communication protocol standards and how the individual PEs contribute to that support. **Table 26-2** lists the non-standard-related processing features of the individual PEs. **Table 26-3** lists the chained processing elements features (Virtual PEs). **Table 26-4** is a list of general MAPLE-B2 features.

Table 26-1. Support for Specific Protocol Standards

Protocol Standard	PE	Features
3GLTE (3GPP TS 36.212)	eTVPE	• Turbo decoding as specified in section 5.1.2.2. • Multiple input data structures support for multiple pre-processing options: —Support for HARQ scheme as specified in section 5.1.2.2: – Combining accumulator with new received punctured Code Block. – Outputting of new accumulator buffer for future HARQ transmissions. – 3 programmable weights. —Support of Sub-Block de-interleaving processing. • Multiple options of decoding output results: —Hard output results —2 optional types of Soft Output results: – 16 bits LLR of Aposteriori data for systematic bits only – 8 bits LLR of Aposteriori data for systematic, parities and tail bits. • 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 • Optional Code Block CRC check on Hard Output results. • Optional Transport Block CRC check on Hard Output results if Transport Block size $\leq$ 6120. • Three different types of Stopping Conditions: —Aposteriori Quality (AQ) based Stopping condition. —CRC pass Stopping Condition —CRC result equals previous iteration CRC result. • Programmable maximum/minimum iteration counters • Programmable number of internal operational Turbo engines • Programmable ascending or descending bit or/and byte ordering for Hard Output data.
	DEPE	• DL-SCH and UL-SCH Turbo encoding and rate matching. • Optional Code Block CRC attachment. • Optional Transport Block CRC attachment for TB $\leq$ 6120. • Filler bits insertion. • Transport Block segmentation assist using bit read granularity. • Code Word generation assist in system memory using bit write granularity • Multiple concatenated jobs (up to 32) in single Buffer Descriptor.

Table 26-1. Support for Specific Protocol Standards (Continued)

Protocol Standard	PE	Features
WiMAX OFDMA IEEE 802.16e-2009/	eTVPE	• Turbo decoding processing. • Multiple input data structures support for multiple pre-processing options: —Support for HARQ scheme: – Combining accumulator with new received punctured Code Block. – Outputting of new accumulator buffer for future HARQ transmissions. – 3 programmable weights. —Support of Sub-Block de-interleaving processing. • Multiple options of decoding output results: —Hard output results —16 bits LLR of Aposteriori data for systematic bits only • Max Log Map with Non Linear Dynamic extrinsic Factorization • Hybrid Linear Log Map (MAX*) with programmable Linear Correction factor • Maximum decoded block size of 4800 • Optional CRC check for padded MAC PDU $\leq$ 4784. • Two different types of Stopping Conditions: —Aposteriori Quality (AQ) based Stopping condition. —CRC result equals previous iteration CRC result. • Programmable maximum/minimum iteration counters • Programmable number of internal Turbo engines • Programmable ascending or descending bit or/and byte ordering for Hard Output data.
	DEPE	• Turbo encoding and rate matching. • Optional CRC attachment for padded MAC PDU $\leq$ 4784. • Optional bit randomization for padded MAC PDU $\leq$ 4784. • Multiple concatenated jobs (up to 32) in single Buffer Descriptor.

Table 26-1. Support for Specific Protocol Standards (Continued)

Protocol Standard	PE	Features
UMTS TS 25.212 FDD, TS 25.222 TDD	eTVPE	• Turbo decoding processing. • Multiple input data structures support for multiple pre-processing options: —Rate De-Matching (depuncturing and de-repetition) as per section 4.8 —Support for HARQ scheme: – Combining accumulator with rate de-matched received Code Block. – Outputting of new accumulator buffer for future HARQ transmissions. – 2 programmable weights. • Multiple options of decoding output results: —Hard output results —2 optional types of Soft Output results: – 16 bits LLR of Aposteriori data for systematic bits only – 8 bits LLR of Aposteriori data for systematic, parities and tail bits. • Max Log Map with Non Linear Dynamic extrinsic Factorization • Hybrid Linear Log Map (MAX*) with programmable Linear Correction factor. • Maximum decoded block size of 5114 • Two different types of Stopping Conditions: —Aposteriori Quality (AQ) based Stopping condition. —CRC result equals previous iteration CRC result. • Programmable maximum/minimum iteration counters • Programmable number of internal Turbo engines • Programmable ascending or descending bit or/and byte ordering for output data.
	DEPE	• HS-DSCH and E-DCH (TDD and FDD) Turbo encoding and rate matching. • CRC attachment for Transport Block $\leq$ 5090. • Optional bit scrambling for Transport Block $\leq$ 5090 (FDD only). • Transport Block segmentation assist using bit read granularity. • Code Word generation assist in system memory using bit write granularity • Multiple concatenated jobs (up to 20) in single Buffer Descriptor.

Table 26-1. Support for Specific Protocol Standards (Continued)

Protocol Standard	PE	Features
UMTS TS 25.211, TS 25.213, TS 25.214	CRPE	• Downlink Chip Rate Processing: —Capacity of up to 512 Physical Channels including MIMO, STTD, TSTD and Closed Loop Mode 1 operation per channel. —Input data precision of 16 bit {8I,8Q} complex symbols. —Spreading with SF 4,8,16,32,64,128, and 256 using internally generated channelization codes. —Scrambling using internally generated, up to 32 independent codes, including support for compressed mode codes. —Supporting SF 1 special channels with spreading, scrambling bypass —Programmable complex gains per Physical Channel, with differentiation between data and control information, supporting various slot formats. —Physical channels combining and flexible assignment to up to 16 virtual antenna's. —Optional Beam Forming operation on combined Physical Channels. —Output data precision of 32 bit {16I,16Q} complex chips. • Uplink Batch Processing - for data and control channels with variable spreading factors —Capacity of up to 512 Physical Channels with up to 3328 total fingers from up to 96 antenna streams with up to 512 chips delay spread. —Optional pre-despreading support with up to 128 Physical Channels with SF 4 —Input data precision of 16 bit {8I,8Q} complex chips. —Optional Internal interpolation of x2 oversampled input stream, up to x16 resolution using 8 tap polyphase filter, with programable 8 bit weights per phase. —Despreading with SF 2,4,8,16,32,64,128,256 using internally generated channelization codes. —Descrambling by short or long codes, with up to 512 different, internally generated, scrambling codes. —Optional fingers combining using programable weights. —Optional frequency correction functionality using programable correction factor. —Multiple output formats: – 16 bit fixed point with user defined scaling/alignment or custom 32 bit (16-bit mantissa and 16-bit exponent) floating point formats for fingers combining option. – complex 32 bit {16I,16Q} for fingers combining bypass option.

Table 26-1. Support for Specific Protocol Standards (Continued)

Protocol Standard	PE	Features
UMTS TS 25.211, TS 25.213, TS 25.214	CRPE (cont.)	• Uplink Fast Processing - for (E)DPCCH, HS-DPCCH and RACH message processing —Capacity of up to 512 Physical channels with up to 3200 total fingers from up to 24 antenna streams. —Input data precision of 16 bit {8I,8Q} complex chips. —Optional Internal interpolation of x2 oversampled input stream, up to x16 resolution using 8 tap polyphase filter, with programable 8 bit weights per phase. —Descrambling by short or long codes with internally generated scrambling codes. —Despreading using SF 256. —Optional correlation with pilot sequence. —Programable slot format and early/on-time/late processing for various fields of control channels. —Output 32 bit {16I,16Q}, per finger, per channel, grouped into up to 18 different cyclic buffers. —Internal commands FIFO for flexible updates of fingers and channels association. —Latency up top 96 chips including processing and write back of results to system memory • PN Code Generator (CGPE) —Short or Long codes generation based on programable init values —Generates scrambling code or scrambling code multiplied by spreading code with programable spreading factor and OVSF. —Two output formats: – 16 bit {8I,8Q} format, with throughput of up to 4 Gsamples/sec – 2 bit {1I, 1Q} format, with throughput of up to 32 Gsamples/sec
WiMAX IEEE 802.16m/D3	ETVPE	• Turbo decoding processing. • Single input data structure support of separate vector per each data type (no HARQ combining support). • Multiple options of decoding output results: —Hard output results —16 bits LLR of Aposteriori data for systematic bits only • Max Log Map with Non Linear Dynamic extrinsic Factorization • Hybrid Linear Log Map (MAX*) with programmable Linear Correction factor • Maximum decoded block size of 4800 • Two different types of Stopping Conditions: —Aposteriori Quality (AQ) based Stopping condition. —CRC result equals previous iteration CRC result. • Programmable maximum/minimum iteration counters • Programmable number of internal Turbo engines • Programmable ascending or descending bit or/and byte ordering for Hard Output data.
	DEPE	—Turbo encoding and rate matching (without bit selection). —Code Block CRC attachment. —Bit randomization for padded MAC PDU ≤ 4784 —Multiple concatenated jobs (up to 32) in single Buffer Descriptor.

Table 26-2. Processing Engine (PE) Features

PE	Features
eTVPE	Viterbi Decoding with the following supported features: • Convolutional Decoding using programmable polynomials. • Encoding Rates of 1/2, 1/3 and 1/4 • Periodic depuncturing scheme for supporting various rate cases (for example, 1/3, 1/2, 2/3, 3/4, 5/6) • Decoding Convolutional codes with K= 5,6,7,8,9 • Maximum internal block sizes of: —6144 (K=5, Encoding Rate of 1/2 or 1/3) —4864 (K=5, Encoding Rate of 1/4) —4800(K=6) —2400 (K=7) —1200 (K=8) —600 (K=9) • Optional support for larger block sizes using multiple job descriptors. • Zero Tail decoding support • Tail biting decoding support using WAVA* (Wrap-Around-Viterbi-Algorithm)[1] • Ascending and descending bit and/or byte ordering for output data
eFTPE (three instantiations)	• Variable length FFT/iFFT processing of 128, 256, 512, 1024, 1536 and 2048 points. • Variable length DFT/iDFT processing of the form $2^k \cdot 3^m \cdot 5^n \cdot 12$, up to 1200 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 and 1200 points • Variable input data size, supporting 16 bit {8I,8Q} or 32 bit {16I,16Q} • Programmable guard band insertion and removal for iFFT and FFT processing. • Programmable Cyclic Prefix insertion and removal. • Pre-Multiplication (“vector multiplication”) support by programable complex values vector. • Post-Multiplication (“vector multiplication”) support by programable complex values vector. • Post multiplication by configurable complex constant value. • Frequency correction argument generation for pre or post transform use. • Input data complex conjugate and re-ordering (1...N samples reversed to N...1) • Zero padding of input data • UMTS scrambled pilot samples generation in frequency domain • Programmable scaling method, supporting one of the following methods: —Automatic adaptive scaling using overflow detection between transform stages, and optional programmable overall scaling factor for the output data —User defined scaling between transform stages • Automatic or programmable input scale up for input data with small values, to increase calculation precision. • Multi-transform execution of same size transforms with single job descriptor (“repeat” operation). • Flexible allocation of double buffers to pre-multiplier, post-multiple or output buffer at configuration stage.
DEPE	• All functionality is standard related. See **Table 26-1** for functional details.

Table 26-2. Processing Engine (PE) Features (Continued)

PE	Features
EQPE	• MMSE (Minimum Mean Square Error) / ZF (Zero Forcing)/ IRC (Interference Rejection Combining) Equalization —Estimation of signal by MMSE/ZF/IRC equalizer using following inputs: – observations (antenna data in frequency domain), channel estimation, noise and transmitted layers covariance matrices and previously detected layer (for cancellation) —Inputs precision 32 bit {16I,16Q} with block floating point 8 bit scaling factor. —Supporting various Rx Antenna (1,2,4,8) and MIMO layers (1,2,3,4) configurations —Optional Channel Estimation interpolation with programmable weights. —Embedded support for advanced receivers with iterative interference cancellation schemes with internal single layer cancellation, rank reduction, layer discarding and Hermitian or diagonal MIMO layers covariance matrix. —Optional support for Hermitian noise & interference covariance matrix, with variable granularity, up to different matrix per sub-carrier (RE). —Flexible output format supporting 32 bit {16I,16Q} with block floating point representation using 8 bit scale factor per group of elements or optional full floating point representation using 8 bit scaling factor per element. • Maximum Likelihood estimation implemented with QRD-M tree search —Estimation of signal by QRD-M equalizer using following inputs: – observations, channel estimation, noise variance. —Supporting various Rx Antenna (1,2,4,8[2]) and MIMO layers (1,2,3,4) configurations. —Optional Channel Estimation interpolation with programable real weights. —Configurable search width, with M up to 64[3], providing performance-latency trade-off. —8 bits LLR output generation, with optional floating point (8e8) output support. • Matrix Inversion —Support for up to 4x4 matrix inversion. —Programming model optimized for batch job of multiple matrix inversions. • High precision floating-point arithmetic —Input samples are received in 32 bit {16I,16Q} block floating point (BLF) with 8-bit scale factor. —Internal calculations performed using custom floating-point (FLP) format: 40 bit {20I,20Q} mantissa and 8-bit exponent. —Matrix Inversion performed with QR-Decomposition using Givens-Rotations algorithm. —Output samples are available in three formats: floating-point, block floating-point or fixed-point.
CRCPE	• Eight available polynomials: —$D^{24} + D^{23} + D^6 + D^5 + D + 1$ —$D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1$ —$D^{16} + D^{12} + D^5 + 1$ —$D^{16} + D^{15} + D^2 + 1$ —$D^{32} + D^{26} + D^{23} + D^{22} + D^{16} + D^{12} + D^{11} + D^{10} + D^8 + D^7 + D^5 + D^4 + D^2 + D + 1$ —$D^{18} + D^{17} + D^{14} + D^{13} + D^{11} + D^{10} + D^8 + D^7 + D^6 + D^3 + D^2 + 1$ —$D^{12} + D^{11} + D^{10} + D^8 + D^5 + D^4 + 1$ —$D^6 + D^5 + D^3 + D^2 + D + 1$ • CRC Check or CRC calculation for block sizes of up to 64Kbyte. • Optional input/output byte reverse CRC processing • Optional initial value for continuous processing. • Optional output byte inverse presentation.

1. For details, on the exact implementation see **Section 26.4.3.2.6.7, *Tail Biting Viterbi Processing (WAVA*)***.
2. In case of 8 Rx antenna system, QR decomposition is executed by DSP cores externally to MAPLE-B2 and QR matrices are input to MLD equalization and detection.
3. M=64 supported for up to 2 layers, M=32 for higher number of layers

Table 26-3. Virtual Processing Elements Features

VPE	Features
CONV VPE	• Convolution and correlations processing utilizing eFTPE and EQPE processing elements. • Convolution and correlation are implemented in frequency domain, using internal flow of eFTPE->EQPE->eFTPE, performing FFT, MAC operations and IFFT respectively • Data transfers between eFTPE and EQPE are performed using internal DMA.
TD-TE VPE	• Turbo Decoding and Re-encoding including rate matching —Turbo Decoding including optional HARQ combining and Rate-De-Matching, supporting 3GLTE, UMTS and WiMAX technologies with hard decision output and optional CRC check. —Re-encoding and rate matching of decoding result. • Functionality performed using single job descriptor. • Data transfers between eTVPE and DEPE are performed using internal DMA.
EQ-IDFT-VPE	• 3G LTE Uplink Processing with SC-FDMA processing - chained acceleration • MMSE Equalization process followed by IDFT processing of equalization result • Executed on programable number of SC-FDMA symbols (1 to 12) • IDFT size is fixed for all time domain symbols and equal to MMSE equalization size (12 to 1200 with 3GLTE RB granularity) • Functionality performed using single job descriptor • Data transfers between EQPE and eFTPE are performed using internal DMA.
DFT-EQ-IDFT-VPE	• 3G LTE Uplink Processing with SC-FDMA including cancellation processing - chained acceleration • DFT, followed by MMSE Equalization using input data from DFT processing, followed by IDFT processing of equalization result • Executed on programable number of SC-FDMA symbols (1 to 12) • DFT and IDFT sizes are fixed for all time domain symbols and equal to MMSE equalization size (12 to 1200 with 3GLTE RB granularity) • Functionality performed using single job descriptor • Data transfers between eFTPE, EQPE and back to eFTPE are performed using internal DMA.

Table 26-4. General MAPLE-B2 Features

MAPLE-B2 General Features
• 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 —32KB of BD rings internal memory —TaskID for every BD • Simultaneous multi-technology support, allowing mixture of 3GLTE and UMTS jobs or 3GLTE and WiMAX jobs. • Flexible interrupt scheme —Up to 32 maskable BD ring interrupts —Programmable allocation of BD rings to interrupts —Single ECC maskable interrupt aggregating all MAPLE-B2 ECC interrupts. • Low Power design and features: —Optional clock gating for every Processing Element. —Dynamic supply gating for eTVPE and EQPE elements in case of temporal no operation —Static supply gating for CRPE element when not used, controlled by device level supply connectivity using dedicated pads. • Quad MBus master interfaces for data transfers to/from system memory • Dual MBus slave for accessing MAPLE-B2 for UMTS chip rate data, configuration, control and debug. • One SBus slave for PSIF2 control

26.3 Modes of Operation

The MAPLE-B2 supports two modes of operation defined by the required standard: 3G mode and WiMAX mode. Activating the operation mode is done during the activation sequence of the MAPLE-B2 using the API (see *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)).

26.3.1 3G Technologies Operation Mode

In this operation mode, the MAPLE-B2 supports jobs of 3GLTE and UMTS standards and can dynamically switch the internal parameters and configuration registers between these standards according to the current job description. WiMAX jobs are not supported in this mode.

26.3.2 3GLTE and WiMAX Technologies Operation Mode

In this operation mode, the MAPLE-B2 supports jobs of 3GLTE and WiMAX **IEEE** 802.16e and 802.16m standards and can dynamically switch the internal parameters and configuration registers between these technologies according to the current job description. UMTS jobs are not supported in this mode.

26.4 Functional Description

The MAPLE-B2 functional description is divided into the following areas.

- Initialization. Includes initialization parameters configured by the MAPLE-B2 API. See **Section 26.4.1**.
- Configuration and Control. Includes the general configuration parameters, descriptions and registers used to configure and control MAPLE-B2 operation. Data management is handled by buffer descriptor rings, which are general to all PEs except the CRPE. See **Section 26.4.2.3** for BD ring operation description. Specific BD handling is also described for each PE except the CRPE under the accelerator specific functional descriptions in this subsection. In addition, MAPLE-B2 controls and arbitrates MAPLE-B2 access to the MBus. See **Section 26.4.2.4** for configuration details.
- Internal module operation. Provides detailed information about PE and VPE operation. Detailed information is provided for the MAPLE-B2 PSIF2, each of the PEs, and the VPE.
  - PSIF2 configuration and operation. See **Section 26.4.3.1** for details.
  - eTVPE FEC decoding flow. See **Section 26.4.3.2** for details.
  - eFTPEs FFT/iFFT/DFT/iDFT operation. See **Section 26.4.3.3** for details.
  - DEPE FEC encoding operation. See **Section 26.4.3.4** for details.
  - EQPE equalization and matrix inversion operation. See **Section 26.4.3.5** for details.
  - CRPE uplink/downlink UMTS Chip Rate processing operation. The functional description is divided by functionality into the following three subsections:

    - CRPE Uplink Batch Mode (CRPE-ULB). See **Section 26.4.3.6.1** for details.
    - CRPE Uplink Fast Processing (CRPE-ULF). See **Section 26.4.3.6.2** for details.
    - CRPE Downlink (CRPE-DL). See **Section 26.4.3.6.3** for details.
  - CRCPE CRC operation. See **Section 26.4.3.7** for details.
  - CGPE operation. See **Section 26.4.3.8** for details.
  - CONVPE convolution/correlation processing operation. See **Section 26.4.3.9** for details.
- External master support. MAPLE-B2 can work with external master through the Serial RapidIO Doorbells. See **Section 26.4.4** for details.
- MAPLE-B2 Internal Task Control. Permits configuration of BD search periods and forcing routine execution. See **Section 26.4.5** for details.
- MAPLE-B2 Power Saving Schemes. There are two types of power saving schemes. One is control of the clock source for not utilized PEs or RISCs. Second scheme is dynamic power gating, which shut's down the supply's of EQPE or eTVPE when not active, and turn's the power on when a new job arrives. All of these features are programmable. See **Section 26.4.6** for details.
- Reset. The MAPLE-B2 has two types of reset. See **Section 26.4.7** for details.

26.4.1 Initialization

Upon power-on reset deassertion, the MAPLE-B2 is in freeze state, that is, its internal RISC engines are denied access to the internal instruction memory. At this stage, the MAPLE-B2 API must be initiated to activate the MAPLE-B2. The MAPLE-B2 API detailed description is in *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)

26.4.1.1 MAPLE-B2 API

The MAPLE-B2 API is 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/reset values (as described in **Section 26.5.3.1**, *General Parameter RAM Description*)
- Initialize configuration parameters for the relevant PEs according to the API input parameters (see **Section 26.4.1.2**).
- Upload the program code into the MAPLE-B2 instruction RAM.
- Activate the RISC engines by allowing them to access to the program memory.

The MAPLE-B2 API must be used after every external reset. It can also be used to generate a soft reset internally to the MAPLE-B2. For details, see *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG).

26.4.1.2 Initialization Parameters

During the MAPLE-B2 initialization (see *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)), the following MAPLE-B2 parameters in the API must be initialized:

- General MAPLE initialization parameters:
  - MAPLE Mode Configuration x Parameters (MMCxP)—General MAPLE-B2 Initialization parameters (for details, see **Section 26.5.2.1**, *MAPLE Mode Configuration 0 Parameter (MMC0P)*, on page 26-320 and **Section 26.5.2.2**, *MAPLE Mode Configuration 1 Parameter (MMC1P)*, on page 26-322).
- PE-specific initialization parameters:
  - eTVPE Mode Configuration Parameter (TVMCP)—General eTVPE related initialization parameter (for details, see **Section 26.5.2.3**, *MAPLE eTVPE Configuration Parameter (MTVCP)*, on page 26-325).
  - CRPE Down Link Output Mode Configuration Parameter (CDOMCP)—CRPE-DL Output mode related initialization parameter (for details, see **Section 26.5.2.4**, *CRPE-DL Output Mode Configuration Parameter (CDOMCP)*, on page 26-326)
  - CRPE Uplink Batch Mode Configuration Parameter (CRUBMCP)—CRPE-ULB General configuration Parameter (for details, see **Section 26.5.2.5**, *CRPE-ULB Mode Configuration Parameter (CRUBMCP)*, on page 26-328).

26.4.2 MAPLE-B2 Configuration and Control

After the MAPLE-B2 API has initialized the required initialization parameters and before actual processing begins, the host must configure the required PEs and the supporting RAM-based structures and registers to perform the required operations to support the specified protocol. See **Table 26-1** to identify the PEs used by the specific supported protocols. In addition, the host must also specify the arbitration methodology for BD/BD rings and MBus access.

26.4.2.1 PE Configuration

Configuration of each PE is done using specific RAM-based parameter fields, descriptor fields, and registers. Data management is done using a set of buffer descriptors usually organized as BD rings. The CONVPE also requires configuration of task descriptors because it combines the functionality of multiple PEs to create a virtual processing element (VPE). Configuration may also require configuration of special structures for the individual PE processing. In addition, each

PE requires special input and output data structures to support the PE operation. Each PE functional description includes processing flow and detailed information about required parameters and data structures. Detailed layout information for parameter fields, descriptor fields and registers is provided by PE in **Section 26.5**, *Programming Model*, on page 26-315.

26.4.2.2 PE Arbitration Between Different Clients

The MAPLE-B2 instantiate 3 eFTPE engines, each can be initiated due to one of the following events:

1. eFTPE Buffer Descriptor (BD) arrival. See **Section 26.4.2.3**, *Buffer Descriptor (BD) Ring Handling* and **Section 26.5.4.3**, *eFTPE Buffer Descriptor Structure* for eFTPE BD description and general BD handling.
2. EQPE BD arrival with *F_DFT* or *POST_IDFT* bits enabled. See **Section 26.5.4.5**, *EQPE BD Structure* for details.
3. CONVPE BD arrival. See **Section 26.5.4.9**, *CONVPE Buffer Descriptor Structure* for details.

The following describe how the MAPLE-B2 handles the arbitration between the 3 eFTPE units and the internal arbitration of each eFTPE unit with respect to the above clients.

26.4.2.2.1 Arbitration Between the Three eFTPE Engines

Each eFTPE engine is related to a dedicated set of eFTPE BD Rings hence there is no arbitration between the eFTPE engines with respect to eFTPE BDs. On internal arbitration between the eFTPE BDs related to the same eFTPE engine, see **Section 26.4.2.3.2**, *BD Arbitration*, on page 26-18.

If EQPE BDs with *F_DFT* or *POST_IDFT* bits enable are detected, the MAPLE-B2 maintain cyclic arbitration between the eFTPE engines with respect to EQPE BD assist: each DFT/iDFT job required by the EQPE BDs is assigned to the eFTPE engines in a cyclic manner, that is, the first eFTPE job (related to that BD) is assigned to eFTPE0, the second eFTPE job (of that EQPE BD or the next EQPE BD) is assigned to eFTPE1 and so forth.

The CONVPE BDs are executed using the FDU logic in the EQPE and the 3 eFTPE engines (see **Section 26.4.3.9**, *CONVPE Convolution and Correlation Processing Operation* for details). Assigning the eFTPE jobs related to the CONVPE BD to the eFTPE engines is done in a cyclic manner, that is, the first eFTPE job (related to the CONVPE BD) is assigned to eFTPE0, the second eFTPE job (of that CONVPE BD) is assigned to eFTPE1 and so forth. The last eFTPE engine index used by the CONVPE BD is kept, so that the next CONVPE BD starts with the following index (cyclic progress).

26.4.2.2.2 Internal eFTPE Arbitration

As described in **Section 26.4.2.2**, *PE Arbitration Between Different Clients*, each eFTPE engine has 3 potential clients which can assign it with eFTPE jobs. The following details the arbitration scheme by which the internal firmware assign the jobs from the different clients to the eFTPE:

For each potential client the internal firmware maintain a queue of jobs sourced from the client BDs. The eFTPE is assigned with jobs from the 3 queues in a cyclic manner thus preventing any potential starvation state for any of the queues. The following figure describe an example of the processing order of the eFTPE jobs given a certain queues state:

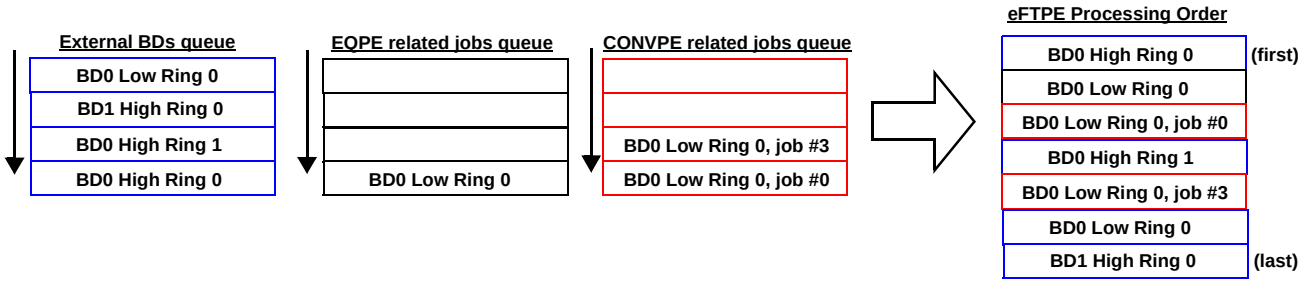


Figure 26-1. eFTPE processing order with respect to multiple internal queues

26.4.2.3 Buffer Descriptor (BD) Ring Handling

For all MAPLE-B2 internal Processing Elements (PEs) except the CRPE, the logical handshake between the DSP device cores and peripherals and the MAPLE-B2 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. To enable higher number of hosts by special configuration, up to 16 hosts can be supported for each PE, with each host receiving only one buffer descriptors ring with the same priority assigned to all the rings. The MAPLE BD Rings Configuration Parameters 0 and 1 (MBDRCP[0–1]) control the arbitration order between the high and low rings for each PE. For details, see **Section 26.4.2.3.2**, *BD Arbitration*.

Note: The CRPE Programming model and handshake with host is not buffer-descriptor based and is described in detail in **Section 26.4.3.6.2**, *Chip Rate Uplink Fast Operation*, **Section 26.4.3.6.3**, *Chip Rate Downlink Processing Operation* and **Section 26.4.3.6.1**, *Chip Rate Uplink Batch Processing Element*.

26.4.2.3.1 BD Rings Arbitration

The MAPLE-B2 manages its buffer-descriptor rings using a set 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 eTVPE; F0 for FTPE_0; F1 for FTPE_1; F2 for FTP#_2; DE for DEPE; EQ for EQPE; CRC for CRCPE, CG for CGPE and CONV for CONVPE.
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 last BD job being processed. This pointer is increment by MAPLE-B2 once it starts with the BD processing. 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-B2 to locate the next BD in the ring. If the OWNER bit of the BD pointed by this field is cleared, the MAPLE-B2 assumes no jobs are available in the ring.
- INT_TRGT—Interrupt target. Describe which of the MAPLE-B2 BD rings interrupts (bdr_done0 to bdr_done31) is to be asserted on completion of a BD from this ring. If CRPE ULB is enabled to use interrupt, it consumes one of the bdr_doneX interrupts.
- VLD—Valid bit. When set, the MAPLE-B2 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-B2 must communicate with it using the serial RapidIO doorbells. See **Section 26.4.4**, *External Masters Support Using Serial RapidIO Doorbell* for details.

- [HOST_ID]—Host ID in case of external master. This field is used by the MAPLE-B2 for the WORD3[EDID:DID] field in the Type10 Outbound Doorbell descriptor.
- [P_TR_IF]—Port Target Interface. Up to 16 possible RapidIO target interfaces. This field is used by the MAPLE-B2 for the WORD4[TINT] field in the Type10 Outbound Doorbell descriptor.
- [EXT_MST]—Specifies whether the master is connected to MAPLE-B2 via regular interrupt or Serial RapidIO Doorbell interrupt.

26.4.2.3.2 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]. 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 26.5.4.1.1**, *MAPLE BD Rings Configuration Parameter 0 (MBDRCP0)*. For example, if the chosen arbitration is H-H-L, the MAPLE-B2 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 in this manner.

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-B2 executes BDs according to the following order: when the low priority rings are accessed, the MAPLE-B2 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-B2 continues to the next ring according to its cyclic order.

The MAPLE-B2 uses the MBDRCP[pe_R_LOOP] to determine the potential valid rings to be scanned in each ring queue. For example, if MBDRCP[TV_R_LOOP] equals 4, the MAPLE-B2 searches 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-B2 performance.

Within each BD ring, MAPLE-B2 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-B2 starts executing this BD (assuming the relevant PE is available). If the [OWNER] bit is not set, the MAPLE-B2 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-B2 how to update the M<pe>BR(H/L)PAxP[BDR_RD_PTR] when searching for next BD to process. 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-B2 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-B2.

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-B2 starts executing this BD and on BD completion, it clears the [OWNER] bit. If the [OWNER] bit is not set MAPLE-B2 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-B2 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-B2 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 26.5.5.1.1**, *PSIF Command Register (PCR),* on page 26-501 and **Section 26.4.5**, *MAPLE-B2 Internal Task Control,* on page 26-312.
- 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-B2 proceeds with the BD execution. For details on job execution, refer to the specific accelerator used.

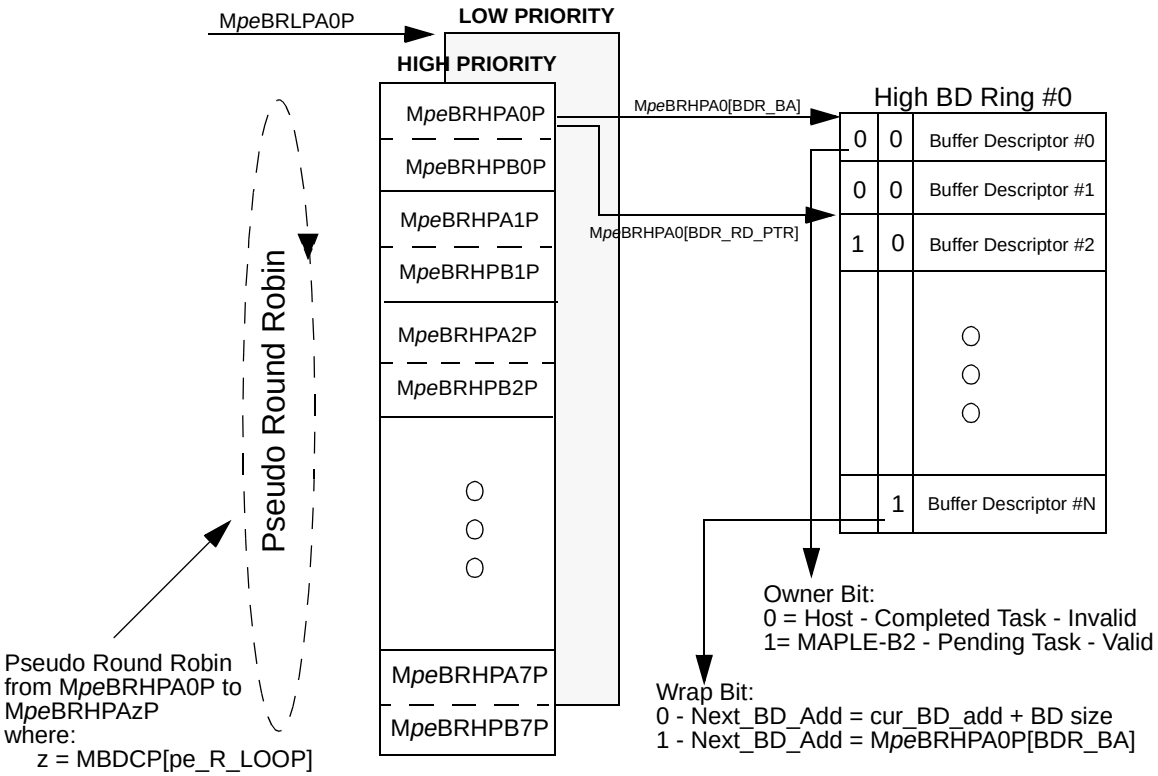
- **Section 26.4.3.2**, *eTVPE HARQ, Rate De-Matching, and Turbo/Viterbi Decoding Flow*, on page 26-28.
- **Section 26.4.3.3**, *eFTPE FFT/iFFT/DFT/iDFT Operation*, on page 26-78.
- **Section 26.4.3.4**, *DEPE Downlink Turbo Encoding Operation*, on page 26-117
- **Section 26.4.3.5**, *EQPE Equalization Processing Operation*, on page 26-144
- **Section 26.4.3.6.1**, *Chip Rate Uplink Batch Processing Element,* on page 26-198
- **Section 26.4.3.6.2**, *Chip Rate Uplink Fast Operation*, on page 26-234
- **Section 26.4.3.6.3**, *Chip Rate Downlink Processing Operation*, on page 26-266
- **Section 26.4.3.7**, *CRC Operation,* on page 26-285
- **Section 26.4.3.8**, *Code Generation Operation,* on page 26-290
- **Section 26.4.3.9**, *CONVPE Convolution and Correlation Processing Operation,* on page 26-298

Note: The CRPE programming model is not based on Buffer Descriptor and is described in the CRPE related Sections (**Section 26.4.3.6.2, *Chip Rate Uplink Fast Operation***, **Section 26.4.3.6.3, *Chip Rate Downlink Processing Operation*** and **Section 26.4.3.6.1, *Chip Rate Uplink Batch Processing Element***).

When a job is completed, the MAPLE-B2 clears the [OWNER] field, and, if specified in the accelerator-specific BD, initiates an interrupt to the host. The MAPLE-B2 uses the M<pe>BDR(H/L)PBxP[INT_TRGT] field to define the target of this interrupt, with up to 32 different target interrupts possible.

If the M<pe>BRHPBxP[EXT_MST] is set to DSP device external master, the MAPLE-B2 initiates a door-bell interrupt via the serial RapidIO port (see **Section 26.4.4**, *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 PSIF2 DRAM of the MAPLE-B2.



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-B2 Buffer Descriptor Data Structure and Mechanism of One PE

26.4.2.3.3 BDs Data Coherency

When each of the internal PEs completes its current job, it is up to the MAPLE-B2 to output the resulting data to the system memory and to negate the relevant [OWNER] bit. If interrupt is

required for this job ([INT_EN] = 1) it is assured by MAPLE-B2 that when the interrupt is asserted and the [OWNER] bit is negated, the internal DMA has completed all relevant data transfer and all result data is already placed in system memory. If no interrupt is required for this job ([INT_EN] = 0), the [OWNER] bit negation occur when all internal DMA configuration is completed but before the DMA has completed its relevant data transfers, hence the [OWNER] bit negation may occur before all the output data is located in system memory. That may cause coherency problems for a host polling the [OWNER] bit to continue with the result data processing, but enables faster response of MAPLE to next tasks. This would typically be very useful in cases of interrupts coalescing for multiple jobs.

Resolving a coherency problems for a host polling the [OWNER] bit instead of enabling the BD interrupt can be done as follows:

1. Assign the relevant BD Ring with an un-used BD Done interrupt (for example, set the M<pe>BRHPBxP[INT_TRGT] to 0x1F, that is, interrupt line #31).
2. Mask interrupt line 31 in the MAPLE PIC (reset BDR_DONE_M[31] in the PSPICMR0 register).
3. Set the interrupt enable ([INT_EN] = 1) for the BD which its [OWNER] bit is being polled.

These actions results in the MAPLE-B2 working assuming interrupt is required, that is, it assure all data is placed in system memory before negating the [OWNER] bit, but with no actual interrupt is asserted towards the host.

26.4.2.4 MBus Priority Scheme Configuration

The MBus masters interfaces support 4 priority levels 0 to 3, where the lowest priority is 0 and the highest is 3. The MAPLE-B2 allows a flexible priority scheme for its MBus master interfaces. There are a few optional priority schemes configurations that should be set during MAPLE-B2 initialization using the API parameters (see **Section 26.5.2.1**, *MAPLE Mode Configuration 0 Parameter (MMC0P)*). The following subsections describe each of these optional priory schemes.

26.4.2.4.1 MBus Fixed Priority Scheme

The MAPLE-B2 initiates all its MBus master accesses with fixed priority, set by the host during its initialization processing. Configure the MAPLE-B2 to work with MBus fixed priority using the following procedure:

1. Configure the [MB_PR_SCH] field of the MMC0P parameter during API initialization as 00.
2. Configure the [MBx_DF_PR] fields (x=0...3) of the MMC0P parameter during API initialization to the required fixed priority level for MBus<x> interface as described in **Section 26.5.2.1**, *MAPLE Mode Configuration 0 Parameter (MMC0P)*.

26.4.2.4.2 MBus Dynamic Priority Accesses Scheme

The MAPLE-B2 determines the priority of each job according to the [MB_PR] field of each BD. The priority configured in the [MB_PR] is valid for both the input data stage and the output data stage related to that BD. For the CRPE modules, where the programming model is not BD based, the [MB_PR] field is defined in their configuration parameters. Use the following procedure to configure the MAPLE-B2 to work with MBus Dynamic Priority Accesses Scheme:

1. Configure the [MB_PR_SCH] field of the MMC0P parameter during API initialization as 01.
2. Configure the [MB_PR] field of each Buffer Descriptor (BD) or CRPE configuration parameters to set the required priority for that BD or CRPE module.

When working with this priority scheme, it is guaranteed that each access gets its priority from its related [MB_PR] field only.

26.4.2.4.3 MBus Dynamic Priority Accesses with DMA Queue Upgrade

Each of the four MAPLE-B2 DMA engines include a DMA command queue of up to 48 entries, each with up to 384 bytes transfer, allowing the RISC engine to launch its DMA commands and continue to its following task while the DMA engines continue with data transfer. In this priority mode the MAPLE-B2 determines the initial priority of the DMA commands of each job according to the [MB_PR] field of its BD. If a certain BD includes MBus high priority requirements, once its DMA commands are placed in the DMA command queue, it automatically upgrades the priority of all preceding DMA commands currently exist in the DMA queue. This priority mode assure that a high priory DMA command is not delayed in the DMA command queue due to low priority commands previously launched to that same DMA engine.

The following example demonstrates the Mbus priority upgrade mechanism:

1. 8 commands (256 bytes size each) are pushed to DMA-0 with Mbus priority 0 due to eFTPE job completion.
2. 3 commands (256 bytes size each) are pushed to DMA-0 with Mbus priority 2 due to eTVPE job completion.
3. As all DMA commands are executed by order, once the first high Mbus priority command (of the eTVPE) is pushed into the DMA queue, all Mbus priority of the existing commands (of the eFTPE) are upgraded to Mbus priority 2, thus allowing higher priority in the system and fast execution of the eTVPE commands as well.

Use the following procedure to configure the MAPLE-B2 to work with MBus Dynamic Priority Accesses with DMA Queue Upgrade Scheme is done by:

1. Configure the [MB_PR_SCH] field of the MMC0P parameter during API initialization as 10.
2. Configure the [MB_PR] field of each Buffer Descriptor (BD) or CRPE configuration parameters to set the required priority for that BD or CRPE module.

26.4.3 Internal Modules

This section describes the MAPLE-B2 internal module functional information, including general operation and flow, specific protocol support, and input and output data and control structures (organized by specific modes) for each of the following modules:

- PSIF2. Includes memory error detection/correction and interrupt configuration and handling. See **Section 26.4.3.1**.
- eTVPE. See **Section 26.4.3.2**
- eFTPE. See **Section 26.4.3.3**
- DEPE. See **Section 26.4.3.4**
- EQPE. See **Section 26.4.3.5**
- CRPE. See **Section 26.4.3.6**
- CRCPE. See **Section 26.4.3.7**
- CGPE. See **Section 26.4.3.8**
- CONVPE. See **Section 26.4.3.9**.

26.4.3.1 MAPLE-B2 Second Generation Programmable System Interface (PSIF2)

The PSIF2 integrates four 128-bit wide MBus master ports used by the MAPLE-B2 to transfer input and output data to and from system memory. It also integrates two 128-bit MBus Slave ports for the following purposes:

- General purpose connection to the system interconnect CLASS fabric. This port is a read/write port and is referred to as MBus Slave0 port. This connection allows any host or peripheral to access the MAPLE-B2 internal memories for the following:
  - Place job-descriptors in the PSIF2 internal memories
  - Read or write data from or to the PE internal memories[1].
- Direct connection to the device Antenna interface (CPRI) for direct data transfer between Antenna interface and MAPLE-B2. This port is a write only port and is referred to as MBus Slave1 port.

1. Using this port antenna interface accesses MAPLE-B2 for antenna data read in Tx (Downlink) direction

The PSIF2 uses four internal RISC engines to manage the data flow and a programmable interrupt controller (PIC) to track events/errors that occur during operation. The PSIF2 internal registers are accessed using the SBus.

Figure 26-3 describes the MAPLE-B2 high level block diagram.

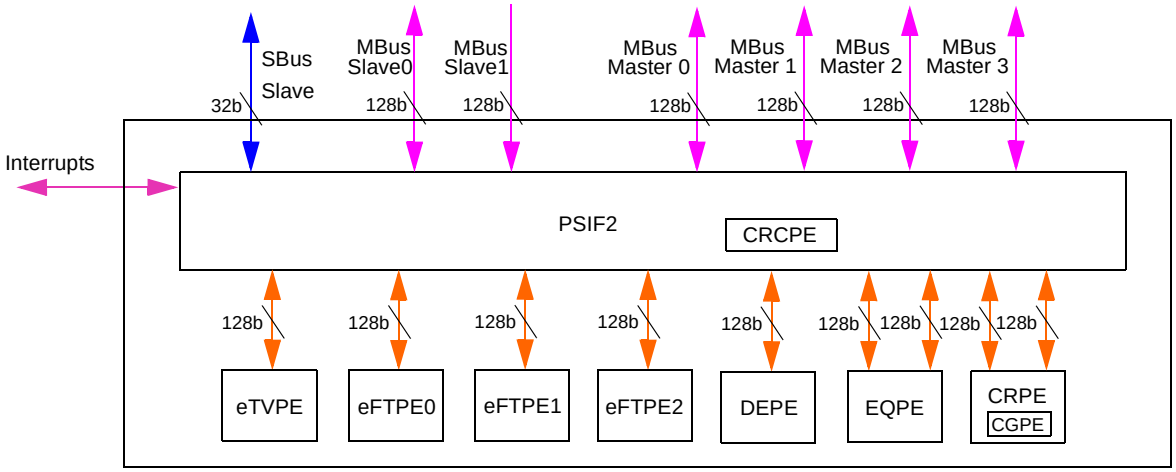


Figure 26-3. MAPLE-B2 High Level Block Diagram

The PSIF2 provides memory error correction/detection support. It uses a system of interrupts to indicate when data transfers are complete and whether system errors occur. The following subsections describes these functions in detail.

26.4.3.1.1 Memory Error Correction/Detection Support

Most of the MAPLE-B2 internal memories support either ECC (Error Correction Code) logic or Parity logic. ECC logic is capable of fixing single bit error or detecting multiple bit error in a single memory line while Parity logic is capable of detecting single bit error in a single memory line. Following is the list of groups of memories with ECC support:

- PSIF2:
  - DRAM memories- Data RAM memory which includes the MAPLE-B2 parameters, Buffer Descriptors and other internal configurations
  - IRAM memories- Instruction RAM which includes code used by the internal RISC engines.
- eTVPE:
  - DRE - Memories related to each of the Turbo engines (DREs)
  - EXT memories - Extrinsic memories which used for intermediate results and soft outputs.
  - HARQ memories - Used for the HARQ combining.
- eFTPE_<x>:
  - IBA memories - Input Buffer A memories. Holds the input data and the data between the radix stages.

- IBB memories - Input Buffer B memories. Holds the input data and the data between the radix stages.
- IBC memories - Input Buffer C memories. Holds the input data and the data between the radix stages.
- BP memories - Buffers Pool memories. Used as Output buffer, Pre-Multiplier buffer (if enabled) and Post Multiplier buffer (if enabled).

- EQPE:
  - Output buffers - Holds the EQPE results.
  - Output Scale buffer - Holds the EQPE output scale results.
  - Input Scale buffer - Holds the input data scales.
  - Interpolation Weights buffers - Holds the Interpolation weights.
  - Y buffers - Hold the input data from the Antenna I/F.
  - F buffers - Hold the Feedback input data.
  - S buffers - Hold the inverted noise covariance input matrices
  - H buffers - Hold the Channel estimation input matrices
- CRPE-DL:
  - Output buffers - Hold the resulting data of the CRPE-DL.
  - Virtual Antenna buffers - Internal memories for holding temporary antenna data.
  - TPC buffers - Internal memories for holding TPC data.
  - Input buffers - Hold the input data for the CRPE-DL.
  - General Control - Internal memories for holding the general control commands.
  - Input Buffer Write Pointers - Internal memories for holding the input buffers write pointers.
  - Slot Format LUT - Internal memories for holding the Look-up Table required for the Slot Format definition.
  - Slot Command List buffers - Internal buffers for holding the slot commands.
- CRPE-ULF:
  - Input Buffer - Holds input data for processing.
  - Command FIFO - Internal memory which holds the commands.
  - PCH Table - Internal memory which holds the Physical Channel parameters.
  - Finger Command Table - Internal memory which holds the fingers commands.
  - Finger List- Internal memory which holds the fingers list
  - Finger Table - Internal memory which holds the fingers tables
  - Packet Generation Table - Internal memory which holds tables for packet generation.
  - Packet Data FIFO - Internal memory which holds the data FIFO packets
  - Packet header FIFO - Internal memory which holds packet FIFO header
  - Magenta Initiator Buffer - Internal memory which holds the data buffer of the magenta initiator.
- CRPE-ULB:
  - Input Buffers. Holds input data for processing.

— Output Buffers. Hold the resulting data of the CRPE-ULB.
— FILE memory. Linked lists entries memory.
— FCQM memory. Fingers Command queue internal memory.
— PCQM memory. PCH command queue internal memory.
— PCHM memory. PCH parameters internal memory.
— OBPM memory. Output parameters internal memory.

Following is the list of groups of memories with Parity support:

- eTVPE. Channel Data (CDL) memories used for Turbo pre-processing.

Enabling/Disabling the ECC/Parity error support is done only during the MAPLE-B2 initialization (see **Section 26.5.2.1**, *MAPLE Mode Configuration 0 Parameter (MMC0P)*), by setting the [ECC_EN] bit in the MMC0P parameter. If ECC/Parity error is enabled, all relevant memories supporting ECC/Parity are initialized by the MAPLE-B2 during MAPLE-B2 API execution, to prevent false ECC/parity errors. If an PSIF2 ECC interrupt indication has occur, it is recommended to reset MAPLE-B2, because the error may be program code error (IRAM ECC) or parameters data error (DRAM). Any other ECC error is a data ECC error and should be handled by the host. For additional details in ECC interrupt see **Section 26.4.3.1.2.3**, *General ECC Error Event Interrupt*.

26.4.3.1.2 MAPLE-B2 Interrupts

The MAPLE-B2 supports up to 32 maskable BD done interrupts and single maskable general ECC interrupt. Detailed information on each type of interrupt is in the following subsections.

26.4.3.1.2.1 BD Rings Done Indication Interrupts

The MAPLE-B2 supports up to 32 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 32 interrupts using the [INT_TRGT] fields of the MAPLE-B2 BD ring configuration parameters. Mapping a given ring to a given interrupt means an interrupt assertion on completion of any BD tasks related to that ring, assuming:

- The [INT_EN] bit in the BD is set.
- The mask bit related to that interrupt in PSIF PIC Mask Register 0 (PSPICMR0) is set.

Each of these 32 interrupt lines can be configured to be either an edge interrupt or a level interrupt using PSIF PIC Edge/Level Register 0 (PSPICELR0). If edge configuration is chosen, the PSIF PIC Interrupts Assertion Clocks Register (PSPICIACR) determines the length of the edge interrupt (2, 4, 6, 8, 10, 12, or 16 MAPLE-B2 clock cycles). If level configuration is chosen, the PSIF_PIC holds an event bit for each of these interrupts in PSIF PIC Event Register 0 (PSPICER0), which is set once an interrupt event has occurred. This bit is cleared by the host when accessing the relevant bit by writing a value of 1 to the bit.

Each of these interrupts can be masked using a mask bit in PSIF PIC Mask Register 0 (PSPICMR0). Masking an interrupt means that only the interrupt line is not asserted; in case of level configuration the event bit related to that interrupt is still set.

Note: If CRPE-ULB is enabled, the BD ring interrupt number 32 is allocated for the CRPE-ULB functionality and must not be allocated to any of the PE's BD rings.

26.4.3.1.2.2 General Error Event Interrupt

The MAPLE-B2 supports a single interrupt line (ipi_general_error) indicating that one or more of the events specified by the PSIF PIC Event Register 1 (PSPICER1) has occurred: For each of these events, a dedicated event bit is located in the PSPICER1. Clearing each of these bits is done by writing 1 to the asserted event bit. Each of the event bits related to this interrupt has a dedicated mask bit in the PSIF PIC Mask Register 1 (PSPICMR1), which 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 PSPICER1 to identify the source of the interrupt.

26.4.3.1.2.3 General ECC Error Event Interrupt

The MAPLE-B2 supports a single general ECC/Parity interrupt, which can be triggered due to a multiple ECC/parity error event in any of the ECC/Parity supported memory groups described in **Section 26.4.3.1.1, *Memory Error Correction/Detection Support***.

26.4.3.1.2.3.1 Detecting the Error Source

Once the ECC/Parity general error is asserted, detecting the source of the ECC/Parity error should be done using the following steps:

1. Read the PSIF PIC Event Register 2 (PSPICER2) status register. It includes specific ECC error indications for the PSIF2 and eTVPE and general ECC error indication on the other PEs.
2. If eFTPE_x ECC indication is asserted (PSPICER2[8:6] bits), read the EFTPE_<x> ECC Interrupt Status Register (FTPE<x>ECCISR) to identify the exact memory group which generated the ECC error.
3. If EQPE ECC indication is asserted (PSPICER2[9] bit), read the EQPE ECC Event Register (EQ_ECCEVENT) to identify the exact memory group which generated the ECC error.
4. If CRPE-DL ECC indication is asserted (PSPICER2[10] bit), read the CRPE-DL ECC Status Register (CDECCSR) to identify the exact memory group which generated the ECC error.
5. If CRPE-ULB ECC indication is asserted (PSPICER2[11] bit), read the CRPE-ULB Event Status Register (CRUBESR) to identify the exact memory group which generated the ECC error.

6. If CRPE-ULF ECC indication is asserted (PSPICER2[12] bit), read the ULF ECC Status Register (ULFECCSR) to identify the exact memory group which generated the ECC error.

Reset an ECC/Parity interrupt by writing ‘1’ to the relevant bit in the relevant status register of the PE (for the cases of eFTPE_x, EQPE, CRPE-DL, CRPE-ULB and CRPE-ULF), followed by resetting the relevant bit in the PSPICER2 register.

Each of the event bits in PSPICER2 related to the interrupt has a dedicated mask bit in the PSIF PIC Mask Register 2 (PSPICMR2) that allows the host to control the events generating the general ECC error interrupt.

The general ECC error interrupt is level interrupt only, allowing the host to access the PSIF PIC Event Register 2 (PSPICER2) to identify the source of the interrupt.

26.4.3.1.2.3.2 Handling a General ECC Error Event

Due to the pipelined native operation flow of the internal PEs, it is impossible to detect the exact job on which the ECC event occurred, therefore the following suggests an optional flow for the host to deal with ECC error interrupt assertion in one of the internal PEs:

1. Reset event indication as specified above.
2. Invalidate all BD Rings related to the PE which the ECC event was sourced.
3. Wait for ~60usec to assure that the current BD already processed by MAPLE-B2 is completed.
4. Re-write all recently completed BDs and all un-completed BDs to the base address of the relevant BD Rings again.
5. Reset the relevant BDR_RD_PTR fields of the relevant BD Rings.
6. Validate the BD Rings.

26.4.3.2 eTVPE HARQ, Rate De-Matching, and Turbo/Viterbi Decoding Flow

The MAPLE-B2 supports Rate De-Matching and Turbo/Viterbi decoding for variable standards and configurations as described in **Section 26.2**, *MAPLE-B2 Features*. These operation are executed using the Enhanced Turbo Viterbi Processing Element (eTVPE).

26.4.3.2.1 eTVPE Pipelining

The eTVPE is a pipelined machine with the pipeline stages for 3GLTE and WiMAX processing shown in **Figure 26-4**:.

Figure 26-4. eTVPE Pipeline Stages for 3GLTE and WiMAX Processing

Figure 26-5 shows the pipeline stages for UMTS processing.

Figure 26-5. eTVPE Pipeline Stages for UMTS Processing

For 3GLTE and WiMAX the following is executed in each pipeline stage:

- HARQ. Rate De-Matching (depuncturing only) and HARQ combining.
- Channel Data. Sub-block de-interleaving and CTC interleaving of systematic data.
- Turbo/Viterbi. Turbo or Viterbi decoding, and optional CRC.

For UMTS the following is executed in each pipeline stage:

- Channel Data. Rate De-Matching.
- HARQ. HARQ combining and CTC interleaving of systematic data.
- Turbo/Viterbi. Turbo or Viterbi decoding.

The eTVPE is pipelined machine such as block N can be processed by the Turbo/Viterbi stage while block N+1 can be processed by the Channel-Data/HARQ stage and block N+2 can be processed by the HARQ/Channel Data stage.

26.4.3.2.2 Turbo Standard Parameter Assumptions

The MAPLE-B2 supports HARQ, Rate De-Matching and Turbo decoding according to the following technologies: WiMAX, 3GLTE and UMTS. For details on MAPLE-B2 operation

modes, refer to **Section 26.3**, *Modes of Operation*. **Table 26-5** lists some of the parameters assumed by MAPLE-B2 for each of supported standards:

Table 26-5. MAPLE-B2 Assumptions for Each Supported Standard

Technology	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)
UMTS	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)

26.4.3.2.3 eTVPE Initialization Parameter

During the MAPLE-B2 initialization (see *MAPLE-B Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)), certain eTVPE related parameter in the API must be initialized: See **Section 26.5.2.3**, *MAPLE eTVPE Configuration Parameter (MTVCP)* for details.

26.4.3.2.4 eTVPE Buffer Descriptor

eTVPE buffer descriptors are described in detail in **Section 26.5.4.2**, *eTVPE Buffer-Descriptor Structure*, on page 26-412.

26.4.3.2.5 eTVPE Input Data Structures

This section describes all possible input data structures for eTVPE supported by the MAPLE-B2. **Table 26-6** explains the meaning of the acronyms used in this section.

Table 26-6. 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)
PF	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)
PS	Parity byte for interleaved half iteration
PSy	First Parity byte for interleaved half iteration (WiMAX Dou-Binary)

Table 26-6. Symbol Identification Acronyms (Continued)

Acronym	Explanation
PSw	Second Parity byte for interleaved half iteration (WiMAX Dou-Binary)
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 26-7 describes the expected data types for each of the possible supported standards, with respect to the algorithm (Turbo/Viterbi) and the encoding rate.

Table 26-7. Turbo/Viterbi Input Data Type

No.	Technology	Descriptor Fields		Input Data	
		ALG[1]	ENC_RATE[2]	Systematic	Parity
1	3GLTE, UMTS	0 = Turbo	1 = 1/3	SD	PF, PS
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 eTVPE buffer descriptor - see **Section 26.5.4.2**, *eTVPE Buffer-Descriptor Structure*."
2. ENC_RATE is the Encoding Rate field in the eTVPE buffer descriptor - see **Section 26.5.4.2**, *eTVPE Buffer-Descriptor Structure*."
3. Viterbi configuration is not dependent on technology.
4. For Viterbi - there is no systematic data, only parity bits.

The input data for each configuration as described in **Table 26-7**, is composed of input samples stored as 8-bit 2s complement numbers, and is arranged in system memory in one of the supported input data structures as described in **Section 26.4.3.2.5**, *eTVPE Input Data Structures*

26.4.3.2.5.1 Input Samples Polarity

As mentioned above, the input data samples to the eTVPE are 8-bit 2's complement soft numbers ranging from –128 to +127. The configuration of which values (positive values or negative values) are to represent the logic 0 and the logic 1 of the input data is controlled by the [POL] field of the MTVCP parameter of the API (see **Section 26.4.3.2.3**, *eTVPE Initialization Parameter*). The two possible representations are:

- [POL] = 0:
  logic 0 is mapped to –1 (–128 in 2's complement representation)
  logic 1 is mapped to +1 (+127 in 2's complement representation)
- [POL] = 1:
  logic 0 is mapped to +1 (+127 in 2's complement representation)
  logic 1 is mapped to –1 (–128 in 2's complement representation)

26.4.3.2.5.2 LTE HARQ Input Data Structure

The LTE HARQ input data structure provides support for HARQ combining with rate de-matching and Sub-Block De-interleaving processing as appears in the 3GLTE (Evolved UTRA) turbo decoding as specified in 3GPP TS 36.212, section 5.1.2.2, and is valid only for 3GLTE technology.

Configuring the MAPLE-B2 to work with LTE HARQ input data structure is done by configuring the [IN_D_STRCT] field of the eTVPE as "000" and is allowed only if the following is applied:

- The [HE] bit of the MTVCP parameter (see **Section 26.4.3.2.3**, *eTVPE Initialization Parameter*) is set.
- The [STD] bit of the eTVPE BD is set, indicating 3GLTE job.

For the LTE HARQ input data structure, the eTVPE performs as shown in **Figure 26-6**.

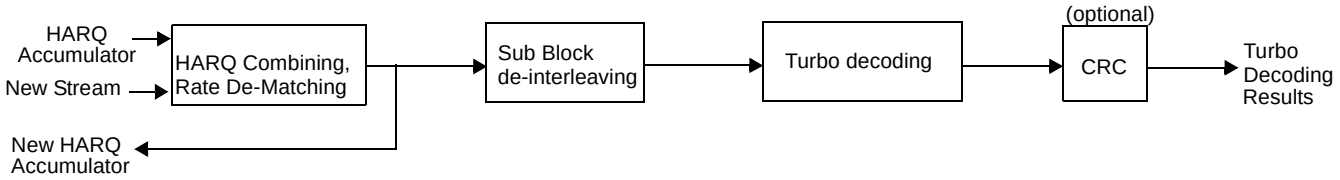


Figure 26-6. eTVPE Processing Flow for LTE HARQ Input Data Structure

The MAPLE-B2 expects two types of input vectors to be fetched into the eTVPE:

- HARQ accumulator buffer generated by the eTVPE while processing this Code Block in previous Redundancy Version transmission. If this is the first transmission of the Code Block and there is no HARQ accumulator exist than the [FTH] bit of the eTVPE BD must be set indicating that no HARQ accumulator buffer is required.
- New Redundancy Version transmission (channel data) to be combined with the existing HARQ accumulator buffer during eTVPE processing.

In addition to all Turbo related eTVPE BD fields, when working with LTE HARQ input data structure, the following fields must be set:

- [BASE_ADDR]: Pointer to the system memory where the new Redundancy Version transmission vector is located.
- All HARQ related fields as described in **Section 26.4.3.2.6.2**, *HARQ Combining*.

As described above, after the HARQ combining stage is completed in the eTVPE, the MAPLE-B2 outputs the new HARQ accumulator data into the system memory, while the eTVPE continues with its processing flow. For details on eTVPE processing flow, refer to **Section 26.4.3.2.6**, *eTVPE Processing*.

26.4.3.2.5.3 WiMAX HARQ Input Data Structure

The WiMAX HARQ input data structure provides support for HARQ combining, rate de-matching and Sub-Block De-interleaving processing as appears in the WiMAX OFDMA turbo decoding (IEEE 802.16e-2009 standard), and therefore is valid only for WiMAX technology.

Configuring the MAPLE-B2 to work with WiMAX HARQ input data structure is done by configuring the [IN_D_STRCT] field of the eTVPE be to 001 and is allowed only if the following applies:

- The [HE] bit of the MTVCP parameter (see **Section 26.4.3.2.3**, *eTVPE Initialization Parameter*) is set.
- The MAPLE-B2 is working under 3GLTE and WiMAX operation mode.

In WiMAX HARQ input data structure the eTVPE performs the operations shown in **Figure 26-7**.

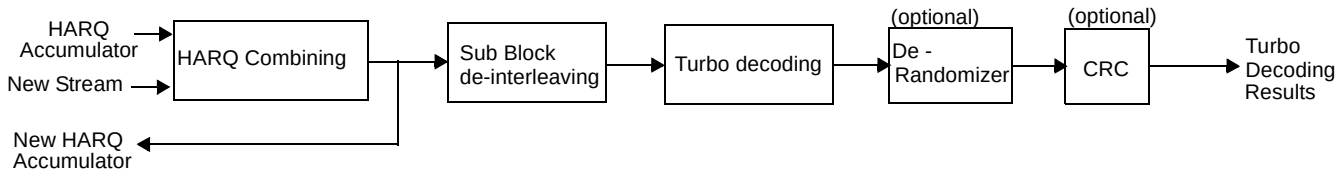


Figure 26-7. eTVPE Processing Flow for WiMAX HARQ Input Data Structure

The MAPLE-B2 expects two types of input vectors to be fetched into the eTVPE:

- HARQ accumulator buffer generated by the eTVPE while processing this Code Block in previous transmission. If this is the first transmission of the Code Block and there is no HARQ accumulator exist than the [FTH] bit of the eTVPE BD must be set indicating that no HARQ accumulator buffer is required.
- New Channel Data transmission to be combined with the existing HARQ accumulator buffer during eTVPE processing.

In additional to all Turbo related eTVPE BD fields, when working with WiMAX HARQ input data structure the following fields must be set:

- [BASE_ADDR]: Pointer to the system memory where the new Channel Data transmission vector is located.
- All HARQ related fields as described in **Section 26.4.3.2.6.2, *HARQ Combining***.

As described above, after the HARQ combining stage is completed in the eTVPE, the MAPLE-B2 outputs the new HARQ accumulator data into the system memory, while the eTVPE continues with its processing flow. For details on eTVPE processing flow, refer to **Section 26.4.3.2.6**.

26.4.3.2.5.4 E-DCH HARQ with Mixed Vector Input Data Structure

The E-DCH HARQ with Mixed Vector input data structure provides support for Rate De-Matching and HARQ combining processing as appears in the 3GPP TS 25.212 Multiplexing and channel coding standard, and therefore is valid only for UMTS jobs.

- Configuring the MAPLE-B2 to work with E-DCH HARQ with Mixed Vector input data structure is done by configuring the [IN_D_STRCT] field of the eTVPE to 010 and is allowed only if the following applies:
- The [HE] bit of the MTVCP parameter (see **Section 26.4.3.2.3**, *eTVPE Initialization Parameter*) is set.
- The [SEP_V] bit of the MTVCP parameter (see **Section 26.4.3.2.3**, *eTVPE Initialization Parameter*) is cleared.
- The MAPLE-B2 is working under 3G operation mode (3GLTE and UMTS technologies).
- The [STD] bit of the eTVPE BD is reset, indicating UMTS job.

In E-DCH HARQ with Mixed Vector input data structure, the eTVPE performs the procedure shown in **Figure 26-8**.

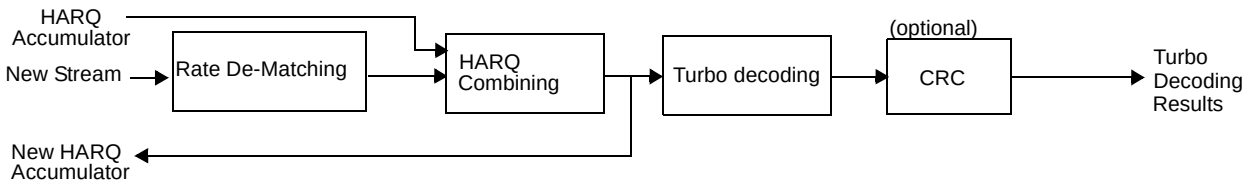


Figure 26-8. eTVPE Processing Flow for E-DCH HARQ Input Data Structure

The MAPLE-B2 expects two types of input vectors to be fetched into the eTVPE:

- HARQ accumulator buffer generated by the eTVPE while processing this Code Block in previous transmission. If this is the first transmission of the Code Block and there is no HARQ accumulator exist than the [FTH] bit of the eTVPE BD must be set indicating that no HARQ accumulator buffer is required.
- New Channel Data transmission to be Rate de-matched and then combined with the existing HARQ accumulator buffer during eTVPE processing.

In additional to all Turbo related eTVPE BD fields, when working with E-DCH HARQ input data structure the following fields must be set:

- [BASE_ADDR]: Pointer to the system memory where the new Channel Data transmission vector is located.
- [BUF_SIZE]: The size of the new Channel Data transmission vector to be fetched by MAPLE-B2 into the eTVPE. See **Section 26.4.3.2.6.1, *E-DCH Rate De-Matching*** For details, on calculating this field.
- HARQ related fields as described in **Section 26.4.3.2.6.2, *HARQ Combining***.

26.4.3.2.5.5 E-DCH HARQ with Separate Vectors Input Data Structure

The E-DCH HARQ with Separate Vectors input data structure provides support for Rate De-Matching and HARQ combining processing as appears in the 3GPP TS 25.222 Multiplexing and channel coding standard, and therefore is valid only for UMTS jobs.

Configuring the MAPLE-B2 to work with E-DCH HARQ with Separate Vectors input data structure is done by configuring the [IN_D_STRCT] field of the eTVPE be to 011 and is allowed only if the following applies:

- The [HE] bit of the MTVCP parameter (see eTVPE Initialization Parameter) is set.
- The [SEP_V] bit of the MTVCP parameter (see eTVPE Initialization Parameter) is set.
- The MAPLE-B2 is working under 3G operation mode (3GLTE and UMTS technologies).
- The [STD] bit of the eTVPE BD is reset, indicating UMTS job.

In E-DCH HARQ with Mixed Vector input data structure, the eTVPE performs the procedure shown in **Figure 26-9**.

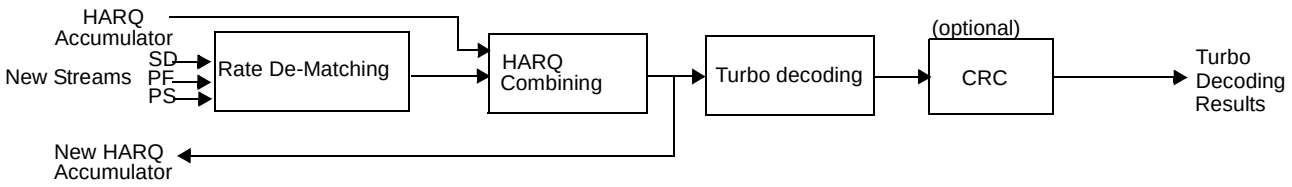


Figure 26-9. eTVPE Processing Flow for E-DCH HARQ Input Data Structure

The MAPLE-B2 expects the following four vectors to be fetched into the eTVPE:

- HARQ accumulator buffer generated by the eTVPE while processing this Code Block in previous transmission. If this is the first transmission of the Code Block and there is no HARQ accumulator exist than the [FTH] bit of the eTVPE BD must be set indicating that no HARQ accumulator buffer is required.
- SD vector of the new Channel Data transmission before Rate de-matching.
- PF vector of the new Channel Data transmission before Rate de-matching.
- PS vector of the new Channel Data transmission before Rate de-matching.

In additional to all Turbo related eTVPE BD fields, when working with E-DCH HARQ with Mixed Vectors input data structure the following fields must be set:

- [BASE_ADDR]: Pointer to the system memory where the SD vector of the new channel data vector is located.
- [VEC_OFFSET]: The offset of the beginning of the PF/PS vectors with respect to the beginning of the SD/PF vectors. The following table describe the expected start address with respect to [BASE_ADDR] and [VEC_OFFSET]:

Table 26-8. Expected Address of Input Vectors

Vector	Start Address
SD	[BASE_ADDR]
PF	[BASE_ADDR] + [VEC_OFFSET]
PS	[BASE_ADDR] + (2 * [VEC_OFFSET])

- [BUF_SIZE]: The size of the SD vector of the new Channel Data transmission to be fetched by MAPLE-B2 into the eTVPE. See
- [PF_BUF_SIZE]: The size of the PF vector of the new Channel Data transmission to be fetched by MAPLE-B2 into the eTVPE.
- [PS_BUF_SIZE]: The size of the PS vector of the new Channel Data transmission to be fetched by MAPLE-B2 into the eTVPE.
- HARQ related fields as described in **Section 26.4.3.2.6.2**, *HARQ Combining*.

See **Section 26.4.3.2.6.1**, *E-DCH Rate De-Matching* for details on calculating the BUF_SIZE fields. As described previously, after the HARQ combining stage is completed in the eTVPE, the MAPLE-B2 outputs the new HARQ accumulator data into the system memory, while the eTVPE continues with its processing flow. For details on eTVPE processing, refer to **Section 26.4.3.2.6**, *eTVPE Processing*

26.4.3.2.5.6 Sub-Block Interleaved Input Data Structure

The Sub-Block Interleaved input data structure provides support for Sub-Block De-interleaving processing as appears in the WiMAX OFDMA turbo decoding (IEEE 802.16e-2009 standard), and the 3GLTE turbo decoding, and therefore is valid only for 3GLTE and WiMAX technologies. Configuring the MAPLE-B2 to work with Sub-Block Interleaved input data structure is done by configuring the [IN_D_STRCT] field of the eTVPE to 100 and is allowed only if the following applies:

- The [HE] bit of the MTVCP parameter (see eTVPE Initialization Parameter) is reset.

In Sub-Block Interleaved input data structure the eTVPE performs the procedure shown in **Figure 26-10**.

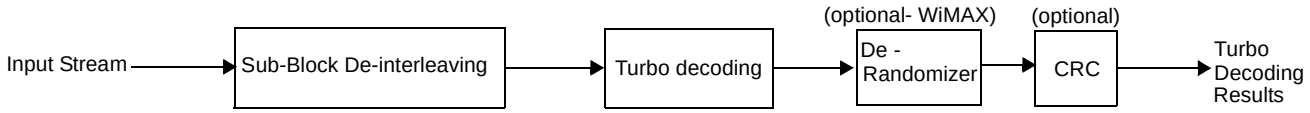


Figure 26-10. eTVPE Processing Flow for Sub-Block Interleaved Input Data Structure

The MAPLE-B2 expects one input vector, pointed by the [BASE_ADDR] field of the eTVPE BD, which should point to the system level post HARQ combining result buffer maintained by the host. In this input data structure, the HARQ combining is assumed being performed externally to MAPLE-B2.

26.4.3.2.5.7 UMTS Mixed Input Data Structure

The UMTS Mixed input data structure provides Turbo decoding support for the UMTS (FDD) standard in the case of Rate De-matching and HARQ processing are performed externally by the host. Configuring the MAPLE-B2 to work with UMTS Mixed input data structure is done by configuring the [IN_D_STRCT] field of the eTVPE be to 101 and is allowed only if the following applies:

- The [HE] bit of the MTVCP parameter (see eTVPE Initialization Parameter) is reset.
- The MAPLE-B2 is working under 3G operation mode (3GLTE and UMTS technologies) with the [STD] bit of the eTVPE BD is reset (UMTS standard).

In UMTS Mixed input data structure the eTVPE performs the procedure shown in **Figure 26-11**.

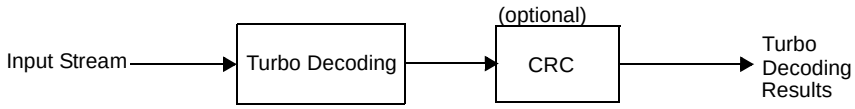


Figure 26-11. eTVPE Processing Flow for UMTS Mixed Input Data Structure

For UMTS Mixed input data structure the MAPLE-B2 expects single input stream, pointed by the [BASE_ADDR] field of the eTVPE BD. The expected structure of the stream is shown in **Figure 26-12**.

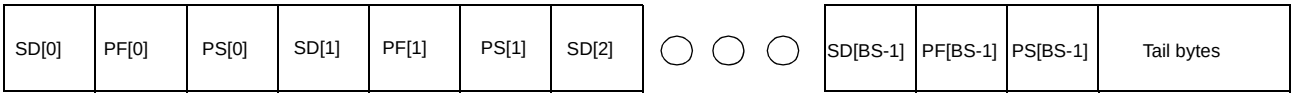


Figure 26-12. UMTS Mixed Stream Structure

The expected Tail bytes structure at the end of the stream is shown in **Figure 26-214**.

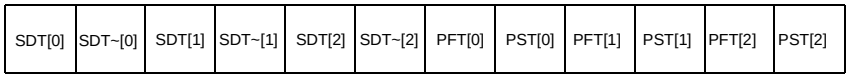


Figure 26-13. UMTS Mixed Tail Stream Structure

26.4.3.2.5.8 Separate Vectors Input Data Structure

The Separate Vectors input data structure provides Turbo decoding support for the 3GLTE, WiMAX (both 802.16e and 802.16m), and UMTS (TDD) technologies in the case of Rate De-matching and HARQ processing are performed externally by the host. Configuring the MAPLE-B2 to work with Separate Vectors input data structure is done by configuring the [IN_D_STRCT] field of the eTVPE be to 110 and is allowed only if the following applies: The [HE] bit of the MTVCP parameter (see eTVPE Initialization Parameter) is reset.

In Separate Vectors input data structure the eTVPE performs the procedure shown in **Figure 26-14**.

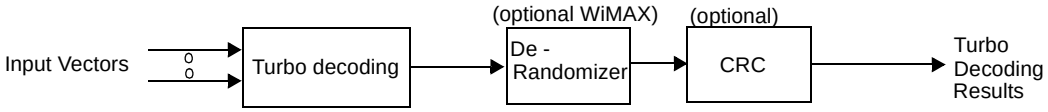


Figure 26-14. eTVPE Processing Flow for Separate Vectors Input Data Structure

In Separate Vectors input data structure, the MAPLE-B2 expects to find the different input data types (SD, PF, PS) in separate vectors as follows:

- For the 3GLTE and UMTS standards the expected vectors are: SD, PFa, PSa
- For the WiMAX technology (Duo-Binary decoding) the expected vectors are: SDa, SDb, PFy, PFw, PSy, PSw

The Tail bits should be added at the end of each vector as described in **Figure 26-15**.

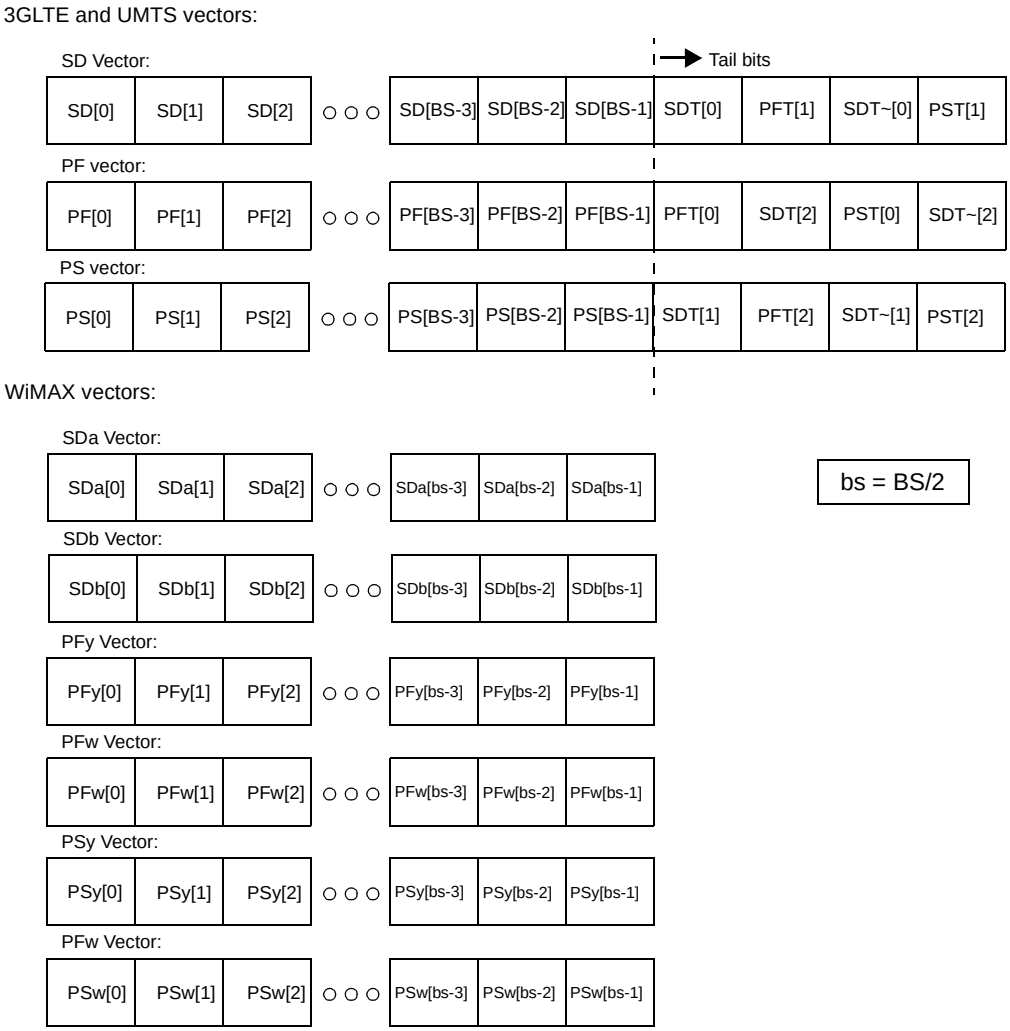


Figure 26-15. Expected Vectors Structure for Separate Vectors

Note: For the 3GLTE standard, the MAPLE-B2 expects three vectors with equal size of [BS] + 4. It requires zero padding of the beginning of the SD and PF vectors with number of zeros equal to the number of filler bits required during the encoding.

The MAPLE-B2 expect to find each of the input vector as described in **Table 26-9**:

Table 26-9. Expected Address of Input Vectors

Standard	Offset of Vector with respect to [BASE_ADDR]					
	no offset	[VEC_OFFSET]	2*[VEC_OFFSET]	3*[VEC_OFFSET]	4*[VEC_OFFSET]	5*[VEC_OFFSET]
3GLTE,UMTS	SD	PF	PS			
WiMAX	SDa	SDb	PFy	PFw	PSy	PSw

where the [VEC_OFFSET] field is a 16 bits field in the eTVPE BD.

26.4.3.2.5.9 Viterbi Periodically Punctured Stream Input Data Structure

The Viterbi Periodic Depuncturing input data structure is the only input data structure supported for Viterbi processing. As such, the MAPLE-B2 assume it is enabled whenever Viterbi processing is enabled ([ALG]=1).

The Periodically Punctured Stream input data structure is targeted for Viterbi processing only. By setting the [ALG] bit for Viterbi job ([ALG] = 1), the MAPLE-B2 assumes Periodically Punctured Stream input data structure is applied. For the Periodically Punctured Stream input data structure, the eTVPE performs the procedure shown in **Figure 26-16**.

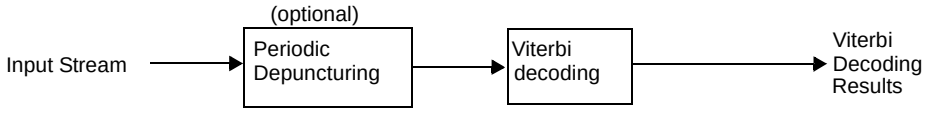


Figure 26-16. eTVPE Processing Flow for Periodically Punctured Stream Input Data Structure

For the Periodically Punctured Stream data structure, the MAPLE-B2 expects to find the relevant input data types in a single vector stream pointed by [BASE_ADDR] field of the eTVPE BD. The order of the symbols (after depuncturing) within the data stream is prescribed in **Table 26-10**.

Table 26-10. 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]

From **Table 26-10**:

- For Viterbi rate 1/2 processing, the expected input stream (after depuncturing) 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 depuncturing) 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 depuncturing) is:

P0[0]	P1[0]	P2[0]	P3[0]	P0[1]	P1[1]	P2[1]	P3[1]	○	○	○	○

When working with the Viterbi Periodic Depuncturing input data structure, 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 input data stream in bytes. The value of [BUF_SIZE] should be calculated according to the following:

1. if ([ZTTB] == 1)

   tmp_buf_size = [BS] + *K* -1;

   else

   tmp_buf_size = [BS];
2. if (*EncRate* == 2)

   tmp_buf_size = (tmp_buf_size + 3) & 0xFFFC;

   else

   tmp_buf_size = (tmp_buf_size +1) & 0xFFFE;
3. [BUF_SIZE] = roundup(tmp_buf_size**PuncRate***EncRate* + 31) & 0xFFF0;

where:

[ZTTB], [BS]. Fields from the eTVPE 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, and so forth.

For example: If [BS]= 164, [ZTTB] =1, *K*=9, Encoding rate of 1/3 and Puncturing rate of 2/3:

1. tmp_buf_size = 164 + 8 = 172
2. tmp_buf_size = (172 + 1) & 0xFFFE = 172
3. [BUF_SIZE] = roundup(172 * (2/3)*3 + 31) & 0xFFF0 = 375 & 0xFFF0 = 368

26.4.3.2.5.10 eTVPE Input Data Structures Summary

Table 26-11 describe the supported eTVPE input data structure with respect to the supported standard

Table 26-11. eTVPE Input Data Structures Summary

Input Data Structure	3GLTE	UMTS	WiMAX (802.16e)	WiMAX (802.16m)
LTE HARQ	Yes	No	No	No
WiMAX HARQ	No	No	Yes	No
E-DCH HARQ with Mixed Vector	No	Yes	No	No
E-DCH HARQ with Separate Vectors	No	Yes	No	No
Sub Block Interleaved	Yes	No	Yes	No
UMTS Mixed	No	Yes	No	No
Separate Vectors	Yes	Yes	Yes	Yes

26.4.3.2.6 eTVPE Processing

The following sections describe the processing algorithms performed in the eTVPE for the different supported algorithms and input data structures

26.4.3.2.6.1 E-DCH Rate De-Matching

The E-DCH Rate De-Matching processing is performed during E-DCH HARQ with Mixed Vector and E-DCH HARQ with Separate Vectors input data structure processing flows only. For these input data structure, the eTVPE executes Rate-De-Matching using depuncturing or using a de-repetition scheme. This scheme provides support for E-DCH, as defined in 3GPP TS 25.212/222 Multiplexing and channel coding, section 4.8.

Since the 3GPP TS 25.212/222 standard defines the rate-matching process over transport block (TB) and not code block (CB) and since the eTVPE works on CB only, the following parameters in the eTVPE BD must be initialized:

- *[E_PARAM_PTR]*. A pointer to a data structure in the system memory which includes the required parameters for the eTVPE for the Rate De-matching processing.
- *[DPNC]*. Indication to the eTVPE whether depuncturing scheme is required ([DPNC]=1) or de-repetition ([DPNC]=0).

The expected parameter structure pointed to by the [E_PARAM_PTR] pointer is shown in **Figure 26-17**.

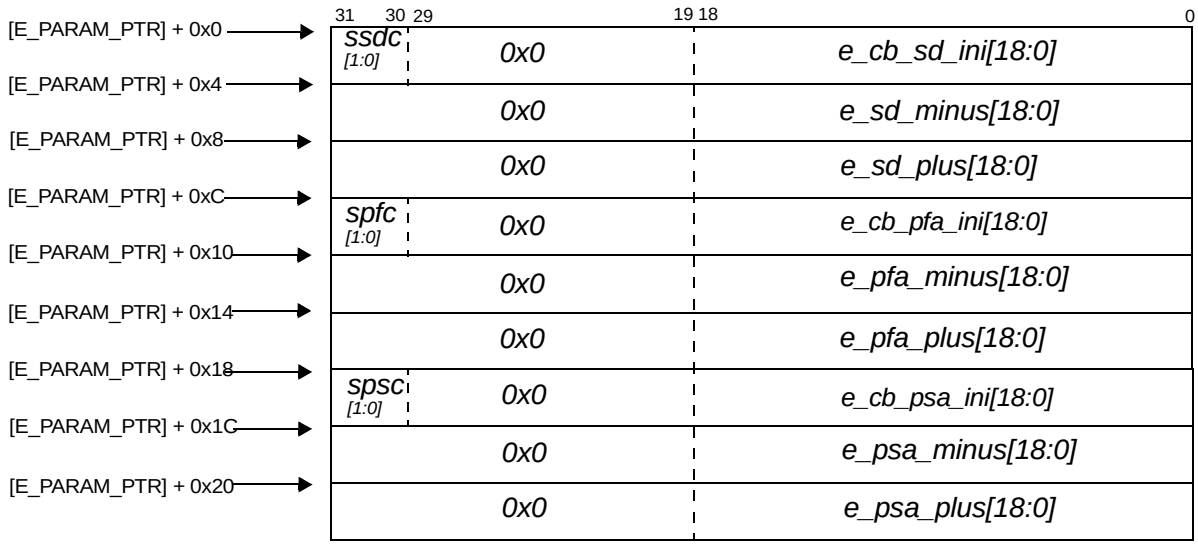


Figure 26-17. E parameters Data Structure Description

where:

e_cb_<xx>_ini[18:0]— The e_{init} parameter for SD, PF and PS (as defined in standard) with the additional offset of the current code block in the transport block for the systematic data. Details on calculating these parameters is described in **Section 26.4.3.2.6.1.4**, *Code Block e_ini Calculation.*

e_<xx>_minus[18:0]—The e_{minus} parameter for SD, PF and PS (as defined in standard)

e_<xx>_plus[18:0]—The e_{plus} parameter for SD, PF and PS (as defined in standard)

s<xx>c[1:0]—Skip SD, PF and PS count value. Details on calculating these parameters is described in **Section 26.4.3.2.6.1.2**, *Skip Count Calculation*

26.4.3.2.6.1.1 Rate Matched Code Block Size Calculation

To find the size of each code block before rate de-matching, the following calculations should be performed for each data type (SD, PF, and PS)

```
rate_matched_bits = rounddown(([BS]+ 4) * e_minus /e_plus)
e_ini_limit = (([BS]+ 4)* e_minus)mod(e_plus)
if (e_ini_limit >= e_cb_ini)
     rate_matched_bit = rate_matched_bits + 1
if ([DPNC] ==0) //repetition
     rate_matched_code_block_size = [BS] + 4 + rate_matched_bits
else //puncturing
     rate_matched_code_block_size = [BS] + 4 - rate_matched_bits
```

26.4.3.2.6.1.2 Skip Count Calculation

Calculating the skip count for each data type (s<xx>c[1:0] parameters) is performed as follows:

```
n - index of current CB in TB (n = 0, 1, 2...)
sd_prev_rm_bits(n) = 0
pf_prev_rm_bits(n) = 0
ps_prev_rm_bits(n) = 0

for (i=0; i < n; i++)
{
     sd_prev_rm_bits(n) = sd_prev_rm_bits(n) + sd_rate_matched_code_block_size(i)
     pf_prev_rm_bits(n) = pf_prev_rm_bits(n) + pf_rate_matched_code_block_size(i)
     ps_prev_rm_bits(n) = ps_prev_rm_bits(n) + ps_rate_matched_code_block_size(i)
}

min_prev_rm_bits(n) = min(sd_prev_rm_bits(n),pf_prev_rm_bits(n),ps_prev_rm_bits(n))

if ([DPNC]==0) //repetition
     ssdc[1:0] = sd_prev_rm_bits(n) - min_prev_rm_bits(n)
     spfc[1:0] = pf_prev_rm_bits(n) - min_prev_rm_bits(n)
     spsc[1:0] = ps_prev_rm_bits(n) - min_prev_rm_bits(n)
else // puncturing
     ssdc[1:0] = 0
     spfc[1:0] = 0
     spsc[1:0] = 0
```

Figure 26-18 describe a Transport Block example which consist of three Code Blocks and was repeated using the rate matching functionality at the transmitter. The figure shows the required input data into the TVPE for the middle CB and the skip count bits location.

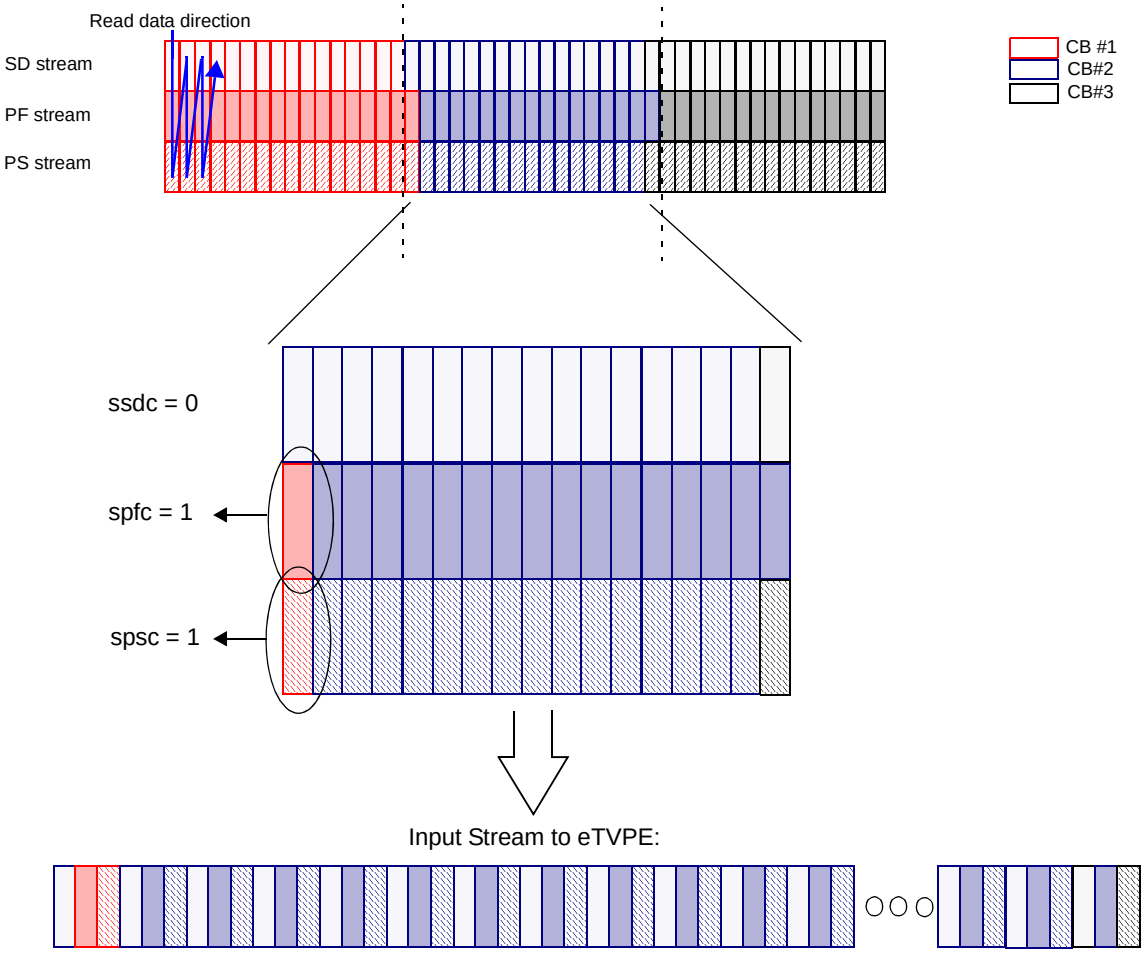


Figure 26-18. Skip Count Bits Location Example

26.4.3.2.6.1.3 Buffer Size Calculations

Calculating the BUF_SIZE fields is done using the parameters calculated in the previous sections according to the following:

- For the E-DCH HARQ with Mixed Vector input data structure:

```
[BUF_SIZE] = 3 * max(sd_rate_matched_code_block_size + ssdc,
pf_rate_matched_code_block_size + spfc, ps_rate_matched_code_block_size + spsc);
```

- For the E-DCH HARQ with Separate Vectors input data structure:

```
[BUF_SIZE] = 'sd_rate_matched_code_block_size'
[BUF_SIZE_PF] = 'pf_rate_matched_code_block_size'
[BUF_SIZE_PS] = 'ps_rate_matched_code_block_size'
```

26.4.3.2.6.1.4 Code Block e_ini Calculation

Calculating the e_cb_ini parameter for each of the data type is performed for each CB in the TB according to the following calculation:

```
e_cb_ini = [(e_ini - ([BS] + 4)* n *e_minus - 1)mod(e_plus)] + 1
e_cb_ini = e_cb_ini + s<xx>c*e_minus
```

where:

e_ini, *e_minus*, *e_plus*- As defined in standard.
[BS] - Block Size field in eTVPE BD
n - Current index of CB where for first CB “n=0”.
s<xx>c - Skip count parameters for each data type as described in Skip Count Calculation.

The calculated values should be used for the ***e_cb_<xx>_ini[18:0]*** parameters.

26.4.3.2.6.1.5 Parameters Calculation Example

The following example describe the parameters calculations required to generate eTVPE BDs for few CBs related to single TB in E-DCH HARQ with Mixed Vector input data structure:

Assuming TB with the size of 16,003 bits to be transmitted. The standard defines the following:

```
num_of_cb = roundup(16003/5114) = 4 // Number of CBs
cb_size = roundup(16003/4) = 4001 // Last CB is added with single padding bit
toal_bit_to_transmit = 3*(cb_size*num_of_cb) + 12*num_of_cb = 48060 // Ne,j
Nsys = Np1 = Np2 = 16020
```

Puncturing is required if $N_{e,data,j}$ is smaller than $N_{e,j}$ and repetition is required if $N_{e,data,j}$ is larger than $N_{e,j}$. In the example it is assumed $N_{e,data,j}$ = 26,007.

The s parameter indicated if systematic data is preferred over parity data. In the example it is assumed s=1, that is, systematic data is preferred, therefore:

$N_{t,sys} = N_{sys} = 16020$
$N_{t,p1} = \text{rounddown}((N_{data} - N_{t,sys}/2)) = 4993$
$N_{t,p2} = \text{roundup}((N_{data} - N_{t,sys}/2)) = 4994$

Calculating the e parameters according to the standard definition is described in the following table:

Table 26-12. e Parameters Calculation Results

Data Type	e_{plus}	e_{minus}	e_ini (first CB)
SD	N_{sys} = 16,020	0	16,020
PF	2 * N_{p1} = 32,040	2*(N_{p1}-$N_{t,p1}$) = 22,054	16,020
PS	N_{p2} = 16,020	N_{p2} - $N_{t,p2}$ = 11,028	16,020

Once all standard parameters are calculated, eTVPE BD parameters can be calculated. **Table 26-13** describe the parameters calculations for each of the BC in the TB

Table 26-13. eTVPE BD parameters calculations

Parameter	CB #1	CB #2	CB #3	CB #4
sd_rate_matched_code_block_size	4005	4005	4005	4005
pf_rate_matched_code_block_size	1249	1250	1249	1249
ps_rate_matched_code_block_size	1249	1249	1249	1249
ssdc	0	0	0	0
spfc	0	0	0	0
spsc	0	0	0	0
[BUF_SIZE]	6504	6504	6504	6504
e_cb_sd_ini	16,020	16,020	16,020	16,020
e_cb_pf_ini	16,020	8010	32,040	24,030
e_cb_ps_ini	16,020	16,020	16,020	16,020

26.4.3.2.6.2 HARQ Combining

The HARQ combining is enabled by setting the HARQ Enable ([HE]) bit of the MTVCP parameter during MAPLE-B2 initialization. It is performed during one of the following input data structures: LTE HARQ, WiMAX HARQ, and E-DCH HARQ.

26.4.3.2.6.2.1 eTVPE BD Parameter Initialization

When working with HARQ combining enabled, the following parameters of the eTVPE BD must be initialized:

1. [HA_IN_BASE_ADDR]. Pointer to the system memory where the HARQ accumulator buffer is located. MAPLE-B2 uses this pointer to fetch the HARQ accumulator buffer. if [FTH] bit is set, this field is ignored as no HARQ accumulator input buffer is expected.
2. [HA_OUT_BASE_ADDR]. Once HARQ combing is completed and the [HAOE] bit is enabled, the MAPLE-B2 use this pointer to write the new HARQ accumulator.
3. [HAOE]. HARQ Output Enable indication. Should be set if the HARQ accumulator buffer after the combining stage with the new stream is required to be output by MAPLE-B2 for future Redundancy Versions Transmissions.
4. [IHBSZ]. The size of the new Redundancy Version transmission vector to be fetched by MAPLE-B2 into the eTVPE. For UMTS standard this field is ignored.
5. [IHBSA]. The relative address of the new Redundancy Version transmission vector in the accumulator HARQ buffer. For UMTS standard this field is ignored. See **Section 26.4.3.2.6.2.3**, *Calculating the IHBSA Parameter* for details on calculating this parameter.
6. [FTH]. First HARQ buffer indication. Should be set if the current job is the first Redundancy Version transmission. If set, no HARQ accumulator is required to be fetched.

Figure 26-19 describes an example of the use of the above parameters.

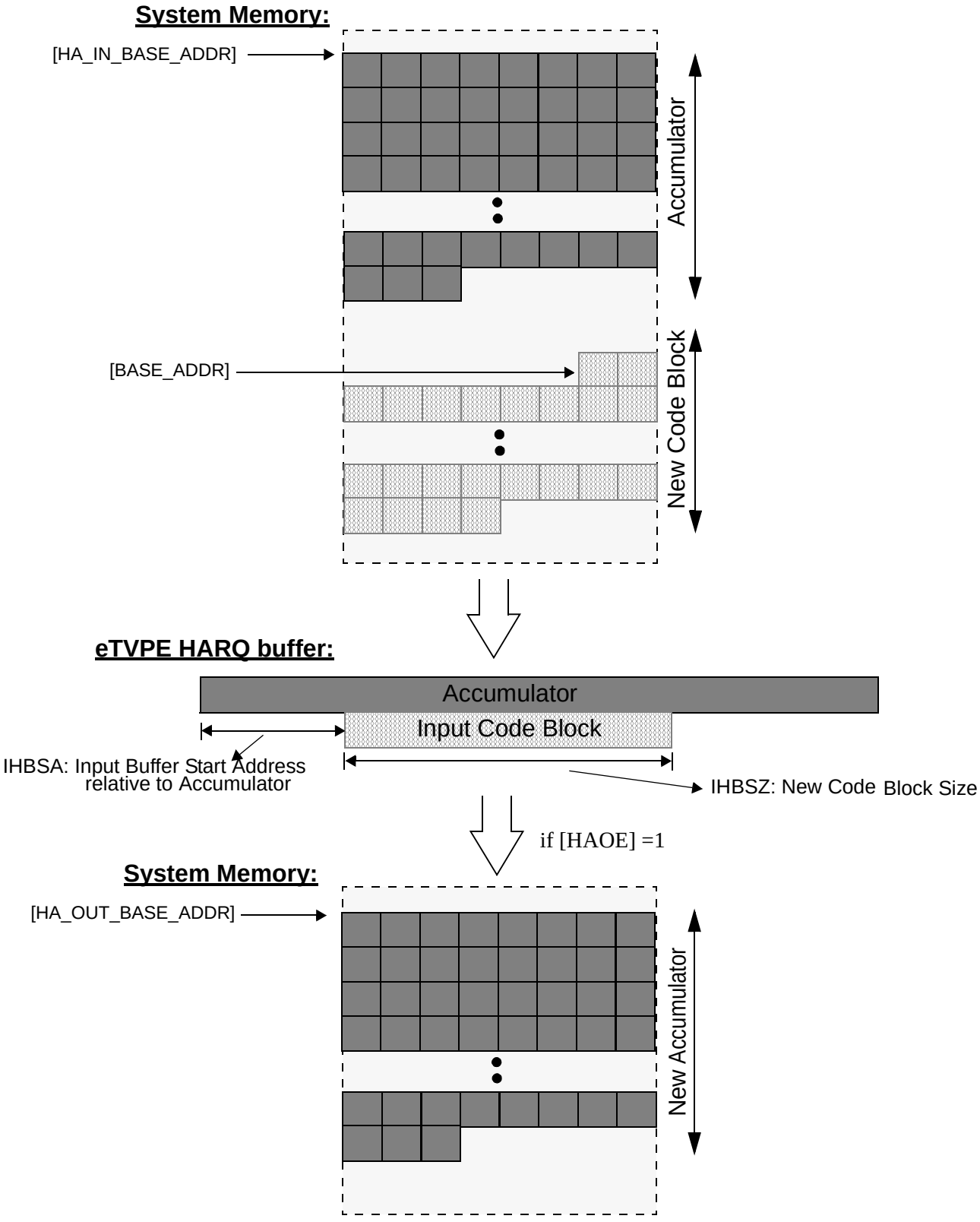


Figure 26-19. HARQ BD Parameters Usage Example

26.4.3.2.6.2.2 Setting the HARQ Combining Parameters

In addition to the above parameters, the following HARQ combining parameters must be set:

1. [HUS]. HARQ Up Scale indication. Allow up-scaling of the new input stream to be accumulated to the HARQ accumulator by 0 (no up-scaling), 2, 4 or 8 (shift left of each input sample by 0,1,2 or 3).
2. [W1E], [W2E], [W3E], [W1], [W2], [W3]. Weight Enable and Weight value. The HARQ accumulator and the new input stream are combined according to the following weight scheme:

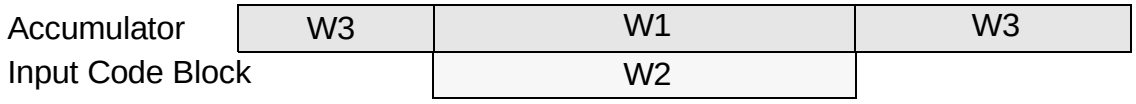


Figure 26-20. HARQ Weight Scheme

W1 - Data from HARQ accumulator which is covered by new input stream.

W2 - Data of new input stream

W3 - Data from HARQ accumulator which is **not** covered by new input stream.

Calculation each input of new HARQ accumulator for data which is covered by new input stream:

case ({[W1E],[W2E]})

‘00’: HARQ_new = HARQ_old + (DATA_new << [HUS])

‘01’: HARQ_new = HARQ_old + (W2/256)*(DATA_new << [HUS])

‘10’: HARQ_new = (W1/256)*HARQ_old + (DATA_new << [HUS])

‘11’: HARQ_new = (W1/256)*HARQ_old + (W2/256)*(DATA_new << [HUS])

Calculation each input of new HARQ accumulator for data which is **not** covered by new input stream:

```
if ([W3E] == 0)
    HARQ_new = HARQ_old
else if ([W3E] == 1)
    HARQ_new = HARQ_old * ([W3]/256)
```

During the HARQ combining calculations, overflow of LLRs may occur. In case of overflow, the eTVPE saturate the LLR results and count the total LLRs which required saturation. The [HSC] status field of the eTVPE BD holds the number of LLRs which were saturated due to overflow in the HARQ combing process.

26.4.3.2.6.2.3 Calculating the IHBSA Parameter

Calculation of the IHBSA parameter differs between the technologies:

For WiMAX technology, the IHBSA parameter equals the $S_{k,i}$ parameter as calculated in the standard.

For 3GLTE technology, calculating the IHBSA parameter is based on the K_0 parameter calculations as following:

```
Num_of_dummy_bits = [32 - (BS+4)%32]%32
IHBSA = K0 - IHBSATable[ RV<<2 + (Num_of_dummy_bits-4)>>3];
```

where:

```
BS - Block Size Parameter from eTVPE Buffer Descriptor.
K0 - Calculated as described in the standard.
RV - Redundancy Version Index (described in standard)
IHBSATable[0] = 1
IHBSATable[1] = 1
IHBSATable[2] = 2
IHBSATable[3] = 2
IHBSATable[4] = 4
IHBSATable[5] = 10
IHBSATable[6] = 17
IHBSATable[7] = 23
IHBSATable[8] = 8
IHBSATable[9] = 20
IHBSATable[10] = 32
IHBSATable[11] = 44
IHBSATable[12] = 10
IHBSATable[13] = 30
IHBSATable[14] = 48
IHBSATable[15] = 66
```

26.4.3.2.6.3 Sub-Block De-interleaving

The Sub-Block Interleaved Vectors input data structure provides support for Turbo decoding, as it appears in the WiMAX OFDMA turbo decoding (**IEEE** 802.16e-2009 standard), and the 3GLTE (Evolved UTRA) sub-block de-interleaving and rate matching, and therefore is valid only during the following input data structures: HARQ LTE, HARQ WiMAX, and Sub-Block Interleaved.

There are two types of sub-block de-interleaving executed by the eTVPE:

1. Singles sub-block de-interleaving. The eTVPE performs sub-block de-interleaving on one type of data, that is, on SD only (SDa only and SDb only for WiMAX). **Figure 26-21** describes an example of single sub-block de-interleaving:

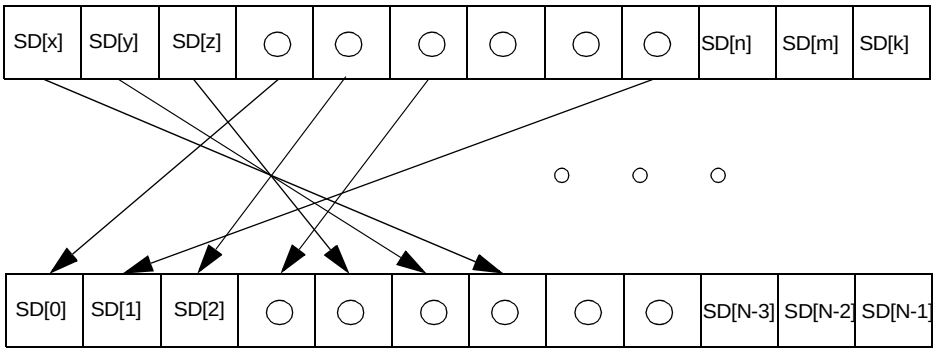


Figure 26-21. Single Sub Block De-Interleaving Example

2. Pairs sub-block de-interleaving. The eTVPE performs sub-block de-interleaving on pairs of data types, that is, on PF+PS, PFy+PSy and PFw+PSw. **Figure 26-22** describes an example of pairs sub-block de-interleaving:

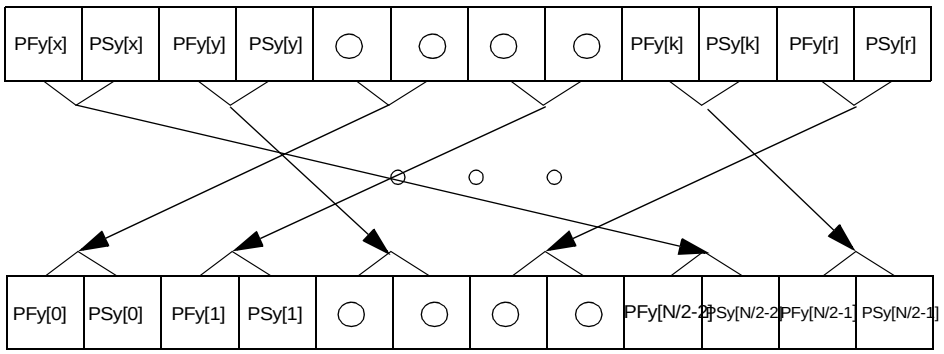


Figure 26-22. Pairs Sub Block De-Interleaving Example

Table 26-14 describes the type of sub-block de-interleaving executed for each of the technologies:

Table 26-14. Sub Block De-Interleaving With Respect to standard

standard	Single Sub-Block De-Interleaving	Pairs Sub-Block De-Interleaving
3GLTE	SD	PF+PS
WiMAX	SDa SDb	PFy+PSy PFw+PSw

26.4.3.2.6.4 Turbo Decoding

The MAPLE-B2 supports Turbo decoding for the different technologies according to its operation mode:

- For 3G operation mode:
  - 3GLTE technology
  - UMTS technology
- For WiMAX operation mode:
  - 3GLTE technology
  - WiMAX technology

Once the MAPLE-B2 operation mode is set during initialization (see **Section 26.4.1**, *Initialization*), it can process Turbo decoding as per the technologies supported in that operation mode only. For each operation mode, it can dynamically switch between the supported technologies.

26.4.3.2.6.4.1 Turbo Channel Data Interleaving

Before the eTVPE can start its decoding operation, the interleaved channel data must be generated. The eTVPE uses its internal parameters/registers, with additional information from the BD (such as Block Size) to internally determine the interleaving configuration. The interleaving process is executed internally and independently by the eTVPE.

26.4.3.2.6.4.2 Turbo Processing Implementation

The following equations describe the eTVPE Turbo implementation of the MAX* 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, and P_{n+1} are the 8 bit received systematic and parity channel data for bit n and n+1, and Ext_n, and 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 and 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 and v:

- ACS2 definition when using LinearLogMap is:

$$ACS2L = cmax(u, v) + f(u, v, lcf)$$

where lcf is the [LLMAP_LCF] field in the eTVPE 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:

$$ACS8M(a, b, c, d, e, f, g, h) = ACS2M((ACS4M[a, b, c, d]), ACS4M[e, f, g, h])$$

26.4.3.2.6.4.3 Number of Turbo Processing Elements (DREs)

The eTVPE includes four DRE (Dual Recursion Element) engines used for Turbo processing. For different block sizes, it is possible to activate 1 DRE engine, 2 DRE engines, or 4 DRE engines. When activating more than one such engine during a given Turbo processing session, the eTVPE 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 eTVPE. The MAPLE-B2 allows two modes of operation with respect to the number of DRE engines:

- *Automatic number of DRE engines*. By setting the [ANDRE] bit of the MTVCP parameter (see **Section 26.5.2.3**, *MAPLE eTVPE Configuration Parameter (MTVCP)*), the eTVPE sets the number of DRE engines automatically to achieve maximum throughput performance.
- *BD controlled number of DRE engines.* By clearing the [ANDRE] bit of the MTVCP parameter, the eTVPE uses the number of DRE engines as configured in the [NDRE] field of the eTVPE BD.

As mentioned previously, when setting the [ANDRE] bit in the MTVCP parameter, the eTVPE internally configures its number of DRE engines to the maximum possible for a given Block Size. **Table 26-15** lists the maximum number of DRE engines used by the eTVPE if the [ANDRE] bit is set. The limitations also apply when working when the [ANDRE] bit is cleared.

Table 26-15. Maximum Possible Number of DRE Engines

Standard	Maximum Number of DRE Engines (NDRE)
3GLTE	if ([BS] < 64) {NDRE = 2} else {NDRE = 4}
WiMAX	if ([BS] < 96) {NDRE = 2} else {NDRE = 4}
UMTS	if ([BS] < 128) {NDRE = 2} else {NDRE = 4}

Note: Working with a number of DRE engines which exceed the limitations described in **Table 26-15** may result in an unknown state. Working with less DRE engines than described in **Table 26-15** results in fewer internally divided sub-blocks of the decoded block and may achieve slightly better BLER performance.

Activating less DRE engines for a given block size (with respect to the values in **Table 26-15**) should be done within the following limitations:

Table 26-16. Minimum Number of DRE Engines

Total Block Size[1]	Minimum number of DRE engines ([NDRE] value)
Total Block Size ≤ 1024	1 (00)
1024 < Total Block Size ≤ 2048	2 (01)
2048 < Total Block Size	4 (10)

1. The Total Block Size as defined in this table should include the 3 Tail bits (if they exist), that is, “Total Block Size” = [BS] + 3.

26.4.3.2.6.4.4 Turbo Stopping Criteria Configurations

The eTVPE offers the following 3 types of stopping criteria:

- Aposteriori Output Quality based Stopping Criteria
- CRC Check based Stopping Criteria
- Steady CRC based Stopping Criteria

Using any of the stopping criteria allow the eTVPE to stop the Turbo decoding before reaching the [MAX_ITER] values specified in the eTVPE BD. If stopping criteria is disabled, the eTVPE 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 eTVPE when any of the stop criteria is enabled.

Note: It is possible to activate two stopping criteria conditions, but it is forbidden to enable both the CRC-Check-based and the Steady-CRC-based stopping criteria in parallel.

26.4.3.2.6.4.5 Aposteriori Output Quality Based Stopping Criteria

In order for the eTVPE to work with the Aposteriori Quality (AQ) based stopping criteria, the following parameters/eTVPE registers must be configured:

- MTVCP[AQC_AUTOSTOP] - Aposteriori Quality Stop Criteria enable bit of the MTVCP This bit must be set during MAPLE-B2 initialization.
- eTVPE register: TVAQCR[AQTH] - Threshold value, to be compared to the Aposteriori results after each internal iteration.

For each iteration output bit, the eTVPE calculates the following to determine if Aposteriori Output quality has been reached:

```
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;
```

The stopping criteria is only checked starting from the iteration number defined in the [MIN_ITER] field, defined in the eTVPE BD, that is, The [MIN_ITER] field dictates the minimum number of iterations executed by the eTVPE regardless of the AQ Based Stopping criteria.

The [CMPLT_RSN] status field in the eTVPE BD, indicates whether the eTVPE completed its decoding process due to stopping criteria, or due to reaching the maximum iteration number given in the [MAX_ITER] field of the eTVPE BD.

The [ITER_CNT] status field in the eTVPE 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 reached its threshold before the number of iterations specified in the [MAX_ITER] has executed, the [ITER_CNT] field contains the actual number of iterations executed until reaching the stopping criteria threshold.

Note: The higher the number of the Aposteriori Threshold value ([AQTH]), the better the decoding quality is. Recommended range of values is between 100 to 150.

26.4.3.2.6.4.6 CRC Check Based Stopping Criteria

The CRC check based stopping criteria means that the eTVPE runs CRC check on its Hard Output results after each full iteration. If the CRC check has passed, the eTVPE stops its operation and notify MAPLE-B2 on job completion.

If the CRC check based stopping criteria is enabled and if the [MIN_ITER] field of the eTVPE BD, which sets the minimum number of iterations to be executed by the eTVPE, is enabled (greater than 0), then the CRC checks on the Hard Outputs are executed only when the minimum number of iterations is reached.

If the CRC check based stopping criteria is enabled and if the [MAX_ITER] field of the eTVPE BD, which sets the maximum number of iterations to be executed by the eTVPE, is reached without passing the CRC check, the eTVPE stop the decoding operation and notify MAPLE-B2 on job completion. In such case, the MAPLE-B2 give a CRC fail indication in the status field of the eTVPE BD ([ERROR_STATUS]).

Enabling the CRC check based stopping criteria is done during MAPLE-B2 initialization by setting the [CRC_AUTOSTOP] bit of the MTVCP parameter (see **Section 26.5.2.3**).

The CRC polynomial configuration for the CRC based stop criteria is determined by the [CRC_POLY] field of the eTVPE BD. The eTVPE supports the following CRC polynomials:

1. CRC24. Polynomial: $D^{24} + D^{23} + D^6 + D^5 + D + 1$ ([CRC_POLY] = ‘00’)
2. CRC24. Polynomial: $D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1$ ([CRC_POLY]=‘01’)
3. CRC16. Polynomial: $D^{16} + D^{12} + D^5 + 1$ ([CRC_POLY]=‘10’).

26.4.3.2.6.4.7 Steady CRC Based Stopping Criteria

The Steady CRC stopping criteria means that the eTVPE compares the hard output bits from previous iteration to the hard output bits from current iteration. If the hard outputs bits of the two iterations is identical, the eTVPE stops its operation and notify MAPLE-B2 on job completion. The comparison of the hard outputs of two successive iterations is performed by calculating the CRC result of the hard output bits of each iteration and comparing their results. The [CRC_POLY] field of the eTVPE BD determines which of the existing CRC polynomials is used for the check (For details on the existing CRC polynomials see: CRC Check Based Stopping Criteria).

If the Steady CRC based stopping criteria is enabled and if the [MIN_ITER] field of the eTVPE BD, which sets the minimum number of iterations to be executed by the eTVPE, is enabled (greater than 0), then the hard outputs compare checks are executed only when the minimum number of iterations is reached.

If the Steady CRC based stopping criteria is enabled and if the [MAX_ITER] field of the eTVPE BD, which sets the maximum number of iterations to be executed by the eTVPE, is reached without passing the hard outputs comparison check, the eTVPE stops the decoding operation and notifies MAPLE-B2 on job completion.

Enabling the Steady CRC based stopping criteria is done during MAPLE-B2 initialization by setting the [SCRC] bit of the MTVCP parameter (see **Section 26.5.2.3**, *MAPLE eTVPE Configuration Parameter (MTVCP)*).

Note: For UMTS jobs, both CRC Check and Steady CRC stopping criteria checks are allowed only with ascending byte/bit ordering configuration of the hard outputs (see **Section 26.4.3.2.7, *eTVPE Output Data Structure***"). Any other byte/bit ordering results in a CRC fail.

26.4.3.2.6.4.8 Stopping Criteria Configuration Limitations

When MAPLE-B2 is configured to work in 3G operation mode (supporting dynamic transitions between 3GLTE and UMTS jobs), there are some limitations as to the Stopping Criteria configurations since these configurations share the two technologies:

1. If CRC Stop Criteria is required for 3GLTE, it should be disabled during UMTS jobs by setting the [MIN_ITER] field of the eTVPE BD to the same value as the [MAX_ITER] field of the eTVPE BD.
2. If CRC Steady or Aposteriori Quality based Stop Criteria is set, it is applied on both technologies. Disabling it for one of the technologies can be done by setting the [MIN_ITER] field to the same value of the [MAX_ITER] field of the eTVPE BD.

26.4.3.2.6.4.9 Stopping Criteria Status Reports

To increase eTVPE performance and to avoid performance degradation due to the Stopping Criteria implementation, the eTVPE performs the Stopping Criteria checks of iteration #n while executing iteration #(n+1). If the Criteria checks pass, the eTVPE stops with the decoding processing at the end of the current iteration (#(n+1)).If hard outputs are required, the eTVPE outputs and hard bits results of iteration #n, but if any of the soft output results are required, the soft results which are used, are of iteration #(n+1), although the Stopping condition passed for iteration #n. In some extreme cases it may happen that the stopping condition, although passed on iteration #n, fails on the results of iteration #(n+1). The MAPLE-B2 reports these cases in the ERR_ST field of the eTVPE BD:

- [ERR_ST] = ‘0010’ indicates that the CRC check or CRC steady stop criteria failed on last iteration (#(n+1)) although passed on previous iteration (#n)
- [ERR_ST] = ‘0100’ indicates that the AQC (Aposteriori Quality Check) stop criteria failed on last iteration (#(n+1)) although passed on previous iteration (#n)

26.4.3.2.6.5 Viterbi Decoding

The MAPLE-B2 (eTVPE) supports Viterbi decoding for the different technologies using configurable polynomials and Viterbi related parameters that can be updated for each Viterbi job. Such implementation makes the Viterbi processing to a non operation mode related, that is, it can perform any Viterbi processing (of any type and any technology), regardless of the current MAPLE-B2 operation mode. The following sections describe different aspects of the Viterbi processing.

26.4.3.2.6.5.1 Viterbi Periodic Depuncturing

The Viterbi Periodic Depuncturing input data structure supports a periodically punctured input data stream, and the MAPLE-B2 is capable of performing periodic depuncturing of the input stream. The MAPLE-B2 uses the MTVPVx(H/L)CP and MTVPPCyP parameters (see **Section 26.5.3.2.1**, *MAPLE Turbo Viterbi Puncturing Vector x High Configuration Parameter (MTVPVxHCP)* to **Section 26.5.3.2.3**, *MAPLE Turbo Viterbi Puncturing Period Configuration y Parameter (MTVPPCyP)*), with the [PUNC_SCHM] field of the eTVPE BD to configure the depuncturing scheme of the input stream. The MAPLE-B2 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 eTVPE BD chooses the correct set for the current job.

Each puncturing set includes two elements:

- Puncturing vector. 63-bit vector that describes where in the input stream depuncturing 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 depuncturing algorithm:

```
j=0; /* depunctured 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 parameters, respectively.

Figure 26-23 shows an example Viterbi decoding periodic depuncturing of a rate 1/3 encoder.

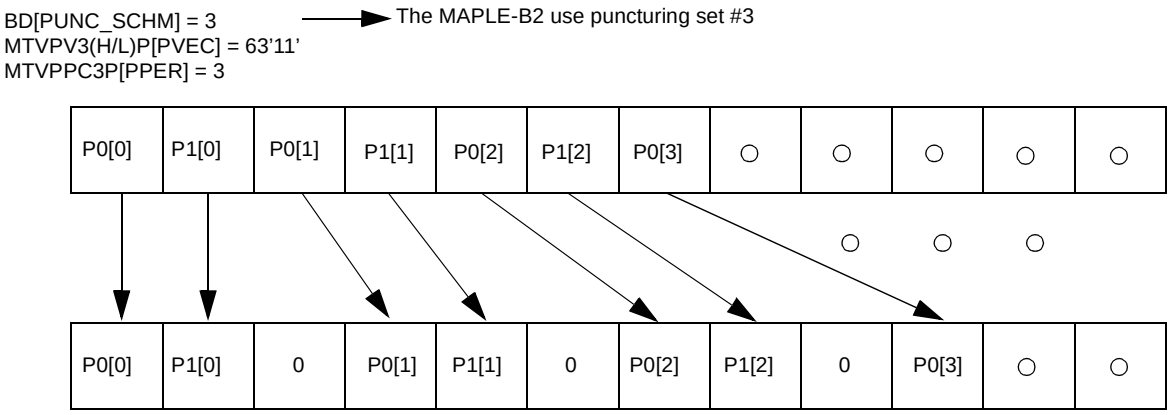


Figure 26-23. Viterbi depuncturing Process Example

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.

For Viterbi decoding, the Tail bytes are expected at the end of the input vector with the same order and with the same puncturing scheme as the body bytes. For example, for Viterbi rate 1/3, the expected order of the Tail bytes at the end of the vector then are:

$P0_t[0], P1_t[0], P2_t[0],..., P0_t[K-2], P1_t[K-2], P2_t[K-2].$

where K is the Viterbi Constraint length.

26.4.3.2.6.5.2 Viterbi Decoding Algorithm

The eTVPE 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[1]:
  - 6144 for K=5 and encoding rate of 1/2 and 1/3.
  - 4864 for K=5 and encoding rate of 1/4.
  - 4800 for K=6
  - 2400 for K=7
  - 1200 for K=8
  - 600 for K=9

1. The maximum block sizes include the tail bits in case of zero tail decoding.

The Viterbi processing in the eTVPE is not standard related, and is fully configurable (including polynomials) regardless of the MAPLE-B2 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, and trace back calculation. Explanations for each of the stages follows.

26.4.3.2.6.5.3 Feed Forward

During this stage, the eTVPE calculates 2^{k-1} path Metrics per trellis step, each with length n, where K is the Constraint Length and n is the block size.

Figure 26-24 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 $PM_{n,s}$.

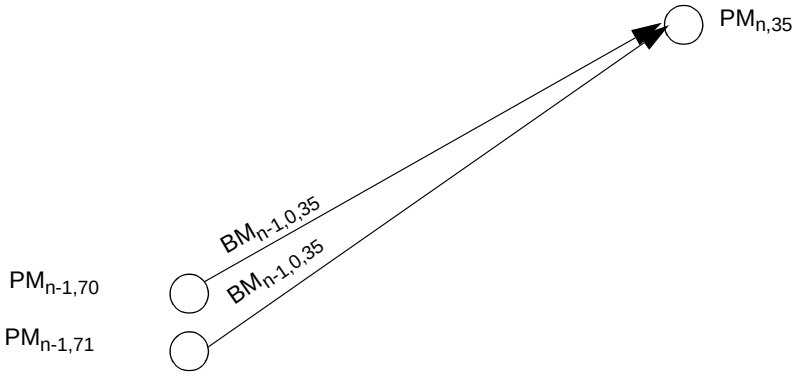


Figure 26-24. $PM_{n,35}$ Calculation

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 26-24**), 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.

$BMn,0,d =$

$$Parity_{0,n} * poly0_{<(d*2+0)\bmod num\ of\ pms>} +$$
$$Parity_{1,n} * poly1_{<(d*2+0)\bmod num\ of\ pms>} +$$
$$Parity_{2,n} * poly2_{<(d*2+0)\bmod num\ of\ pms>} +$$
$$Parity_{3,n} * poly3_{<(d*2+0)\bmod nuim\ of\ pms>}$$

$BMn,1,d =$

$$Parity_{0,n} * poly0_{<(d*2+1)\bmod num\ of\ pms>} +$$
$$Parity_{1,n} * poly1_{<(d*2+1)\bmod num\ of\ pms>} +$$
$$Parity_{2,n} * poly2_{<(d*2+1)\bmod num\ of\ pms>} +$$
$$Parity_{3,n} * poly3_{<(d*2+1)\bmod num\ of\ pms>}$$

where d is destination state, pms is path metric states and poly<x> is the Viterbi polynomials used.

To support dynamic switching of multiple Viterbi polynomial configurations, the MAPLE-B2 maintains three sets of polynomial configurations parameters which can be dynamically (per eTVPE BD) be replaced by the MAPLE-B2 firmware. The control of which set is being used for each eTVPE BD is done by the [VIT_SET] filed of the eTVPE BD. Configuring the [VIT_SET] to 01, 10 or 11 causes the MAPLE-B2 to use its MTVPVS1CxP, MTVPVS2CxP or MTVPVS3CxP parameters sets respectively for the current BD.

The expected parity bytes are the parities that are calculated on the source state using the parity polynomials described in **Figure 26-25**.

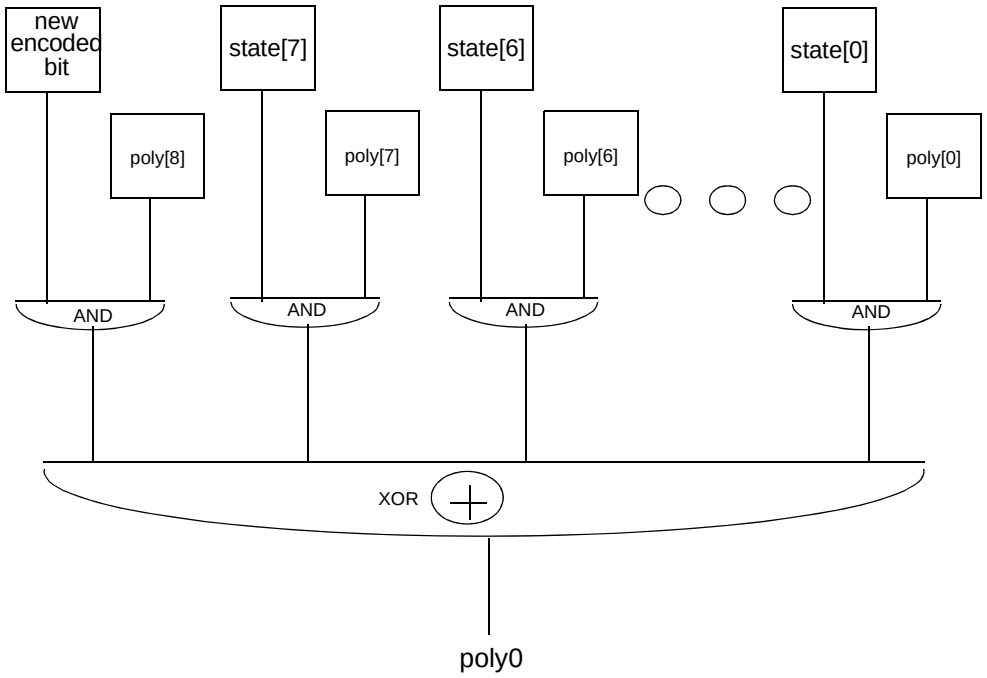


Figure 26-25. Calculation of a Parity of a Step

The Polynomial Generation fields described in any of the MTVPVSyCx Pares used for calculating the parity bits per state.

$polyz_{<x>}$ =

(new_encoded_bit & VI_POLYGEN‘z’[8]) ^

(x[7] & VI_POLYGEN‘z’[7]) ^

(x[6] & VI_POLYGEN‘z’[6]) ^

(x[5] & VI_POLYGEN‘z’[5]) ^

(x[4] & VI_POLYGEN‘z’[4]) ^

(x[3] & VI_POLYGEN‘z’[3]) ^

(x[2] & VI_POLYGEN‘z’[2]) ^

(x[1] & VI_POLYGEN‘z’[1]) ^

(x[0] & VI_POLYGEN‘z’[0])

where VI_POLYGEN is the field of MTVPVSyCx parameter and 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 does not work properly.

Note: The value of the VI_POLYGEN fields must not exceed the $2^k - 1$ value, where K is the Viterbi Constraint Length.

The Path Metric is calculated using compare-select between the addition of the source states and their respective branch matrices as follows:

$PM_{n,d} = compare\_2\_select\ ((PM_{n-1,(d*2+0)mod\ pms} + BM_{n,0,d}), (PM_{(n-1,(d*2+1)mod\ pms} + BM_{n,1,d}))$; *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 initial $PM_{<d>}$ field kept in the eTVPE before execution.

26.4.3.2.6.5.4 Maximum Calculations

The eTVPE 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 26-17**.

Table 26-17. 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

26.4.3.2.6.5.5 Trace-Back Calculations

The eTVPE, starting from a given trellis step traces back the winning path according to an internal history buffer. While tracing back the winning path, the eTVPE generates the Hard Output bits, which are then transferred to external memory.

26.4.3.2.6.6 Zero Tail Viterbi Processing

For Zero Tail Viterbi processing ([ALG]=1, [ZTTB]=1), the eTVPE 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 eTVPE BD, the eTVPE executes the following:

```
case ({[TBZE], MAX_STATE}) //MAX_STATE =1 if the result state from the maximum
calculations is non zero
{
        '00': Start Trace-Back calculation from state zero
        '01': Start Trace-Back calculation from the maximum state result
        '10': Start Trace-Back calculation from state zero
        '11': Start Trace-Back calculation from state zero and set [ZT_MAX]
}
```

5. If the [HOE] bit of the eTVPE BD is set, then Hard Outputs bits are generated during the trace back calculations and the MAPLE-B2 outputs them to system memory according to the pointers given in the BD.

26.4.3.2.6.7 Tail Biting Viterbi Processing (WAVA*)

For Tail Biting Viterbi processing ([ALG]=1, [ZTTB]=0), the eTVPE perform the WAVA* (Wrap-Around-Viterbi-Algorithm). The sequence of events executed by the eTVPE when implementing Viterbi Tail biting using WAVA* is:

1. Initialize all (path metrics) 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, that is, the PM initial values are the results of the first Feed Forward calculation.
4. Initiate additional Feed Forward calculations on the same data, but without initializing the PM values, that is, the PM initial values are the results of the first Feed Forward calculation.
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 which generates the Hard Outputs.

The following figure describes the WAVA* high level flow:

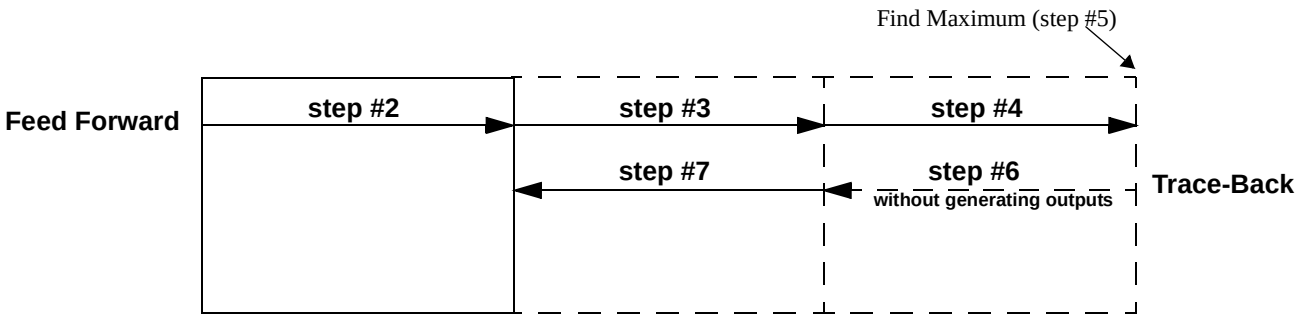


Figure 26-26. Viterbi Tail-Biting WAVA* flow

26.4.3.2.6.8 Viterbi Large Blocks Partitioning Support

The MAPLE-B2 allows Viterbi decoding of blocks with size of up to 32 KB 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 26.4.3.2.8.3.2, *Hard Output Byte and Bit Ordering***) supported for Viterbi large block sizes is either byte/bit ascending order or byte/bit descending order. The Hard Outputs data structure cannot use ascending byte order with 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 26-18:

Table 26-18. 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 26-18, 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 26-19:

Table 26-19. Viterbi Maximum Chunk Sizes During Large Block Partitioning Scheme

Viterbi K	Max Chunk for Partitioning
5	8192
6	5376
7	2688
8	1280
9	640

To maximize decoding quality, the host must partition the large block into overlapping chunks, thus enabling "warm-up" section for both, the Feed-Forward stage and the Trace-Back stage. The size of the "warm-up" section in each block can be configured by the host using the BUF_SIZE

field of the eTVPE BD and must be a multiplication of 64. **Figure 26-27** describes an example of 24Kbit (24576) viterbi code and its partitioning into several chunks in the case of K=5:.

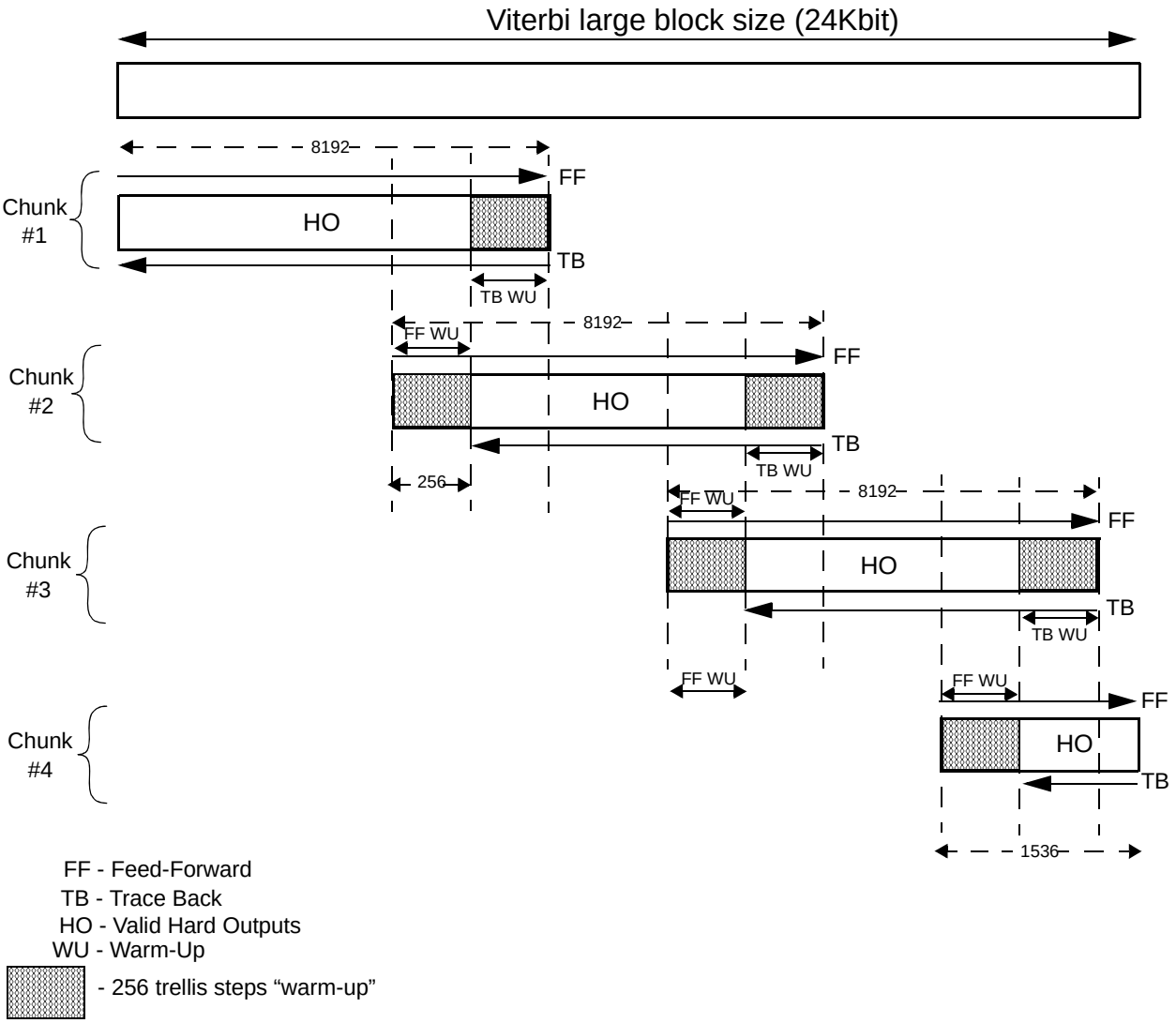


Figure 26-27. Viterbi Large Blocks Size Partitioning Example

For better understanding of the example, the following is assumed:

- 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 26-20 describes the relevant fields in eTVPE BD for each chunk as described in **Figure 26-27**:

Table 26-20. eTVPE 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
	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' 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
2	VIT_K	001	K = 5
	ZTTB	0	Zero Tail decoding
	ENC_RATE	00	Encoding rate: 1/2
	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-B2
	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

Table 26-20. eTVPE BD Fields of Viterbi Large Block Size Partitioned Into Chunks

Chunk #	Field name	Value	Comments
3	VIT_K	001	K = 5
	ZTTB	0	Zero Tail decoding
	ENC_RATE	00	Encoding rate: 1/2
	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-B2
	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
4	VIT_K	001	K = 5
	ZTTB	1	Zero Tail decoding
	ENC_RATE	00	Encoding rate: 1/2
	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-B2
	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

26.4.3.2.6.9 De-Randomization

The eTVPE can execute de-randomization processing for the hard outputs results from the Turbo decoding as appears in the WiMAX OFDMA turbo decoding (IEEE 802.16e-2009).

Configuring the eTVPE to perform de-randomization is allowed only during WiMAX operation mode and is done by configuring the [RANE] field of the MTVCP parameter.

Figure 26-28 describe the eTVPE de-randomizer implementation:

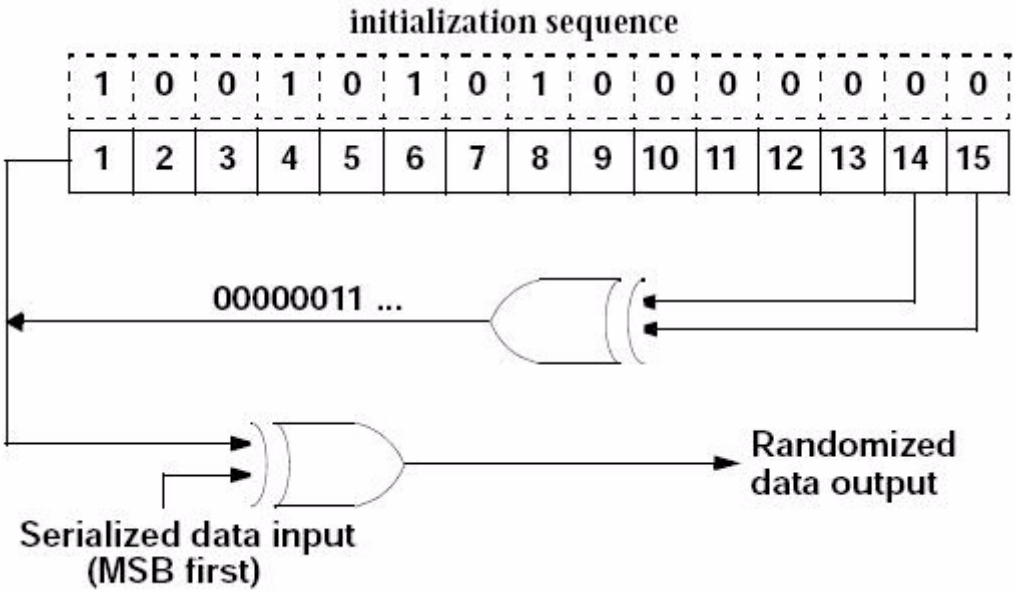


Figure 26-28. eTVPE De-Randomizer implementation

26.4.3.2.6.10 CRC Check

The eTVPE can execute a single CRC check on the Hard Outputs of the Turbo decoded blocks. The CRC check, which is executed internally by the eTVPE, supports 3 possible CRC polynomials determined by the [CRC_POLY] field of the eTVPE BD (as described in CRC Check Based Stopping Criteria).

To enable the CRC check for a given BD, the [CRC_EN] bit of the BD must be set. Once set, the eTVPE executes the CRC check on the Hard Output results, and only then indicates on job completion to MAPLE-B2. If the CRC check failed, the MAPLE-B2 gives a CRC fail indication in the [ERROR_STATUS] field of the eTVPE BD. If the CRC check passes no indication is given.

If CRC Check Based Stopping criteria is enabled (see CRC Check Based Stopping Criteria), the eTVPE executes a CRC check after every Turbo full iteration, regardless of the [CRC_EN] bit of the eTVPE BD.

Note: The eTVPE do not support TB CRC check for UMTS (for TB < 5090) as the UMTS technology dictates reversing the polynomial and it is not supported in the eTVPE.

26.4.3.2.7 eTVPE Output Data Structure

The eTVPE can generate few types of output data ordered internally according to two configuration bits. The following sections describe the output data types and output data structure.

26.4.3.2.8 Output Data Types

The eTVPE is capable of generating the following types of data:

- 16 bits of Extrinsic Output data (Systematic data only) - used for Debug purposes only.
- 16 bits LLR of Aposteriori Output data (Systematic data only)
- 8 bits LLR of Soft Output Data (Systematic + Parities)
- Hard Output bits

The Hard Output bits generation can be enabled regardless of the type of the required Soft Outputs.

26.4.3.2.8.1 Aposteriori/Extrinsic Output Data

Aposteriori/Extrinsic Output data generation is relevant only for Turbo decoding, and is enabled by enabling the [SOE] bits in the eTVPE BD. This option is relevant if additional processing (external to the MAPLE-B2) is needed on the data before calculating the final Hard Outputs, or for debug purposes.

The eTVPE can generate two types of such outputs:

- If [SOE] = 01, the eTVPE generates Aposteriori output.
- If [SOE] = 10, the eTVPE generates Extrinsic output

26.4.3.2.8.1.1 Aposteriori Output Data

The eTVPE generates the Aposteriori Outputs differently when executing Duo-Binary Turbo decoding (WiMAX) and Binary Turbo decoding (3GLTE, UMTS). **Table 26-21** describes the Aposteriori Output data for both cases.

Table 26-21. Aposteriori 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: $MLLR\_1_{xy}$ = Ln[APP_00/APP_01] $MLLR\_2_{xy}$ = Ln[APP_00/APP_10] $MLLR\_3_{xy}$ = 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 Aposteriori 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-B2 outputs the Aposteriori Output data to system memory in the location specified in the [SFT_RSLT_ADDR] field of the eTVPE BD. The structure of the Aposteriori Output results differs according to the decoding method (Binary for 3G technologies and Duo binary for WiMAX technology). **Figure 26-29** gives an example of a Aposteriori Output results buffer for the 3G technologies in a 128 bit wide system memory:

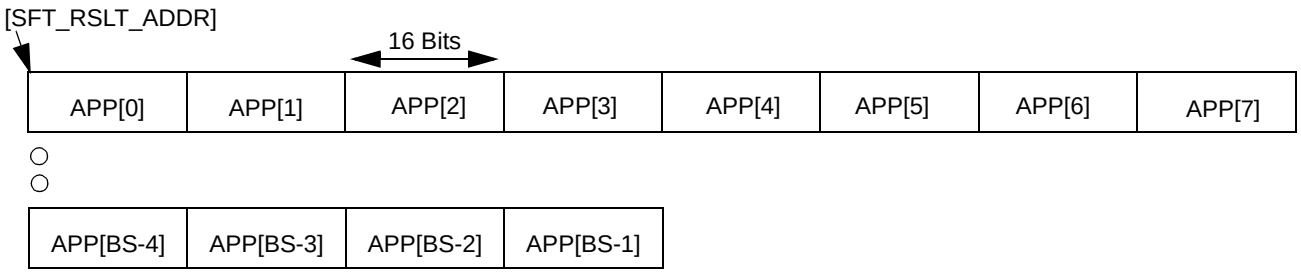


Figure 26-29. Example of Aposteriori Output Results Buffer for 3G Technologies in a 128-Bit System Memory

Figure 26-30 gives an example of a Aposteriori Output results buffer for the WiMAX technology in a 128 bit wide system memory:

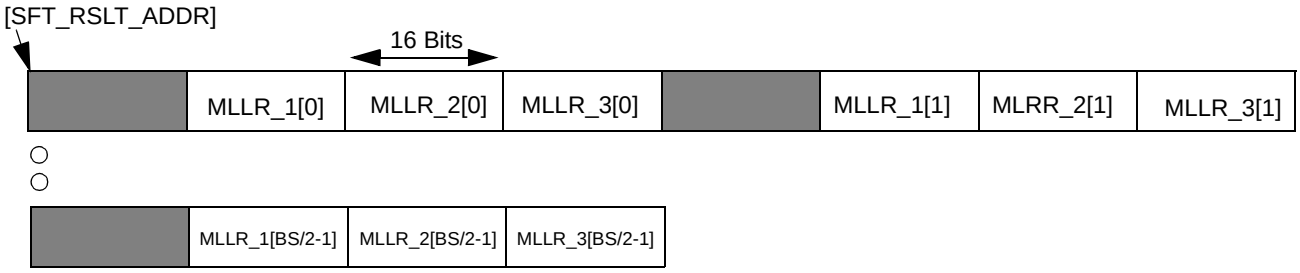


Figure 26-30. Example of Aposteriori Output Results Buffer for WiMAX technology in a 128 Bit System Memory

26.4.3.2.8.1.2 Extrinsic Output Data

The eTVPE 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 technology) and Binary Turbo decoding (3GLTE and UMTS technologies). **Table 26-22** describes the Extrinsic Output data for both cases.

Table 26-22. 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-B2 outputs the Extrinsic Output data to system memory in the location specified in the [SFT_RSLT_ADDR] field of the eTVPE BD. The structure of the extrinsic Output results differs according to the decoding method (Binary for 3G technologies and Duo binary for WiMAX technology). **Figure 26-31** gives an example of an extrinsic Output results buffer for the 3G technologies in a 128 bit wide system memory:

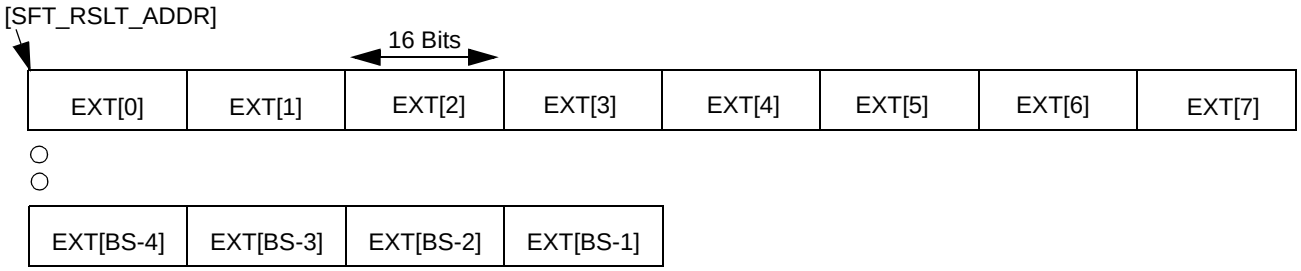


Figure 26-31. Example of Extrinsic Output Results Buffer for 3G Technologies in a 128-Bit System Memory

Figure 26-32 gives an example of a Extrinsic Output results buffer for the WiMAX technology in a 128 bit wide system memory:

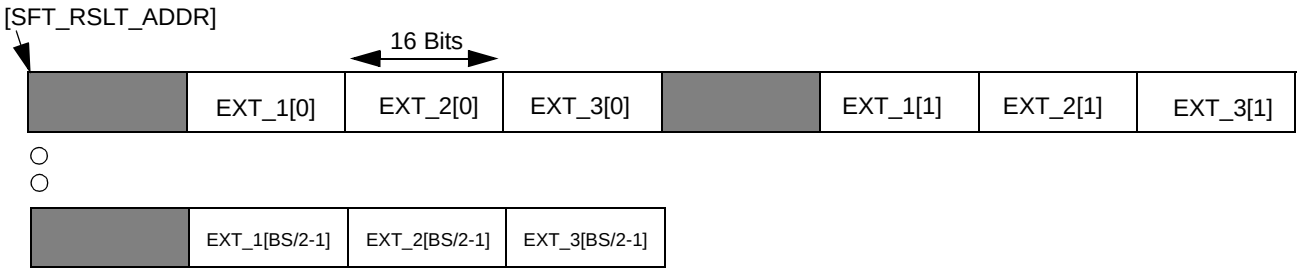


Figure 26-32. Example of Extrinsic Output Results Buffer for WiMAX technology in a 128-Bit System Memory

26.4.3.2.8.2 Soft Output Data

The Soft Output data generation is relevant only for Turbo decoding and only for the UMTS or 3GLTE technologies and is enabled by configuring the [SOE] bit of the eTVPE BD to 11. The Soft Output support is required to implement various iterative turbo aided receivers.

The eTVPE internal soft output data structure is composed of 16 bits LLR samples for each systematic and parity bit, but the MAPLE-B2 outputs only 8 bits of each LLR. Determining which 8 bits out the internal 16 bits are being output is done by the [LLR_OUT_SF] field of the eTVPE BD according to the following:

$$OUT\_LLR[7:0] = INTERNAL\_LLR[(7 + LLR\_OUT\_SF): (0 + LLR\_OUT\_SF)]$$

For example if *[LLR_OUT_SF]* = 3, then *OUT_LLR[7:0] = INTERNAL_LLR[10:3]*

When Soft Output data is required, the MAPLE-B2 outputs three vectors for each data type (SD, PF and PS). At the end of each vector there are 4 Tail bytes arranged as described in **Figure 26-33**:

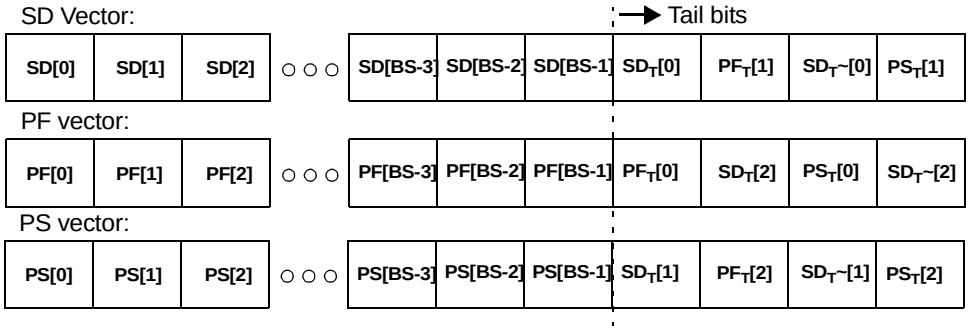


Figure 26-33. Soft Output Data Vectors

The MAPLE-B2 outputs the three Soft Output vectors to the system memory according to the following ruling:

- SD Vector is placed in [SFT_RSLT_ADDR]
- PF Vector is placed in [SFT_RSLT_ADDR] + 256 * 2^[SFT_RSLT_OFF]
- PS Vector is placed in [SFT_RSLT_ADDR] + 512 * 2^[SFT_RSLT_OFF]

For example, if [SFT_RSLT_ADDR] == 0x1000 and [SFT_RSLT_OFF] == 4, then:

- SD Vector is placed in 0x1000
- PF Vector is placed in 0x2000
- PS Vector is placed in 0x3000

Note: Overrun of one vector by the other may occur if offset between vectors is smaller than vector size.

Note: When working with Soft Output Data (*SOE*=11), the following must apply: *(MIN_ITER* $\geq$ 2)

26.4.3.2.8.3 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 eTVPE BD.

The Hard Outputs for the Turbo decoding are generated from the Aposteriori data as follows:

- For Binary decoding (3GLTE, UMTS):
  - 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-B2 outputs the Hard Output data to system memory in the location specified in the [HRD_RSLT_ADDR] field of the eTVPE BD.

26.4.3.2.8.3.1 Hard Output Offset

The MAPLE-B2 allows outputting the hard outputs with a bit resolution address. Configuring the [HOFF] field of the eTVPE BD determines the location of the first valid bit in the first byte of the hard outputs. Along with the byte resolution of the [HRD_RSLT_ADDR] pointer, the host is capable of determining the exact bit location in the system memory of the first valid bit. Supporting bit resolution of the hard bits output enabled concatenating hard outputs of all CBs related to a single TB thus eliminating the need to concatenate them by the host.

The following figure describe an example of concatenating 3 CB into single TB:

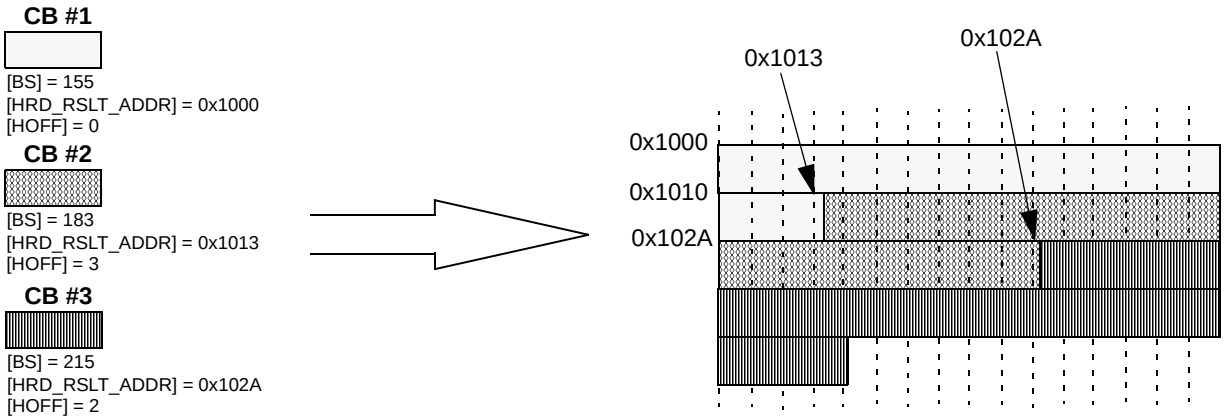


Figure 26-34. Concatenating Hard outputs

Note: For Viterbi processing the Hard Output Offset is applicable only if the DOSBY/DOSBI configuration bits of the eTVPE Configuration 0 Register (TVPEC0R) are set.

26.4.3.2.8.3.2 Hard Output Byte and Bit Ordering

The Hard Outputs byte and bit order of the eTVPE can be programmed by setting the TVPEC0R[DOSBY] and the TVPEC0R[DOSBI] to one of the following configurations:

- DOSBY, DOSBI = 0b11. Ascending Byte and Bit ordering. The eTVPE hard Outputs are ordered in ascending byte and ascending bit order as illustrated in the example in **Figure 26-35**.

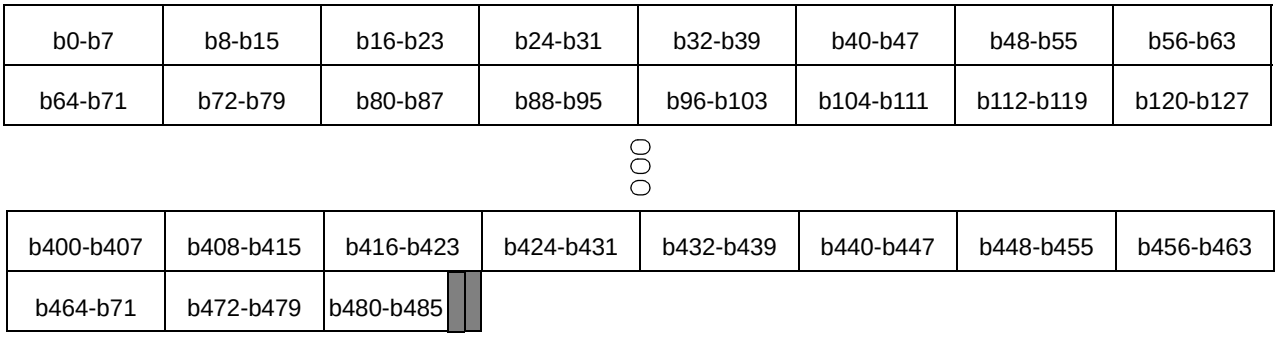


Figure 26-35. Ascending Byte and Bit Ordering example (DOSBY=1, DOSBI=1)

- DOSBY, DOSBI = 0b10. Ascending Byte and descending Bit ordering. The eTVPE hard Outputs are ordered in ascending byte order and descending bit order as illustrated in the example in **Figure 26-36**.

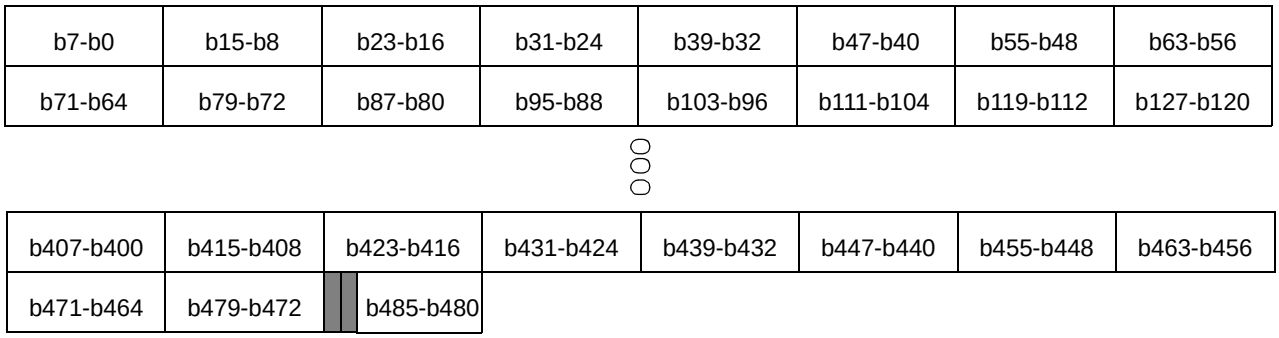


Figure 26-36. Ascending Byte and Descending Bit Ordering Example (DOSBY=1, DOSBI=0)

- DOSBY, DOSBI = 0b01. Descending Byte and ascending Bit ordering. The eTVPE hard Outputs are ordered in Descending byte order and ascending bit order as illustrated in the example in **Figure 26-37**.

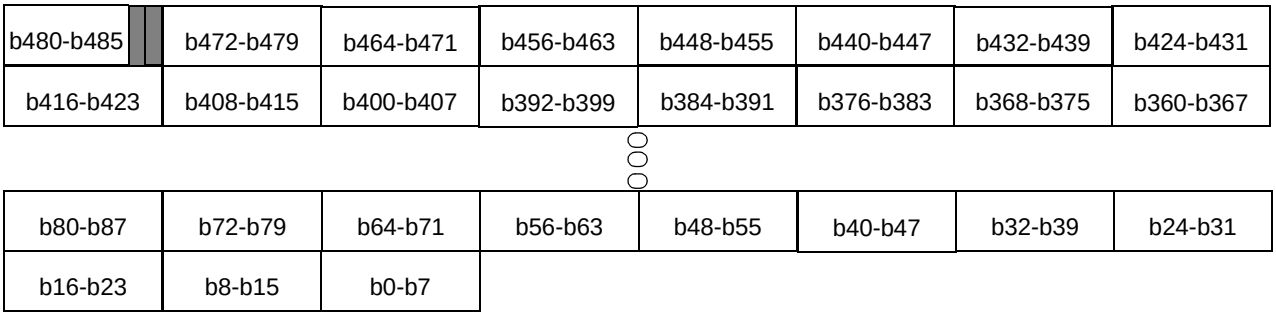


Figure 26-37. Descending Byte and Ascending Bit Ordering Example (DOSBY=0, DOSBI=1)

- DOSBY, DOSBI = 0b00. Descending Byte and descending Bit ordering. The eTVPE hard Outputs are ordered in Descending byte order and descending bit order as illustrated in the example in **Figure 26-38**.

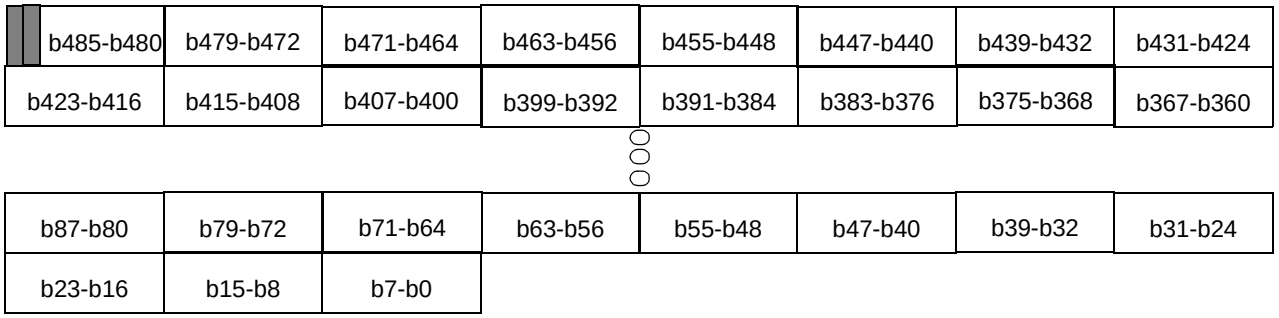


Figure 26-38. Descending Byte and Descending Bit Ordering Example (DOSBY=0, DOSBI=0)

Note: The **default** value of MAPLE-B2 is ascending byte and ascending bit ordering

26.4.3.2.8.3.3 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 eTVPE BDs in MAPLE-B2. It is recommended to configure this register only once (after MAPLE-B2 activation), and leave it fixed for the whole operation duration.
- For UMTS jobs, all CRC calculations (CRC check based stopping criteria, CRC Steady based stopping criteria and CRC check on Hard Outputs) assume the default Byte/Bit ordering only (ascending byte/bit ordering). Working with different configuration results in incorrect CRC results.

26.4.3.2.9 eTVPE Debug

The eTVPE includes some debug capabilities which allow a host to view its intermediate results. By allowing the eTVPE to output its internal extrinsic/Aposteriori results (using the [SOE] field in the eTVPE BD), and by allowing it to limit the number of executed iterations (using the [MAX_ITER] field of the eTVPE BD), it is possible to generate a flow by which the eTVPE outputs its extrinsic/Aposteriori results after every iteration.

For example, configuring the [SOE] field to 10 and the MAX_ITER to 0001 results in the eTVPE executing two iterations only and on completion the MAPLE-B2 output the extrinsic results from the eTVPE internal memories to the system memory to the address described in the [SFT_RSLT_ADDR] field of the eTVPE BD. If the extrinsic results of the third iteration are also of interest, it can be done by configuring the MAPLE-B2 with an additional BD, identical to the previous one, excluding the [MAX_ITER] field which should now be configured to 0010 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.

26.4.3.3 eFTPE FFT/iFFT/DFT/iDFT Operation

The MAPLE-B2 supports FFT/iFFT/DFT/iDFT operations in variable block sizes as listed in **Section 26.2**, *MAPLE-B2 Features* using three identical eFTPE modules instantiated in MAPLE-B2. These three eFTPE modules share the same programming model, the same operation flow, and, therefore, the same operational description.

Note: The FFT/iFFT/DFT/iDFT processing executed by the eFTPE modules is not standard related and, therefore, is not affected by the MAPLE-B2 operation mode.

Note: In this section, an FFT/iFFT job refers to a job with a Transform Length indicating one of the following sizes: 128, 256, 512, 1024, 1536 or 2048. All other sizes are referred to as DFT/iDFT jobs.

26.4.3.3.1 eFTPE Buffer Descriptors

eFTPE buffer descriptors are described in detail in **Section 26.5.4.3**, *eFTPE Buffer Descriptor Structure*, on page 26-423. eFTPE buffer descriptor extensions are described in detail in **Section 26.5.4.3.2**, *eFTPE Buffer Descriptor's Extension*, on page 26-433.

26.4.3.3.2 BD Repeat Option

The MAPLE-B2 allows the user to generate a single eFTPE Buffer Descriptor to describe up to 128 different jobs 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 26.4.3.3.9.4**, *eFTPE Internal Scaling Calculations*)
   - *OVA_SCL* - Overall Scaling (enabled or disabled. See **Section 26.4.3.3.9.4**, *eFTPE Internal Scaling Calculations*)
   - *ITE* - Inverse Transform Enable. See **Section 26.4.3.3.9.2**, *Inverse Transform Processing*
   - *TL_ID* - Transform Length ID. See **Section 26.5.4.3.3**, *Transform Length Encoding*
   - *USR_SCLx* - User Defined Scaling for stage 0 to stage 5. See **Section 26.4.3.3.9.4**, *eFTPE Internal Scaling Calculations*
   - *IN_SCL* - Input Scaling. See **Section 26.4.3.3.9.4**, *eFTPE Internal Scaling Calculations*
   - *ADP_OVA_SCL* - Adaptive Overall Scaling. See **Section 26.4.3.3.9.4**, *eFTPE Internal Scaling Calculations*
   - *GI* - Guard Insertion. See **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT*
   - *PME* - Pre-Multiply Enable. See **Section 26.4.3.3.5.1**, *Pre-Multiplication Processing Support in the eFTPE*
   - *PSMTE* - Post Multiplier Enable. See **Section 26.4.3.3.6**, *Scalar Post Multiplication in eFTPE*
   - *GR* - Guard Removal. See **Section 26.4.3.3.4.2.1**, *Guard Band Removal for FFT*
   - *CPIE* - Cyclic Prefix Enable. See **Section 26.4.3.3.4.2.2**, *Cyclic Prefix Insertion for iFFT*

- *PSTMV* - Post multiply vector. See **Section 26.4.3.3.5.2**, *Post-Multiplication processing support in the eFTPE*
- *PO_SCL* - Post multiplier scaling. See **Section 26.4.3.3.9.4.2**, *User Defined Scaling*
- *AIUS* - Adaptive input scaling. See **Section 26.4.3.3.9.4.3**, *Adaptive Scaling*
- *EXS* - Extra scaling. See **Section 26.4.3.3.9.4.3**, *Adaptive Scaling* and **Section 26.4.3.3.3**, *Identical Output Scale Alignment for BD Repeat*
- All frequency correction related fields. See **Section 26.4.3.3.7**, *Frequency Correction Support in eFTPE*
- *LSS* See **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*
- *SSN* See **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*
- *CL* See **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*
- *NNPI* See **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*

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 26-41**.
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-B2 starts the BD execution, it sequentially executes all the jobs. The MAPLE-B2 writes the output data from the jobs sequentially into system memory, starting at the address described in the *OBA* (Output Buffer Address) field of the eFTPE BD.

The *OWNER* bit is cleared by the MAPLE-B2 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-B2 uses the *BD_STAT_PTR* field of the eFTPE BD as a pointer into system memory where it copies the two status fields of each job (*CMP_RSN*[2:0] and *ADP_OVA_SCL_ST*[7:0]).

Figure 26-39 describes the data structure of the status fields in system memory:

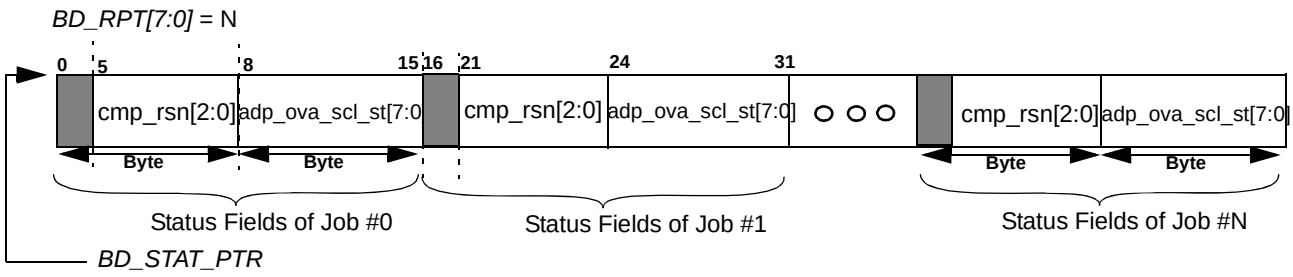


Figure 26-39. Status Table Data Structure in System Memory in Case of BD Repeat

For details, on the *ADP_OVA_SCL_ST* field see **Section 26.4.3.3.9.4**, *eFTPE Internal Scaling Calculations*.

For details, on the CMP_RSN field see **Table 26-230**

Note: The repeat option is highly recommended for use since it reduces the overhead of finding and parsing the following BD, thus increasing MAPLE-B2 performance.

Note: It is possible to disable the status fields update to the system memory by clearing the *BD_STAT_PTR* field in the BD (*BD_STAT_PTR*=0x0). Doing so may slightly improve the throughput performance of the eFTPE.

Note: The *BD_STAT_PTR* field is relevant for non repeat jobs as well. If enabled (*BD_STAT_PTR* > 0), the MAPLE-B2 outputs the status fields regardless of the repeat option.

26.4.3.3.3 Identical Output Scale Alignment for BD Repeat

The MAPLE-B2 supports the option of forcing identical Output Scale value for all the tasks included in the BD (in case the BD Repeat is enabled) without the need to force User Defined Scaling on the BD (see **Section 26.4.3.3.9.4.2**). Enabling this option is done by setting the *AFS* bit of the eFTPE BD. To support such option, the following flow is executed internally for all the tasks in the BD:

1. The First task is configured to run in Adaptive Scale mode (**Section 26.4.3.3.9.4.3**) and with Adaptive Overall Scaling Disabled (**Section 26.4.3.3.9.4.4**).
2. Once this first task is completed, the Adaptive Overall Scaling Status indication is being read.
3. The following tasks in the BD are configured to run in Adaptive Scale mode and with Adaptive Overall Scaling enabled where the Adaptive Overall Scaling value used is the Adaptive Overall Scaling Status indication of the first task in the BD.

The result of the above flow is that the output results of all the tasks in the BD shares the same Adaptive Overall Scaling results.

If the Extra Scaling (*EXS* field of the eFTPE BD) is enabled (see **Section 26.4.3.3.9.4.8**, *Extra Scaling*) in case of BD Repeat, the MAPLE-B2 applies the extra down scaling for the first task only. The following tasks of the BD Repeat are configured with *EXT*=0. This allows reducing the risk of multiple saturated samples in all the tasks which use the Adaptive Overall Scaling value result of the first task in the BD Repeat.

26.4.3.3.4 eFTPE Data Structures

The input data to the eFTPE is fetched by the MAPLE-B2 internal DMA. The location of the input data in system memory should be specified in the *IBA*[31:3] field of the eFTPE BD. Upon job completion, the output data of the eFTPE is transferred to system memory by the MAPLE-B2 internal DMA. The address to transfer the data is specified in the *OBA*[31:3] field of the eFTPE BD. The following sections describe the MAPLE-B2 expected input and output data structures in system memory.

26.4.3.3.4.1 Input Data Structures

After parsing the eFTPE BD, MAPLE-B2 starts fetching the input data into its internal memories for the processing operation. The eFTPE supports two types of input data structure:

1. Each input sample is 32 bits: 16 bits real component following by 16 bits imaginary component. **Table 26-23** describe a 128 bits bus structure for this case.
2. Each input sample is 16 bits: 8 bits real component following by 8 bits imaginary component. **Table 26-24** describe a 128 bits bus structure for this case.

Table 26-23. Four 32 Bits Samples As They are Expected to Arrive From System Memory

Bits	127:112		111:96		95:80		79:64		63:48		47:32		31:16		15:0	
Bytes	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Samples	Sample k				Sample k+1				Sample k+2				Sample k+3			
Value	Real		Imag.		Real		Imag.		Real		Imag.		Real		Imag.	

Table 26-24. Eight 16 Bits Samples As They are Expected to Arrive From System Memory

Bits	127:120	119:112	111:104	103:96	95:88	87:80	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
Bytes	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Samples	Sample k		Sample k+1		Sample k+2		Sample k+3		Sample k+4		Sample k+5		Sample k+6		Sample k+7	
Value	Real	Imag.	Real	Imag.	Real	Imag.	Real	Imag.	Real	Imag.	Real	Imag.	Real	Imag.	Real	Imag.

Enabling any of the above two input data structure options is done by the *SINS* field of the eFTPE BD according to the following:

— *SINS*=0: Input sample structure is 32 bits (16I, 16Q)
— *SINS*=1: Input sample structure is 16 bits (8I, 8Q)

Figure 26-40 shows an example of a DFT block of Transform Length 12, because it should be placed in a 128-bit memory structure and for 32 bits sample structure (*SINS*=0). The base address of the example buffer is 0x18:

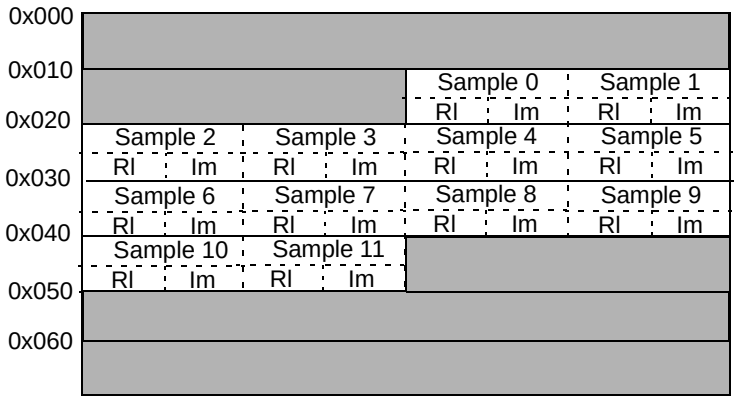


Figure 26-40. DFT Block, Transform Length 12, Start Address 0x18, *SINS*=0

Note: The Input Buffer Address in the external memory must be aligned to a 64 bit address, as shown in the example in **Figure 26-40**.

If the *BD_RPT* field in the eFTPE BD is enabled (*BD_RPT* > 0), then the Input Buffer Address should point to the location in system memory, where all the tasks related to that BD are located successively. **Figure 26-41** illustrates an example of how the data should be arranged in a 128 bit wide system memory if 5 tasks of Transform Length 24 are assigned to one BD:

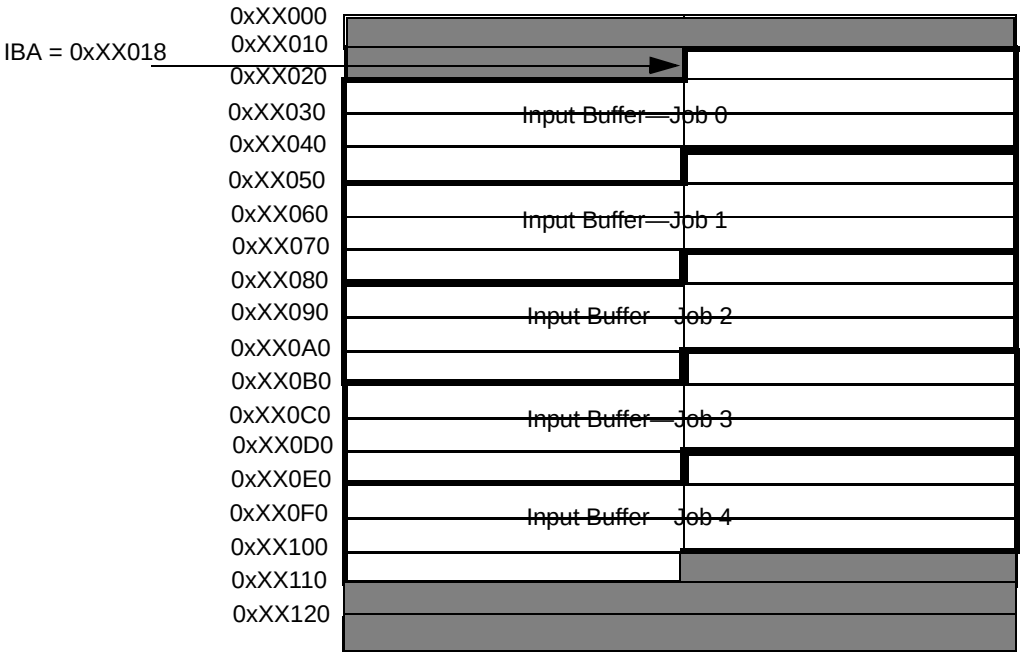


Figure 26-41. Input Buffer Structure for *BD_RPT*= 4, Transform Length = 24 and *SINS*=1

26.4.3.3.4.1.1 Guard Band Insertion for iFFT

To reduce system loading, the eFTPE supports a special input data structure for iFFT processing. This special input data structure allows the MAPLE-B2 to fetch only the relevant parts of the input data, and the eFTPE can internally generate the whole input data block structure.

Figure 26-42 is an example of the internal structure in the frequency domain of a 1024 samples Transform to be input for iFFT processing:

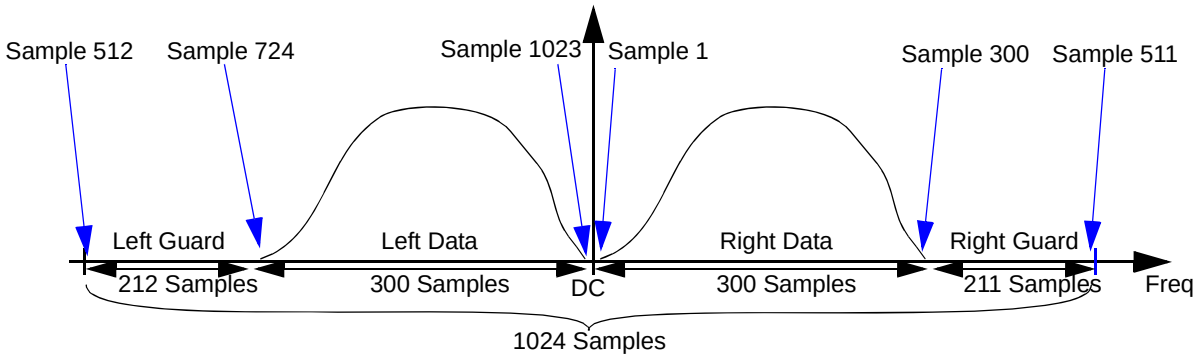


Figure 26-42. 1024 Samples Frequency Domain Example

As seen in **Figure 26-42**, 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 Guard Insertion bit (*GI* bit in the eFTPE BD) is enabled, the MAPLE-B2 fetches only the Left-data following by the Right-data from system memory, and the eFTPE internally generates the whole symbol structure as seen in **Figure 26-43**.

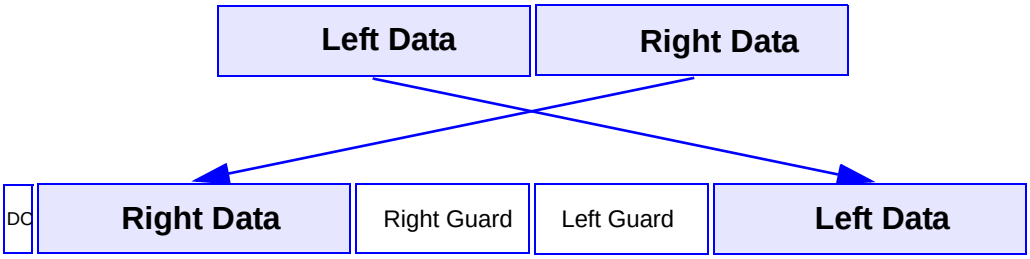


Figure 26-43. Generating Whole iFFT Data Structure Using the Right/Left Data

The size of the Right/Left data for each of the possible transform lengths are described in the eFTPE FTPEDSRx registers (**Section 26.5.3.3.1** to **Section 26.5.3.3.3**). **Table 26-25** describes

which fields in these registers relate to which Transform Length, and also the reset value for each of the fields.

Table 26-25. FTPEDSRx Fields and the Related Transform Length

Register [Field]	Transform Length	Reset Value for 3GLTE
FTPEDSR0[DS0]	128	36
FTPEDSR0[DS1]	256	90
FTPEDSR1[DS2]	512	150
FTPEDSR1[DS3]	1024	300
FTPEDSR2[DS4]	1536	450
FTPEDSR2[DS5]	2048	600

Updating the values in these registers can be done by one of the following options:

1. Host writing to these registers once during initialization. For each eFTPE BD with *GI* bit set, the value of the Data Size Set (*DSS)* field is ‘000’ indicating the MAPLE-B2 that the registers values were initialized once and require no update. Working this way requires no initialization of the FTPEDSSxPy parameter sets (see **Section 26.5.3.3.1**, *eFTPE Data Size Set x Parameter 0 (FTPEDSSxP0)* to **Section 26.5.3.3.3**, *eFTPE Data Size Set x Parameter 2(FTPEDSSxP2)*).
2. Host initialize the FTPEDSSxPy sets during initialization (see **Section 26.5.3.3.1**, *eFTPE Data Size Set x Parameter 0 (FTPEDSSxP0)* to **Section 26.5.3.3.3**, *eFTPE Data Size Set x Parameter 2(FTPEDSSxP2)*) and for each BD, set the *DSS* field of the eFTPE BD according to the required configuration parameter set. Once *DSS* field is set to any value between ‘001’ to ‘111’, MAPLE-B2 copy the relevant parameter set into the FTPEDSRx[DSy] registers (see **Section 26.5.5.3.1**, *EFTPE_<x> Data Size Register 0 (FTPE<x>DSR0)* to **Section 26.5.5.3.3**, *EFTPE_<x> Data Size Register 2 (FTPE<x>DSR2)*) in the relevant eFTPE module.

NOTE

- The *GI* bit can be used only while processing an iFFT job. It must be cleared for any other operation.
- Guard band insertion can’t be enabled together with Zero Padding process enabled by *ZP* bit in the same BD.
- Guard band insertion can’t be enabled when source of samples is internal code generation, enabled by the *CG* bit of the eFTPE BD.

The eFTPE 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 - 2 * FTPEDSRx[DSy]

where the *x* and *y* of the FTPEDSR*x*[DS*y*] are derived from the Transform Length value and according to **Table 26-25**.

26.4.3.3.4.1.2 Zero Padding Support

To reduce system loading, the eFTPE supports zero padding of the input data. This feature allows the MAPLE-B2 to input only the relevant part of the data while the eFTPE internally pads the input data to reach the required transform Length. To enable the Zero Padding feature, configure the following eFTPE BD parameters:

- *ZP* = 1: Zero padding Indication. Indicates the eFTPE that Zero Padding is required for the current job.
- *DSTZP* = <Size of data to be padded>: Indicates the Actual data size to be fetched by the MAPLE-B2 into the eFTPE and to be internally padded to reach the required Transform Length.

The eFTPE use the *DSTZP* field to calculate the required Zero Padding according to the following equation:

Required Zero Padding = (Transform Length) - (*DSTZP*)

The Zero Padding is added at the end of the input data. **Figure 26-44** describe an example of the input buffer of the eFTPE after the Zero Padding completion:

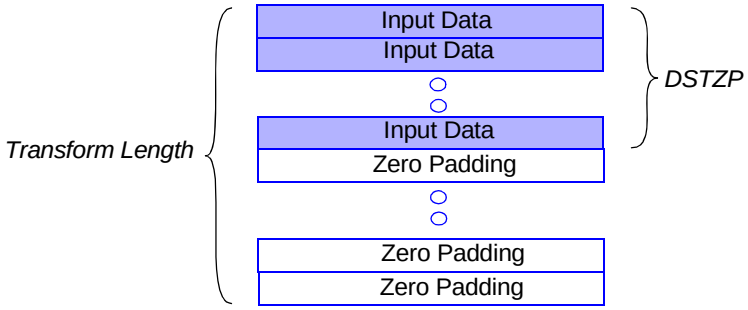


Figure 26-44. eFTPE internal Input Buffer description after Zero Padding

Note: The size of the Zero Padding is limited to numbers which are product of 4 samples and it must be smaller than the Transform Length (Transform Length > *DSTZP*).

26.4.3.3.4.1.3 Cyclic Prefix Removal for FFT BD Repeat

To reduce required BD configurations of the MAPLE-B2, and to improve usage of BD Repeat option (see **Section 26.4.3.3.2**, *BD Repeat Option*), the MAPLE-B2 supports removing of the Cyclic Prefix (CP) addition from the FFT inputs received from the Antenna Interface. **Figure 26-45** describe an example of the assumed input data received from the Antenna. By enabling the *CPRE* bit and by setting the *CPS* field to the cyclic prefix size in the eFTPE BD while BD repeat

is enabled, the MAPLE-B2 skips the cyclic prefix and fetches only the FFT inputs as described in the example in **Figure 26-45**

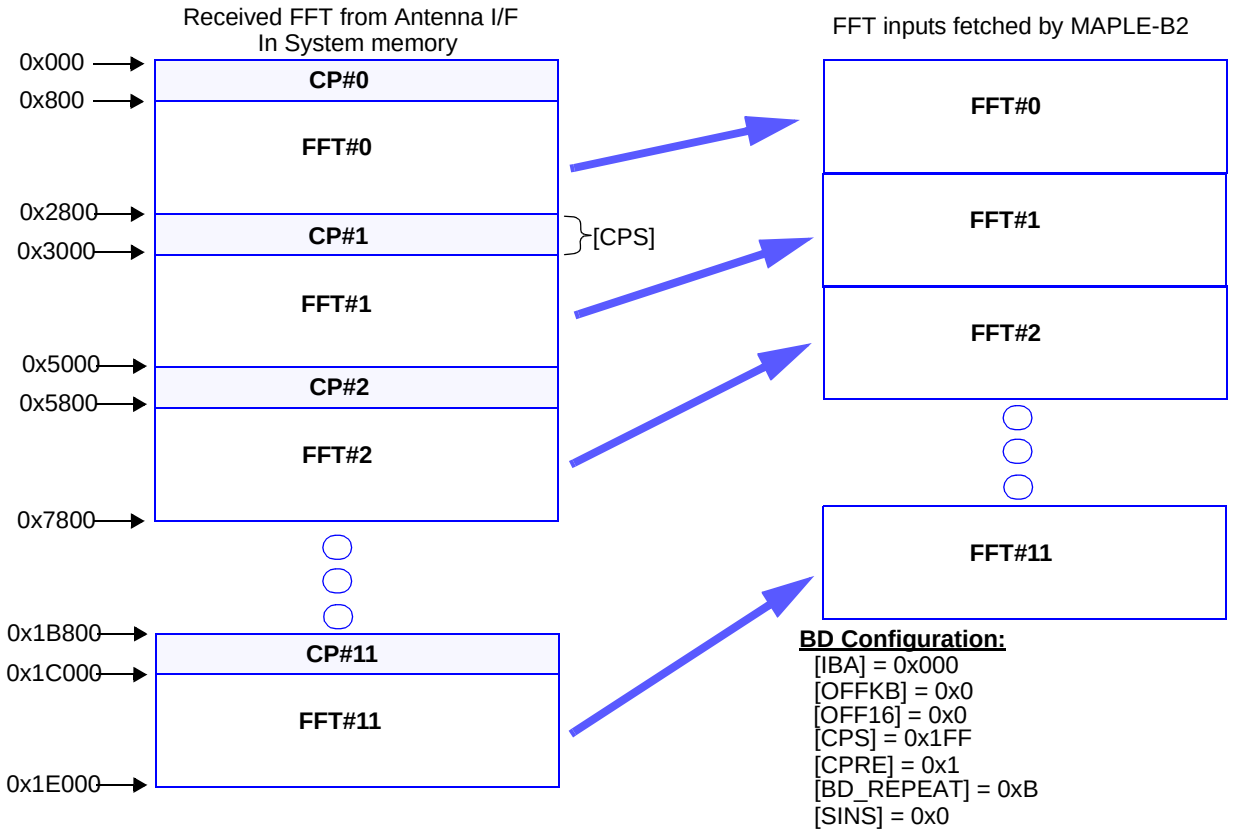


Figure 26-45. Cyclic Prefix Removal During BD Repeat

As can be seen in **Figure 26-45**, if the Cyclic Prefix Removal Enable bit (CPRE) in the eFTPE BD is set during BD Repeat, the MAPLE-B2 jump over the cyclic prefix while fetching the next FFT job. The size of the cyclic prefix should be set in the CPS field of the BD.

Note: In this configuration the *CPS* must not exceed the Transform Length.

26.4.3.3.4.1.4 Input Buffers Offset (in KBs) for FFT BD Repeat

To reduce required BD configurations of the MAPLE-B2, and to improve usage of BD Repeat option (see **Section 26.4.3.3.2**, *BD Repeat Option*), the MAPLE-B2 supports fixed offset between the input buffers required for the BD repeat option during FFT operation. The target of this feature is to allow using the BD repeat option for multiple receive antennas buffers located in fixed offsets in the system memory. **Figure 26-46** describe an example of the assumed input data received from N+1 Antennas. By enabling the *OFFKB* bit and by setting the *CPS* field to the offset size (in KBytes) in the eFTPE BD while BD repeat is enabled, the MAPLE-B2 fetches the FFT inputs as described in the example in **Figure 26-46**

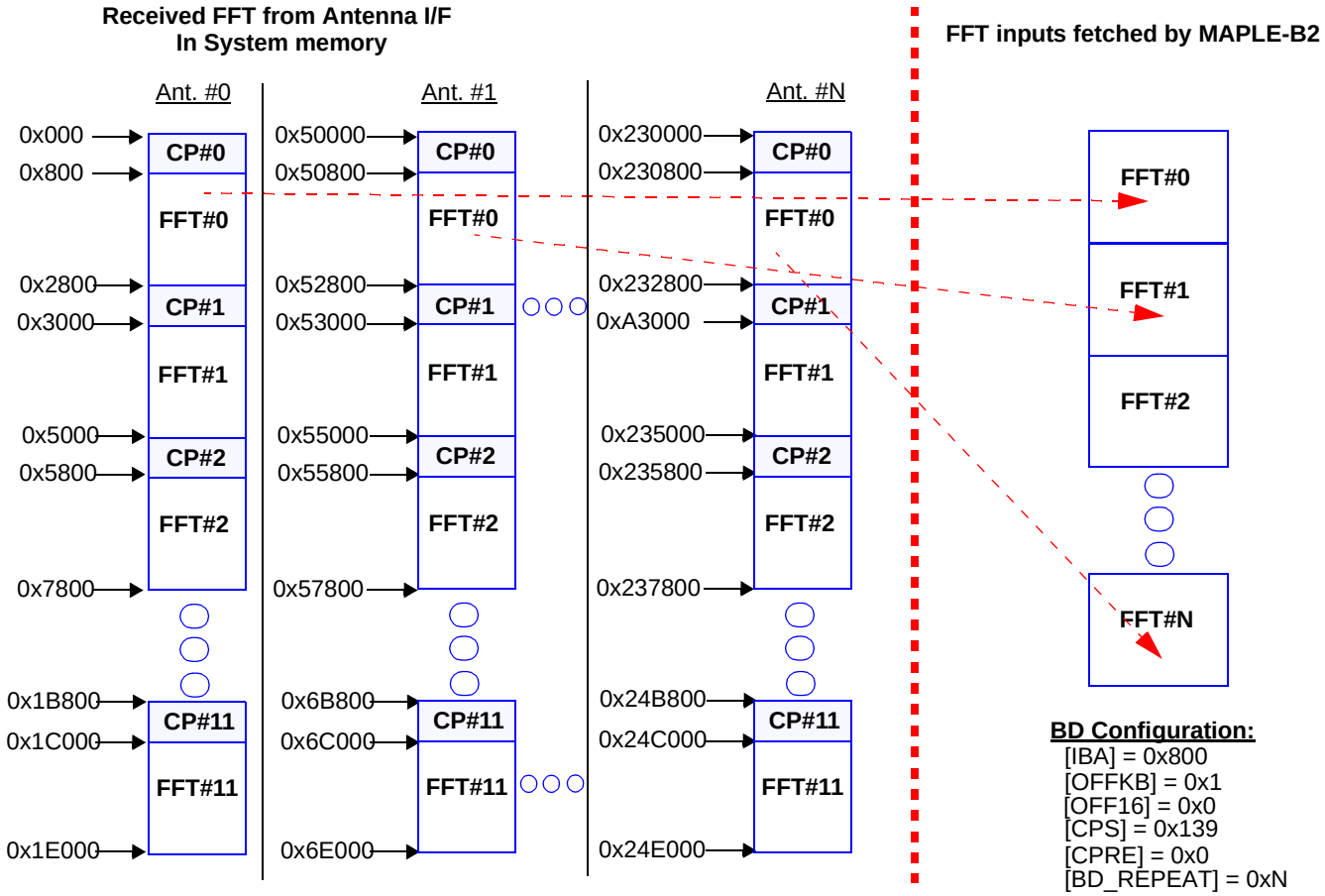


Figure 26-46. Input Buffers Offset (in KBs) During BD Repeat

26.4.3.3.4.1.5 Input Buffers Offset (in 16 Bytes multiplication) for DFT/iDFT BD Repeat

To reduce required BD configurations of the MAPLE-B2, and to improve usage of BD Repeat option (see **Section 26.4.3.3.2**, *BD Repeat Option*), the MAPLE-B2 supports fixed offset between the input buffers required for the BD repeat option during DFT/iDFT operation. **Figure 26-47** describe an example of using the 16 bytes multiplication offset usage. By enabling the *OFF16* bit and by setting the *CPS* field to the offset size (in multiplications of 16 bytes) in the eFTPE BD while BD repeat is enabled, the MAPLE-B2 fetches the DFT/iDFT inputs as described in the example in **Figure 26-47**

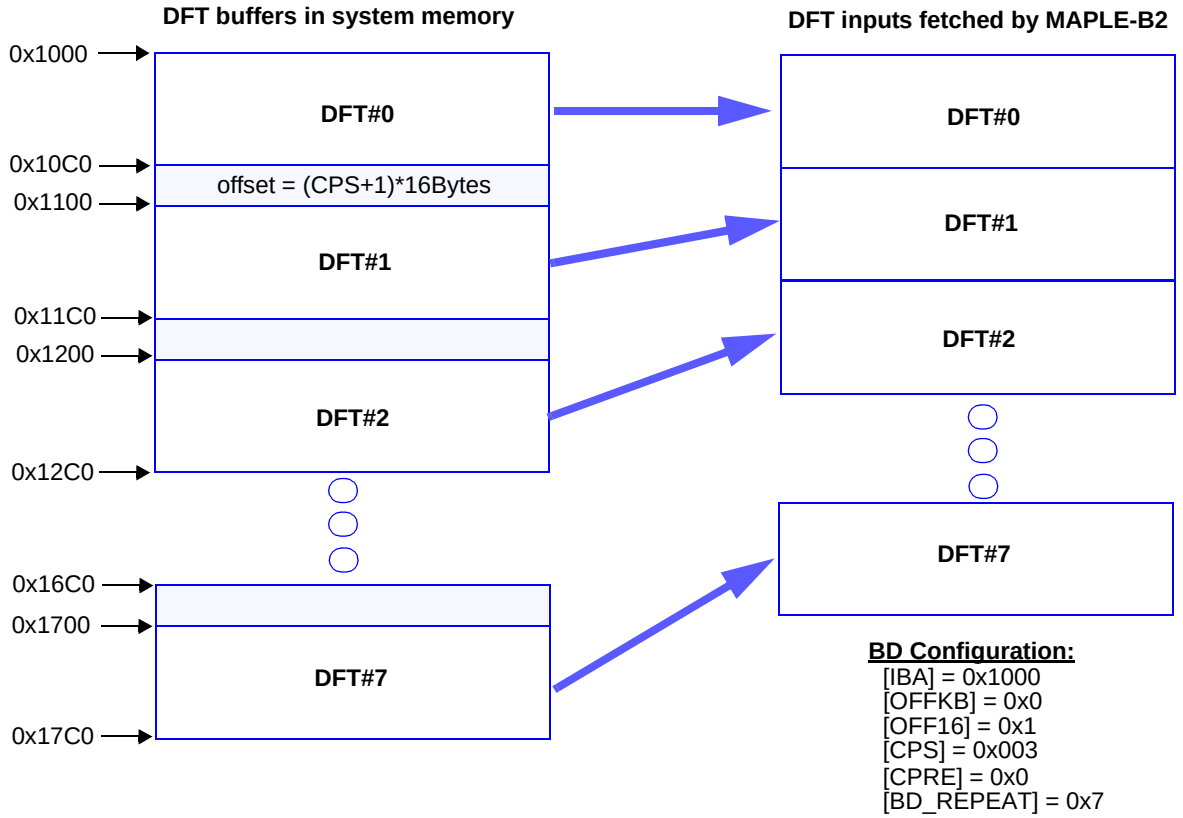


Figure 26-47. Input Buffers Offset (16 bytes multiplications) During BD Repeat

26.4.3.3.4.2 Output Data Structure.

Once the eFTPE 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 eFTPE is identical to the 32 bits sample input data structure as described in **Table 26-23**, that is, it is composed of 32 bit samples, and it is written to system memory with the same structure as the input data buffer (see **Figure 26-40**).

If *BD_RPT* is enabled (*BD_RPT* > 0), the MAPLE-B2 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 26-41**).

26.4.3.3.4.2.1 Guard Band Removal for FFT

To reduce system loading, the eFTPE supports a special output data structure for FFT processing only (*ITE* = 0). This special output data structure allows the MAPLE-B2 to output only the relevant parts of the output data.

Figure 26-48 is an example of the internal structure in the frequency domain of a 1024 samples Transform after FFT Processing[1].

As seen in **Figure 26-48**, 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 (not relevant for UL FFT processing of 3GLTE)

When the Guard Removal bit (*GR* bit in the eFTPE BD) is enabled, the MAPLE-B2 outputs only the Left-data following by the Right-data to system memory as seen in **Figure 26-48**.

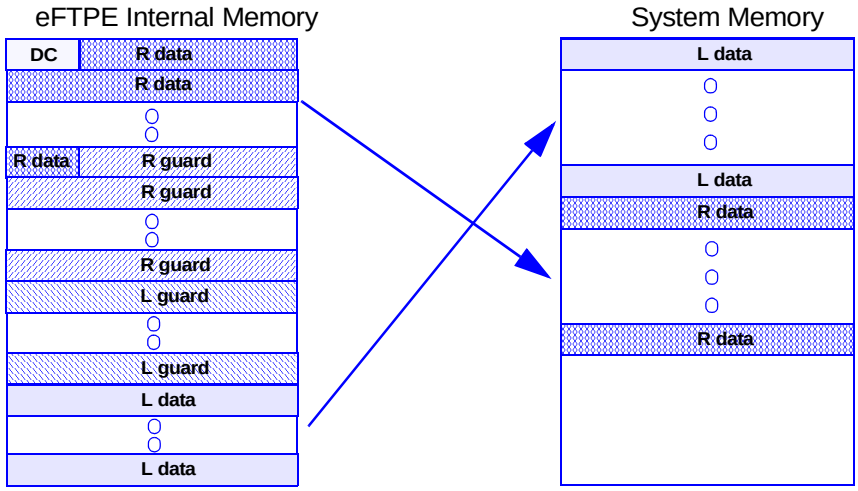


Figure 26-48. Output Data Structure of FFT Processing When *GR* Is Enabled

The *GRDC* bit of the eFTPE BD indicates MAPLE-B2 if the DC sample in the eFTPE output buffer should be regarded as data or as DC thus should be skipped during the output data stage. If the *GRDC* bit is set, MAPLE-B2 read the first R_data sample (see **Figure 26-48**) from offset 0x4 of the eFTPE output buffer. If the *GRDC* is cleared, MAPLE-B2 read the first R_data sample from offset 0x0.

The size of the Right/Left data for each of the possible transform lengths (R_data & L_data in **Figure 26-48**) should be specified to MAPLE-B2 as follows:

1. During MAPLE-B2 initialization one (or more) of the FTPEDSSxP parameter sets (see **Section 26.5.3.3.1**, *eFTPE Data Size Set x Parameter 0 (FTPEDSSxP0)* to **Section 26.5.3.3.3**, *eFTPE Data Size Set x Parameter 2(FTPEDSSxP2)*) must be initialized with the Data Size values for the relevant FFT Transform Lengths.
2. The Data Size Set (*DSS)* field of the eFTPE BD should get a value between ‘001’ to ‘111’ indicating the MAPLE-B2 which of the FTPEDSSxP parameter sets (which was initialized during MAPLE-B2 initialization as described in step #1) should be used for this job. The MAPLE-B2 read the relevant parameter set and output the required data from eFTPE output buffer accordingly.

1. For 3GLTE standard the DC sample does not exist after UL FFT processing.

Note: The *GR* bit can be used only while processing an FFT job (*ITE* = 0). It must be cleared for any other operation.

Note: The FTPEDSSxP parameter sets are shared to all eFTPE modules.

26.4.3.3.4.2.2 Cyclic Prefix Insertion for iFFT

To reduce system loading, the MAPLE-B2 supports generating cyclic prefix addition to the iFFT results while written into the system memory. By enabling the *CPIE* bit and the *CPS* field in the eFTPE BD, the MAPLE-B2 generates the following data structure in the system memory:

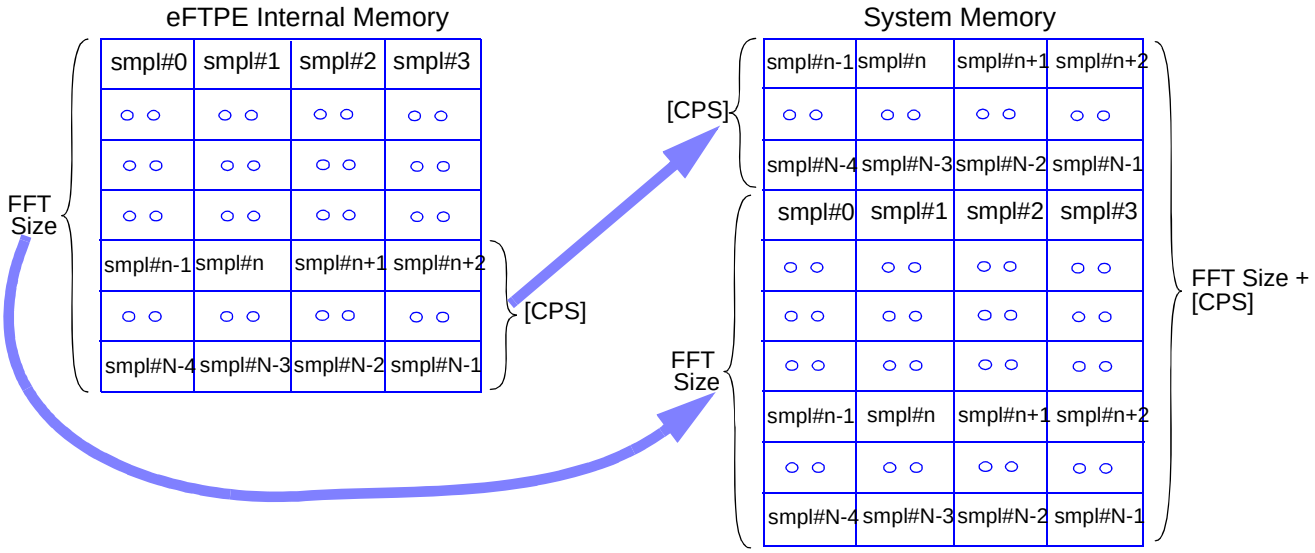


Figure 26-49. Cyclic Prefix Insertion in System Memory for iFFT Processing

As shown in **Figure 26-49**, if the Cyclic Prefix Insertion Enable bit (*CPIE*) in the eFTPE BD is set, the MAPLE-B2 first outputs the number of samples as described in the Cyclic Prefix Size (*CPS*) to the system memory, following by the whole FFT size as being processed by the eFTPE.

Note: The *CPS* must not exceed the Transform Length.

Note: If the Repeat mode is enabled (*BD_RPT* > 0) while the Cyclic Prefix Generation is enabled, the MAPLE-B2 generates the data structure as described in **Figure 26-49** for each of the repeated jobs.

Note: Disabling the Cyclic Prefix Generation (*CPIE* = 0) may slightly improve the eFTPE performance as the data output stage becomes shorter thus enabling the eFTPE to start with the next job earlier.

26.4.3.3.5 Pre and Post Vector Multiplication Support

The eFTPE supports pre and/or post vector multiplication of the processed samples. Pre multiplication means each sample of the input vector is multiplied (before the DFT/FFT

processing) by each sample of the pre-multiplier buffer in the eFTPE. Post multiplication means each sample of the DFT/FFT processing result is multiplied by each sample of the post-multiplier buffer in the eFTPE. The following sections elaborate on these features.

26.4.3.3.5.1 Pre-Multiplication Processing Support in the eFTPE

The eFTPE includes a pre-multiplication unit which allows array multiplication of the eFTPE 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.

Enabling the pre-multiplication feature in the eFTPE is done by configuring the *PME* field of the eFTPE BD as follows:

- *PME* = 00: Pre-multiplication is disabled.
- *PME* = 01: Pre-multiplication is enabled. No pre-multiplication memory update is required.
- *PME* = 11: Pre-multiplication is enabled. The MAPLE-B2 is required to update the pre-multiplication memory before the beginning of the processing. The MAPLE-B2 update the content of the pre-multiplication memory with data pointed by the *PRM_PTR* field of the eFTPE BD.

The order of the multiplications of the complex numbers in the pre-multiplication memory is always executed according to the order to 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 8KB to fit the maximum FFT size of 2048 (4 bytes for each complex number). Consequently, for smaller Transform Length sizes the pre-multiplication memory is not fully utilized.

Figure 26-50 describes the pre-multiplication memory utilization for the different TL sizes and the order of multiplications of its complex numbers for FFT/iFFT operation. This functionality is applicable for DFT/iDFT operation as well with similar memory organization:

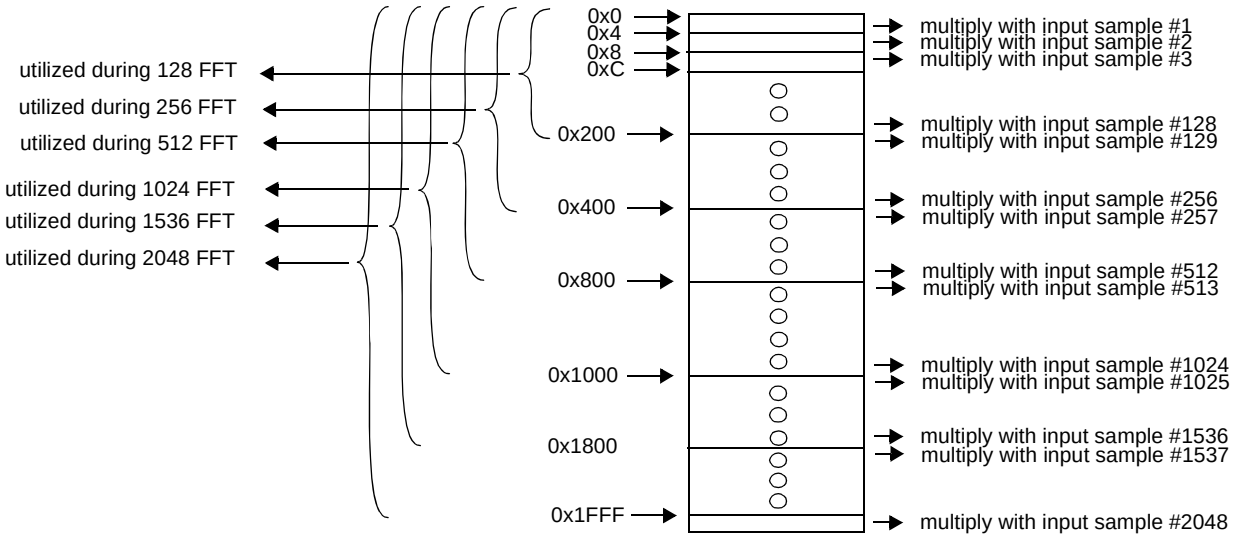


Figure 26-50. Pre- Multiplication Memory Utilization for the Different FFT Sizes

Note: If the *GI* bit of the eFTPE BD is asserted during iFFT processing, the expected input data is different (see **Section 26.4.3.3.4.1.1**, *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 input data order during *GI*. For example in case of 2048 points iFFT in 3GLTE technology with *GI* enabled, the pre-multiplier memory should contain 1200 samples, starting with Left data pre-multiplier samples followed by Right data pre-multiplier samples. The data structure is similar to the one shown in **Figure 26-43**.

Note: It is recommended to minimize the pre-multiplication memory update to allow maximum eFTPE throughput performance. Frequent updating of the pre-multiplication memory degrades eFTPE performance.

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 36.211, section 5.6 as well as for many other purposes in various technologies.

Note: Pre-multiplication can not be enabled when Frequency Correction is done on input samples. For details see **Section 26.4.3.3.7**, *Frequency Correction Support in eFTPE*.

When Pre-Multiplication is enabled with a BD Repeat option enabled (*BD_RPT* > 0), there are three options to handle the pre multiplication vector update for the internal tasks of the BD:

- All the tasks in the BD Repeat use the same pre-multiplication vector which is already placed in the eFTPE Pre-Multiplier internal memory. To enable this work flow the *PME* field should be set to the value of 1 and the pre-multiplication memory in the eFTPE must contain the required pre-multiplication vector. For details on initializing the pre-multiplication buffer see **Section 26.4.3.3.5.3**, *'One-Shot' Initialization of the Pre/Post Multiplication buffers*.
- All the tasks in the BD Repeat use the same pre-multiplication vector, but it is a new pre-multiplication vector which requires fetching from system memory. Enabling this work flow requires the following BD configuration: *PME*=3 and *PRMUR*=1. For this configuration the MAPLE-B2 fetches a single pre-multiplication vector expected at the address pointed by the *PRM_PTR* field and with the size of a single task as implied by the *TL_ID* field of the BD.
- Each of the tasks in the BD Repeat use a different pre-multiplication vector, which requires fetching from the system memory. Enabling this work flow requires the following BD configuration: *PME*=3 and *PRMUR*=0. For this configuration the MAPLE-B2 fetches a different pre-multiplication vector for each of the tasks in the BD Repeat. Each of the pre-multiplication vectors is expected to be with the size of the single task as implied by the TL_ID field of the BD. All the pre-multiplication vectors are expected to be placed successively in the system memory at the address pointed by the *PRM_PTR* field of the BD.

Note: Setting the *PME* to the value of 1 indicates that pre-multiplication is enabled and that all the tasks in the BD repeat use the same pre-multiplication vector already placed in the eFTPE pre multiplication internal buffer. Clearing the *PRMUR* indicates that the pre-multiplication buffer requires update between each task in the BD Repeat. Such configuration creates a contradiction hence forbidden.

Figure 26-51 described an example of the expected pre-multiplication data for the above options #2 and #3 (for option #1 no pre-multiplication data is expected).

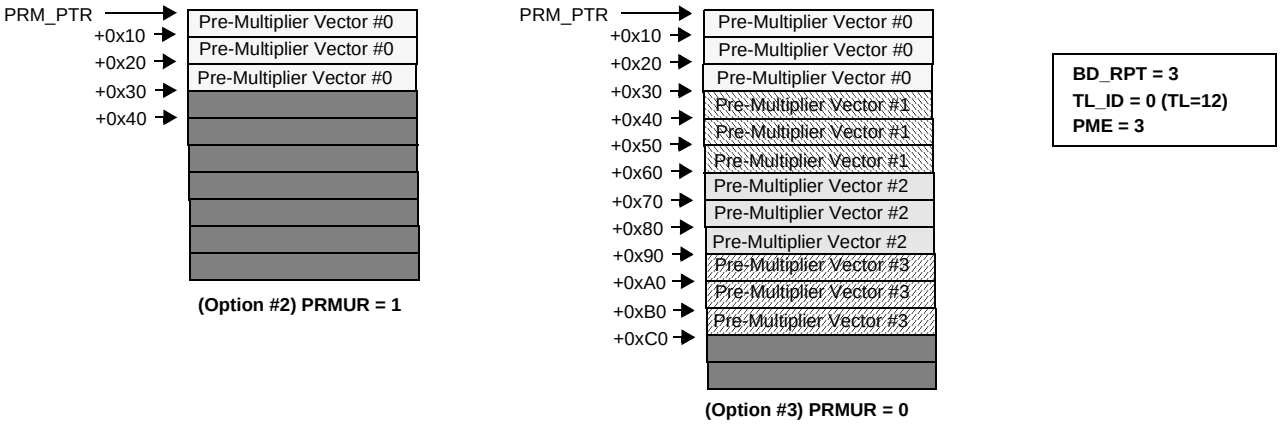


Figure 26-51. Pre-Multiplication Expected Vectors Example

26.4.3.3.5.2 Post-Multiplication processing support in the eFTPE

The eFTPE includes Apost-multiplication unit which allows array multiplication of the eFTPE samples after DFT/FFT processing 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.

Enabling the post-multiplication feature in the eFTPE is done by configuring the *PSTMV* field of the eFTPE BD as follows:

- *PSTMV* = 00: Post-multiplication is disabled.
- *PSTMV* = 01: Post-multiplication is enabled. No post-multiplication memory update is required.
- *PSTMV* = 11: Post-multiplication is enabled. The MAPLE-B2 is required to update the post-multiplication memory before the beginning of the processing. The MAPLE-B2 update the content of the post-multiplication memory with data pointed by the *POM_PTR* field of the eFTPE BD.

The order of the multiplications of the complex numbers in the post-multiplication memory is always executed according to the order to the input samples stream, that is, the first input sample (after DFT/FFT processing) is multiplied by the complex number located in address 0x0 of the post-multiplication memory, the second input sample is multiplied by the complex number located in address 0x4 of the post-multiplication memory and so forth. The post-multiplication memory size is 8KB to fit the maximum FFT size of 2048 (4 bytes for each complex number). Consequently, for smaller Transform Length sizes the post-multiplication memory is not fully utilized.

Figure 26-50 on page 26-92 describes the post-multiplication memory utilization (identical to the pre-multiplication memory utilization) for the different TL sizes and the order of multiplications of its complex numbers

When Post-Multiplication is enabled with a BD Repeat option enabled (*BD_RPT* > 0), there are three options to handle the post multiplication vector update for the internal tasks of the BD:

- All the tasks in the BD Repeat use the same post-multiplication vector which is already placed in the eFTPE Post-Multiplier internal memory. To enable this work flow the *PSTMV* field should be set to the value of 1 and the post-multiplication memory in the eFTPE must contain the required post-multiplication vector.For details on initializing the pre-multiplication buffer see **Section 26.4.3.3.5.3**, *‘One-Shot’ Initialization of the Pre/Post Multiplication buffers*.
- All the tasks in the BD Repeat use the same post-multiplication vector, but it is a new post-multiplication vector which requires fetching from system memory. Enabling this work flow requires the following BD configuration: *PSTMV*=3 and *PSMUR*=1. For this configuration the MAPLE-B2 fetches a single post-multiplication vector expected at the

address pointed by the *POM_PTR* field and with the size of a single task as implied by the *TL_ID* field of the BD.

- Each of the tasks in the BD Repeat use a different post-multiplication vector, which requires fetching from the system memory. Enabling this work flow requires the following BD configuration: *PSTMV*=3 and *PSMUR*=0. For this configuration the MAPLE-B2 fetches a different post-multiplication vector for each of the tasks in the BD Repeat. Each of the post-multiplication vectors is expected to be with the size of the single task as implied by the TL_ID field of the BD. All the post-multiplication vectors are expected to be placed successively in the system memory at the address pointed by the *POM_PTR* field of the BD.

Note: Setting the *PSTMV* to the value of 1 indicates that post-multiplication is enabled and that all the tasks in the BD repeat use the same post-multiplication vector already placed in the eFTPE post multiplication internal buffer. Clearing the *PSMUR* indicates that the post-multiplication buffer requires update between each task in the BD Repeat. Such configuration creates a contradiction hence forbidden.

Figure 26-51 described an example of the expected post-multiplication data for the above options #2 and #3 (for option #1 no post-multiplication data is expected).

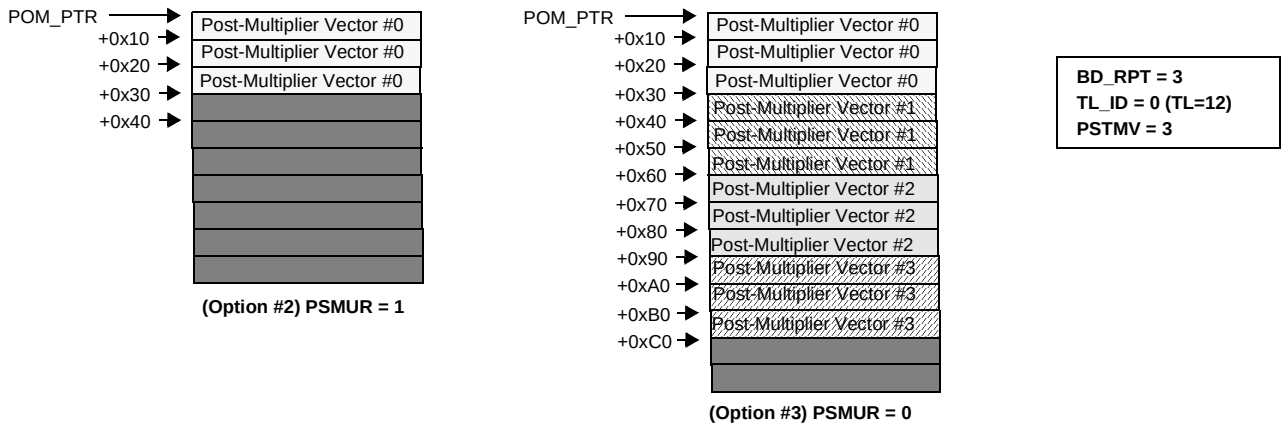


Figure 26-52. Post-Multiplication Expected Vectors Example

Note: It is recommended to minimize the post-multiplication memory update between BDs to allow maximum eFTPE throughput performance. Frequent updating of the post-multiplication memory degrades eFTPE performance.

Note: Post-multiplication can not be enabled when Frequency Correction (FC) is enabled on output samples (*FCPRE* = 0), as the FC on output samples uses the post multiplication buffer to keep the generated FC samples. Such implementation implies that when FC on output samples is enabled, the post-multiplication buffer is invalid and needs to be reloaded on the next job that the vector post-multiplication feature is enabled. For details see **Section 26.4.3.3.7**, *Frequency Correction Support in eFTPE*.

26.4.3.3.5.3 'One-Shot' Initialization of the Pre/Post Multiplication buffers

In case no frequent update of the pre/post multiplication buffers is required, the MAPEL-B2 support initiating a dedicated internal routine ('Maple_eftpe_update_pre_post_buffers') which allows single update of the pre/post multiplication buffers of each of the eFTPE modules. To initiate the 'Maple_eftpe_update_pre_post_buffers' routine, the following should be done:

1. Set the *FTPExUPRMBPP*, *FTPExUPSMBPP* and *FTPExUBSP* parameters as described in **Section 26.5.3.3.4** to **Section 26.5.3.3.6**
2. Clear the *FTPECUBRP[UBC]* field (see **Section 26.5.3.3.7**, *eFTPE Complete Update Buffers Routine Parameter (FTPECUBRP)*).
3. Access the PCR register with opcode of '010000' (see **Section 26.4.5**, *MAPLE-B2 Internal Task Control*).

Once initiated the routine performs (for each eFTPE module) the following:

1. Check if *FTPExUPRMBPP[PRE_BUF_PTR]* is initialized, that is, has a different value than zero. If yes, the value is assumed to point to the new pre-multiplication buffer in the system memory and the MAPLE-B2 fetched the new table into the relevant eFTPE pre-multiplication buffer. The size of the table is determined according to the *FTPExUBSP[PRE_BUF_SIZE]* configuration. For details see **Section 26.5.3.3.4** and **Section 26.5.3.3.6**
2. Check if *FTPExUPSMBPP[PRE_BUF_PTR]* is initialized, that is, has a different value than zero. If yes, the value is assumed to point to the new post-multiplication buffer in the system memory and the MAPLE-B2 fetched the new table into the relevant eFTPE post-multiplication buffer. The size of the table is determined according to the *FTPExUBSP[PST_BUF_SIZE]* configuration. For details see **Section 26.5.3.3.5** and **Section 26.5.3.3.6**

Activating this routine should be done with the following limitations:

- All relevant eFTPE modules are in idle and there are not pending jobs (Buffer Descriptors) in their BD Rings.
- All relevant eFTPE modules output buffer configuration (see **Section 26.4.3.3.10.1**, *eFTPE Configuration Register*) meet the following: *BC* != 4.

Once the 'Maple_eftpe_update_pre_post_buffers' routine is initiated, no new Buffer Descriptors should be written into MAPLE-B2 internal memory until the MAPLE-B2 has completed the routine. The completion of the routine is indicated by MAPLE-B2 by setting the *FTPECUBRP[UBC]* bit to '1'.

26.4.3.3.6 Scalar Post Multiplication in eFTPE

The eFTPE allows multiplication of all resulting samples with a single constant complex value. Using this feature can resolve, for example, the need to multiply all DFT output samples with the value of 1/n (or 1/sqrt(n)), where n is the DFT size. Enabling this feature is done by setting the Post Multiplication Enable bit (*PSTME*) in the eFTPE BD, and by assigning the required multiplication value to the *PSTM_REAL* and *PSTM_IMG* fields of the eFTPE BD. The single limitation when using the Post Multiplication feature is that the values of the complex multiplication number represented by the *PSTM_REAL* and *PSTM_IMG* fields of the eFTPE BD must maintain the following equation:

$$(PSTM\_REAL)^2 + (PSTM\_IMG)^2 < 2$$

where *PSTM_REAL* and *PSTM_IMG* are 1q15 representation numbers as described in **Figure 26-249**.

26.4.3.3.7 Frequency Correction Support in eFTPE

The eFTPE support vector multiplication of the transform vector with internally generated Frequency Correction (FC) function. The function is a sequence of Phasor arguments with a constant shift between each argument. The FC arguments generation is executed “on the fly” in the eFTPE internal pipe, therefore doesn’t have any impact on the eFTPE performance.

When enabling the FC option, the following eFTPE BD parameters should be initialized:

- ***FC_CG***: Should be set. Indicates the MAPLE-B2 that the BD extension, where all FC related parameters are placed, is valid (see **Section 26.5.4.3.2**, *eFTPE Buffer Descriptor’s Extension*).
- ***FCG***: FC Generation indication. Setting this bit enables the FC Generation in the eFTPE.
- ***FCPRE***: FC as Pre-Multiplier. The eFTPE supports applying the FC function on either samples data before the DFT/FFT processing or samples data after the DFT/FFT processing. By assigning the value of ‘0’ to the *FCPRE*, the FC multiplication is performed on samples after the DFT/FFT processing. By assigning the value of ‘1’ to the *FCPRE*, the FC multiplication is performed on samples before the DFT/FFT processing.
- ***FCB_REAL***, ***FCB_IMAG***: FC Base value (Real part and Imaginary part). Used as the initial value to be used for the first sample.
- ***FCS***: FC Size. Indicates the eFTPE the amount of samples (out of the total Transform Length) which the FC function should be applied.
- ***FCSS***: FC Step Size. Indicates the number of input samples that should use the same correction factor before it is increment by the shift value.
- ***FCISC***: FC Input Step Counter. Indicates the initial value of the step counter which determines when to increment the correction factor. For example, if for a certain job the Step Size (*FCSS*) is set to 20, and the Step Counter init value (*FCISC*) is set to 16, the

eFTPE uses the FC Base value for the first 4 samples before incrementing it by the shift value.

- ***FCS_REAL***, ***FCS_IMAG***: FC Shift Value (Real part and Imaginary part). Used as the step value whenever Correction factor increment (complex multiplication) is required.

The following pseudo code illustrates the eFTPE usage of the above parameters:

```
if ((FC_CG==1) and (FCG==1))
{
      correction_factor = FCB_REAL, FCB_IMAG; //correction factor init value
      correction_step = FCISC;
      for (i=1; i ≤ FCS; i++) // increment every input sample (starting the input first sample)
      {
           Sample = Sample * correction factor;
           if (FCSS==1) {correction_factor = correction_factor * {FCS_REAL, FCS_IMAG};}
           else
           {
                correction_step += 1;
                if ((correction_step % FCSS)==0)
                {correction_factor = correction_factor * {FCS_REAL, FCS_IMAG};}
           }
      }
}
```

where “Sample” is either the input data before DFT/FFT processing (*FCPRE*=1) or after DFT/FFT processing (*FCPRE*=0).

Some notes regarding FC usage:

- The FC can not be enabled in the following cases:
  - If the Guard Insertion feature is enabled (*GI* bit of the eFTPE BD. See **Section 26.4.3.3.4.1.1**).
  - If Pre-multiplication is enabled (*PME* field of the eFTPE BD. See **Section 26.4.3.3.5.1**) then FC can not be enabled as Pre-Multiplication (*FCPRE*=1).
  - If Post-multiplication is enabled (*PSTMV* field of the eFTPE BD. See **Section 26.4.3.3.5.2**) then FC can not be enabled as Post-Multiplication (*FCPRE*=0).
- FC can be enabled along with the Code Generation feature. In that case, the “Sample” expression refers to the internally generated data as described in **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*
- Frequent switching between FC as Pre (*FCPRE*=1) and FC as Post (*FCPRE*=0) is not recommended because it degrades eFTPE performance.

The MAPLE-B2 returns the following FC related status fields in the eFTPE BD on BD completion:

- ***FCF_REAL_ST***, ***FCF_IMAG_ST***: FC Factor (Real part and Imaginary part). Indicates the correction factor value end state. Can be used as the *FCB_REAL* and *FCB_IMAG* values for the next (successive in time domain) job.
- ***SCOS***: Step Counter State. Indicates the end state of the step counter. Can be used as the FCISC value for the next (successive in time domain) job.

26.4.3.3.7.1 Frequency Correction Support during BD Repeat

When enabling the Frequency Correction (FC) feature and BD Repeat is enabled (*BD_RPT*>0), the MAPLE-B2 assume continuous FC is required and the following is applied by the MAPLE-B2:

1. The first repeated job is using all configuration parameters as described in the BD.
2. The followed repeated job (job #n) use the results of the previous job (job #(n-1)) in the repeat chain as follows:
   a. The FC factor results (*FCF_REAL_ST* and *FCF_IMAG_ST* status fields) of the previous job are used as the FC Base values (*FCB_REAL* and *FCB_IMAG*) for the current job .
   b. The step counter state (*SCOS* status field) of the previous job is used as the FC initial step counter (*FCISC*) for the current job
3. All other BD fields are valid for all jobs in the repeated BD.

Note: When enabling the FC during BD repeat, the output results of each job in the repeated BD are used as input control for the following job in the repeated BD. This flow prevents full utilization of the eFTPE as the internal pipeline between repeated jobs cannot be implemented.

26.4.3.3.8 WCDMA Scrambled Pilot Code Generation

The eFTPE supports WCDMA processing assist tasks which include FFT processing on scrambled and spread symbols of pilot fields generated internally in the eFTPE. The eFTPE can internally produce the scrambled symbols (according to given parameters) and perform FFT processing on the generated sequence according to given FFT parameters, thus both reducing system traffic and external core load required for these sequences generation.

When enabling the Code Generation (CG) option, the following eFTPE BD parameters should be initialized:

- ***FC_CG***: Should be set. Indicates the MAPLE-B2 that the BD extension, where all CG related parameters are placed, is valid (see **Section 26.5.4.3.2**, *eFTPE Buffer Descriptor's Extension*).
- ***CG***: Code Generation indication. Setting this bit enables the internal CG in the eFTPE.

- ***CNPI***: Current Number of Pilots. Indicates the N_{pilot} value (see 3GPP 25.211 section 5.2) of the current frame which the code start at. There are six valid values (0 to 5) indicating N_{pilot} values of 3 to 8.
- ***NNPI***: Next Number of Pilots. Indicates the N_{pilot} value of the next frame. Required for the case where the required code starts at the current frame and cross to the next frame during its generation. There are six valid values (0 to 5) indicating N_{pilot} values of 3 to 8.
- ***CO***: Chip Offset indication. Indicates the chip offset from the beginning of the current frame where the scrambling code generation is to start. The chip offset can be any value from 0 to 38399 (number of chips in frame).
- ***SGO***: Slot Generation Off. Each of this 4 bits field represent a slot, starting from the current slot (bit [0]) and until the following 3 slots. A value of '1' in a bit indicates that the pilot code generation for the slot it is representing is disabled.
- ***LSS***: Long Scrambling Sequence. Indicates the eFTPE the scrambling sequence generation mode. There are two supported modes: long and short. Setting this field indicated the eFTPE that long scrambling sequence mode is required. Resetting this field indicates short scrambling sequence generation mode is required.
- ***SSN***: Scrambling Sequence Number. A 24 bits number (0 to $(2^{24} - 1)$) which is used for the scrambling code generation.
- ***CL***: Code Length. Indicates the total number of the scrambled symbols to be generated in the current job.
- ***SN***: Slot Number. Indicates the index of the current slot in the frame.

The above parameters are used to generate the pilots and the scrambling code which are then combined to generate the final scrambled code to be used as input to the FFT processing. The following sections elaborate on Pilot generation, scrambling code generation and the combining process of the two.

26.4.3.3.8.1 Pilot Symbols Generation and Spreading

The eFTPE generates pilot symbols from DPCCH to be scrambled according to DPCCH definition in 3GPP 25.211. The pilot symbols are chosen from an internal table according to the slot format (*CNPI*), slot number (*SN*), and the chip offset in the slot (*CO*). Also, there are "silent" slots, which indicate slots that do not contain any relevant pilot symbols, and therefore create 0 symbols for the slot. For these slots, the pilot generation is disabled. Enabling silent slots is done by setting the relevant bits in the Slot Generation Off (*SGO*) field. The first bit of the *SGO* field specified the silent indication of the current slot in which the first chip is required. The next bits of the *SGO* specify the silent indication for the following slots, in case the chip index is crossing slots boundaries during pilot generation.

Figure 26-53 describe the internal processing of pilot symbols generation:

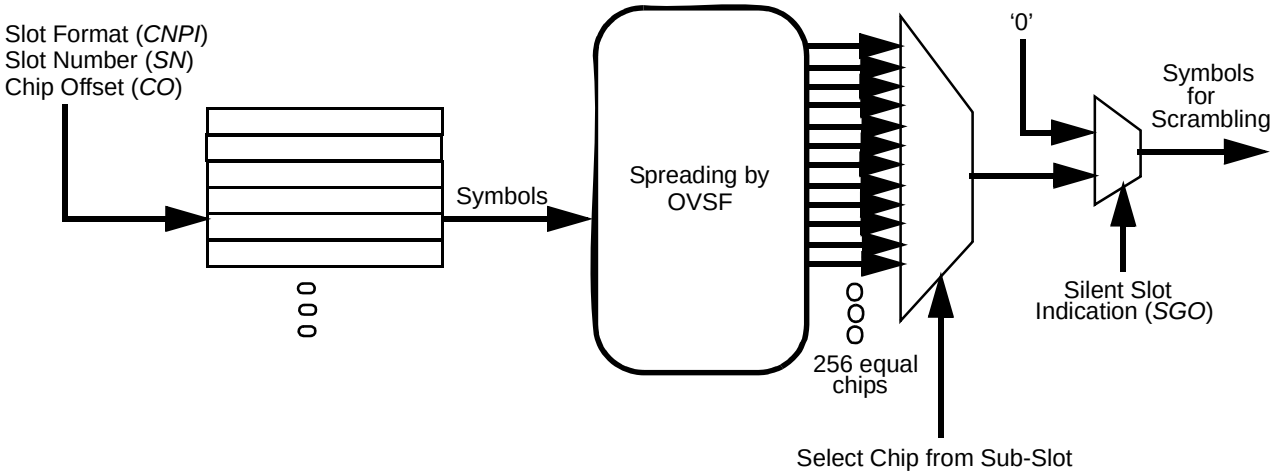


Figure 26-53. Pilot Generation Illustration

As can be seen in **Figure 26-53**, the symbols generation is done using a table (taken from the 3GPP 25.211 standard) which uses the current slot format, slot number and chip offset to extract the symbol, which is then spread using Orthogonal Variable Spreading Factor (OVSF). For DPCCH symbols, the OVSF equals 256 '1's. Note that if during the pilot generation a frame boundary is crossed, the eFTPE uses the slot format number specified in the *NNPI* field for the following symbols generation. The resulting chips either continue *as is* to scrambling or replaced with zero in case the current slot is silent slot. The values of the generated pilots before scrambling are either 1 or -1 (representing j or -j), and when the current slot is silent (or when current chip offset is bigger than the spread number of pilots), the symbols value is 0.

Note: To maximize internal calculations precision the generated pilots are created with the values of +0.5/-0.5 and the input scale value is assumed as +1. For details on eFTPE scaling see **Section 26.4.3.3.9.4**, *eFTPE Internal Scaling Calculations*

26.4.3.3.8.2 Scrambling Code Generation

The scrambling machine is designed to implement the 3GPP 25.213 section 4.3.2, including supporting long and short scrambling sequences. To get the requested scrambling code, the machine uses the Chip Offset (*CO*), Long/Short Scrambling Sequence indication (*LSS*) and the Scrambling Sequence Number (*SSN*) parameters as described in the standard.

26.4.3.3.8.3 Combining Pilot Symbols and Scrambling Code

The final generated code sample to be written to input data buffer (SP[i]), is combined from the scrambling machine result for the requested chip (PN[i]), and the pilot symbols machine result for the same requested chip (P[i]).

Creating SP[i] is done by multiplying (complex multiplication) PN[i] with P[i]:

SP[i] = PN[i]*P[i].

The final code samples results could be one of the following: 1+j, 1-j, -1+j, -1-j and 0.

26.4.3.3.8.4 Code Generation with BD Repeat

When enabling the Code Generation (CG) feature and BD Repeat is enabled (*BD_RPT*>0), the MAPLE-B2 assume continuous CG is required and the following is applied by the MAPLE-B2:

1. The first repeated job is using all configuration parameters as described in the BD.
2. The followed repeated job (job #n) use the results of the previous job (job #(n-1)) in the repeat chain as follows:
   a. The Next Number of Pilots results (*NNPIS* status field) of the previous job is used as the Current Number of Pilots (*CNPI* field) for the current job.
   b. The Next Slot Generation Off indication (*NSGO* status field) of the previous job is used as the Slot Generation Off indication (*SGO*) for the current job.
   c. The Next Slot Number result (*NSN* status field) of the previous job is used as the Slot Number (*SN*) indication for the current job.
   d. The Next Chip Offset results (*NCO* status field) of the previous job is used as the Chip Offset (*CO*) field for the current job
3. All other BD fields are valid for all jobs in the repeated BD.

Note: When enabling the CG during BD repeat, the output results of each job in the repeated BD are used as input control for the following job in the repeated BD. This flow prevents full utilization of the eFTPE as the internal pipeline between repeated jobs cannot be implemented.

Note: The generated codes included in the BD Repeat should not be spread over more than 4 slots.

26.4.3.3.8.5 Code Generation with Zero Padding

In some cases the required pilot code size is not aligned with the supported FFT sizes. In that case it is possible to instruct the eFTPE to add zero padding to the generated code to reach the required FFT size. Enabling the zero padding can be done as follows:

1. Configure *TL_ID* according to the required FFT size (see **Table 26-246**)
2. Configure the required Code Length in the *CL* field of the eFTPE BD.
3. If (*CL* < Transform Length,) then set the Zero Padding field (*ZP*) and configure the Data Size to Zero Padding (*DSTZP*) field must to the same value as the *CL* field, which directs the eFTPE to add zero padding to the generated code to reach the required Transform length

26.4.3.3.8.6 Code Generation Status

The MAPLE-B2 returns the following CG related status fields in eFTPE BD:

- ***NNPIS***: Next Number of Pilot State. If the current job crossed a frame boundary during its CG processing, then that value of this field equals the *NNPI* field, else it equals the *CNPI* field.
- ***NSGO***: Next Slot Generation Off. If the current job crossed slots boundaries during its CG processing, then the value of this field would be:

  NSGO = *SGO* >> (number of slot boundaries crossed)

  else it would be equal to *SGO* field.
- ***NSN***: Next Slot Number. If the current job crossed slots boundaries during its CG processing, then the value of this field would be:

  NSN = (*SN* + (number of slot boundaries crossed)) % 15

  else it would be equal to *SN* field.
- ***NCO***: Next Chip Offset. The value of this field would be:

  NCO = (*CO* + *CL*) % 38400

These statuses can be used as configuration inputs for the next BD (if required).

26.4.3.3.9 eFTPE Processing

The following sections describe the different aspects of the eFTPE processing.

26.4.3.3.9.1 eFTPE 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.

The following figures show an example of Mixed Radix DIF (Decimation in Frequency) partitioning of a transform length 24 into three stages[1]. During the first stage the eFTPE executes DFT-4 using its Radix-4 module; during the second stage the eFTPE 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 26-54** describes the mathematics of stage 0 partitioning into DFT-4 and DFT-6:

1. This example is for explanatory purposes only, not representing exactly the implementation of DFT24 in eFTPE.

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi nk}{N}} \qquad k = 0, \ldots, N-1$$

$$X(k) = \sum_{n=0}^{23} x(n) \cdot e^{-j\frac{2\pi nk}{24}} = \sum_{n=0}^{5} x(n) \cdot e^{-j\frac{2\pi nk}{24}} + \sum_{n=6}^{11} x(n) \cdot e^{-j\frac{2\pi nk}{24}} + \sum_{n=12}^{17} x(n) \cdot e^{-j\frac{2\pi nk}{24}} + \sum_{n=18}^{23} x(n) \cdot e^{-j\frac{2\pi nk}{24}}$$

$$= \sum_{n=0}^{5} x(n) \cdot e^{-j\frac{2\pi nk}{24}} + \sum_{n=0}^{5} x(n+6) \cdot e^{-j\frac{2\pi nk}{24}} \cdot e^{-j\frac{\pi k}{2}} + \sum_{n=0}^{5} x(n+12) \cdot e^{-j\frac{2\pi nk}{24}} \cdot e^{-j\pi k} + \sum_{n=0}^{5} x(n+18) \cdot e^{-j\frac{2\pi nk}{24}} \cdot e^{-j\frac{3\pi k}{2}}$$

$$= \sum_{n=0}^{5} \{x(n) + (-j)^k x(n+6) + (-1)^k x(n+12) + (j)^k x(n+18)\} \cdot e^{-j\frac{2\pi nk}{24}}$$

$$X(4k) = \sum_{n=0}^{5} \{x(n) + x(n+6) + x(n+12) + x(n+18)\} \cdot e^{-j\frac{2\pi nk}{6}}$$

$$X(4k+1) = \sum_{n=0}^{5} [x(n) - jx(n+6) - x(n+12) + jx(n+18)] \cdot e^{-j\frac{2\pi n}{24}} \cdot e^{-j\frac{2\pi nk}{6}}$$

$$X(4k+2) = \sum_{n=0}^{5} [x(n) - x(n+6) + x(n+12) - x(n+18)] \cdot e^{-j\frac{4\pi n}{24}} \cdot e^{-j\frac{2\pi nk}{6}}$$

$$X(4k+3) = \sum_{n=0}^{5} [x(n) + jx(n+6) - x(n+12) - jx(n+18)] \cdot e^{-j\frac{6\pi n}{24}} \cdot e^{-j\frac{2\pi nk}{6}}$$

DFT 6 DFT 4 Twiddles DFT 6

Figure 26-54. Example Mixed Radix Partitioning of Transform Length 24 (First Stage Mathematics)

Figure 26-55 shows the first stage partitioning implementation:

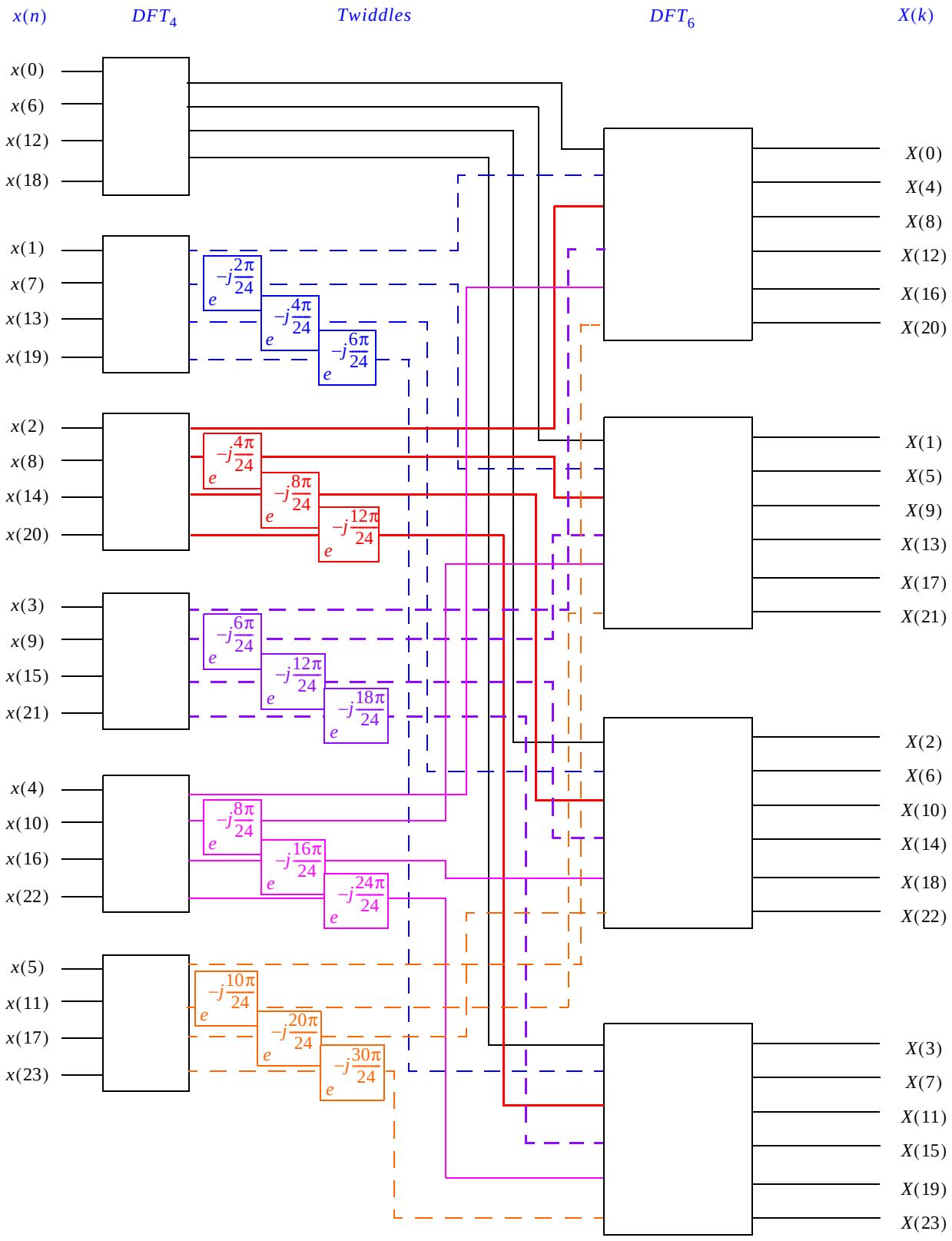


Figure 26-55. Example Mixed Radix Partitioning of Transform Length 24 (First Stage Implementation)

Figure 26-56 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

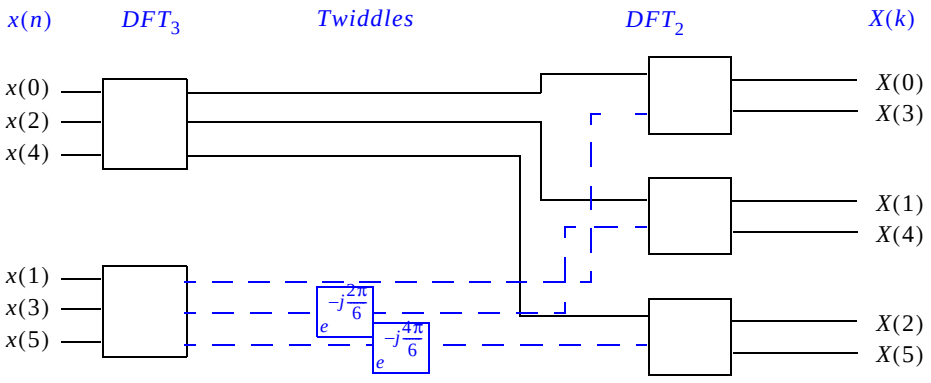


Figure 26-56. Example Mixed Radix Partitioning of Transform Length 24 (Second and Third Stage Mathematics)

26.4.3.3.9.2 Inverse Transform Processing

The eFTPE 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 eFTPE BD must be set.

26.4.3.3.9.3 Transform Length Partitioning

For each possible DFT/iDFT/FFT/iFFT transform length (as listed in **Section 26.2**, *MAPLE-B2 Features*) the eFTPE 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.

The implemented Radices in the eFTPE are: Radix- 8, Radix- 5, Radix- 4 and Radix- 3.

The Transform Length of each job is determined in the *TL_ID* field of the eFTPE BD (see **Table 26-246**).

Table 26-26 describes the partitioning of all the possible DFT transform lengths of the eFTPE. Each stage column describes the executed Radix for that stage.

Table 26-26. Partitioning Path for Supported DFTPE Transforms

Transform Length	Stage 0	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
12	4	3				
24	8	3				
36	4	3	3			
48	4	4	3			
60	5	4	3			
72	8	3	3			
96	8	4	3			
108	4	3	3	3		
120	8	5	3			
144	4	4	3	3		
180	5	4	3	3		
192	8	8	3			
216	8	3	3	3		
240	5	4	4	3		
288	8	4	3	3		
300	5	5	4	3		

Table 26-26. 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	8	5	3	3		
384	8	4	4	3		
432	4	4	3	3	3	
480	8	5	4	3		
540	5	4	3	3	3	
576	8	8	3	3		
600	8	5	5	3		
648	8	3	3	3	3	
720	5	4	4	3	3	
768	8	8	4	3		
864	8	4	3	3	3	
900	5	5	4	3	3	
960	8	8	5	3		
972	4	3	3	3	3	3
1080	8	5	3	3	3	
1152	8	4	4	3	3	
1200	5	5	4	4	3	
1536	8	8	8	3		
128	8	4	4			
256	8	8	4			
512	8	8	8			
1024	8	8	4	4		
2048	8	8	8	4		

26.4.3.3.9.4 eFTPE Internal Scaling Calculations

The eFTPE supports two main types of scaling methods, which are applied in between the DFT/FFT calculation stages and are targeted to reduce or eliminate the need to saturate samples to avoid overflow.

The scaling methods are:

- User Defined Scaling
- Adaptive Scaling

Along with these two scaling methods, which are applied during the transform processing, there are additional parameters which affect to total scaling calculations in the eFTPE:

- Overall Scaling Amount Configuration. Allows to align the final scaling results during Adaptive Scaling mode to a required value. For more see **Section 26.4.3.3.9.4.4**, *Overall Scaling Amount Programming*.
- Input Exponent. Allows inputting the eFTPE with the input exponent (scale) value of the current job. Used by the eFTPE when calculating the Overall Scaling. For details see **Section 26.4.3.3.9.4.5**, *Input Exponent*
- User Defined Input Scaling. Allows to up-scale the input samples by a configurable number. For details see **Section 26.4.3.3.9.4.6**, *User Defined Input Scaling*
- Adaptive Input Scaling. If enabled the eFTPE performs maximal up scale of all the input samples to maximize precision and avoid overflow. for details see **Section 26.4.3.3.9.4.7**, *Adaptive Input Scaling*.
- Extra Scaling. Allows incrementing the final scale results (exponent value) during Adaptive Scaling mode on the expenses of the output samples data (mantissa). for details see **Section 26.4.3.3.9.4.8**, *Extra Scaling*.

Some of the above scaling configurations may results in internal overflow of the samples data (mantissa) and therefore requires saturation. For details see **Section 26.4.3.3.9.4.9**, *Saturation*.

26.4.3.3.9.4.1 Scaling During Internal Radix Calculations.

In the eFTPE implementation, each Radix module inputs a different number of bits. The number of input bits to each Radix depends on the maximum bit expansion of the Radix. The number of output bits of all the Radix stages is 20, with data representation of 5q15[1]. The number of input bits to each Radix stage is described in **Table 26-27**.

Table 26-27. Input Bits per Radix

Radix	Input Bits	Input data representation
3	17	2q15
4	17	2q15
5	16	1q15
8	16	1q15

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.

26.4.3.3.9.4.2 User Defined Scaling

Given the information on the number of input and output bits for each Radix (**Table 26-27**), and the Radix stage order for each transform length (**Table 26-26**), the eFTPE can be programmed to execute down scaling after each stage according to user programming, to prevent the need to saturate overflown samples. To enable User Defined Scaling, the *SCL_TYPE* bit in the eFTPE BD must be set. Using the *USR_SCLx* (x = 0... 5) fields, the *PM_SCL* field and the *PO_SCL* field of the eFTPE BD, the eFTPE can be programmed to execute down scaling after each of the possible stages. For example, by assigning a value to *USR_SCL0*, the eFTPE performs down scaling on the output of the Radix which was applied during stage 0 with the scaling factor in the *USR_SCL0*.

The *USR_SCLx* fields can be programmed with values from 0 to 4 as the amount of down scaling to be performed. The *PM_SCL* field can be programmed with values of 0 or 1 as the potential down scaling after the pre-multiplication processing (see **Section 26.4.3.3.5.1**, *Pre-Multiplication Processing Support in the eFTPE*) is 1. The *PO_SCL* field can be programmed with values of 0 to 2 as the potential down scaling after the post-multiplication processing (see **Section 26.4.3.3.5.2**, *Post-Multiplication processing support in the eFTPE*) is 2.

26.4.3.3.9.4.3 Adaptive Scaling

In Adaptive Scaling mode the eFTPE performs down scaling after each processing stage if needed. The amount of scaling applied after each processing stage is the minimum down scaling necessary to prevent overflow of any of the samples during the next processing calculation.

To enable Adaptive Scaling, the *SCL_TYPE* bit of the eFTPE BD must be cleared.

The total amount of down or up scaling performed by the eFTPE during the transform processing using Adaptive Scaling mode is accumulated and then written to the *ADP_OVA_SCL_ST* field of the eFTPE BD.

If *BD_RPT* is enabled, which means more than one task for a single BD, the total overall scaling amount for all the jobs is transferred to system memory to the address pointed to by the *BD_STAT_PTR* field in the eFTPE BD as described in **Section 26.4.3.3.2**, *BD Repeat Option*.

Figure 26-57 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 (see **Section 26.4.3.3.9.4.6**, *User Defined Input Scaling*)
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. Since the next stage performs Radix 2, the output samples of the current Radix (Radix 3) are scaled down by 2 bits to fit the Radix 2 input size (18 bits).

5. To fit the 16 bits output requirement, the 20 bit results of the Radix 2 as described in **Figure 26-57** are scaled down by 1 bit and the 3 msbs are removed (since the 4 msbs are identical the eFTPE consider the 3 msbs as sign extension).

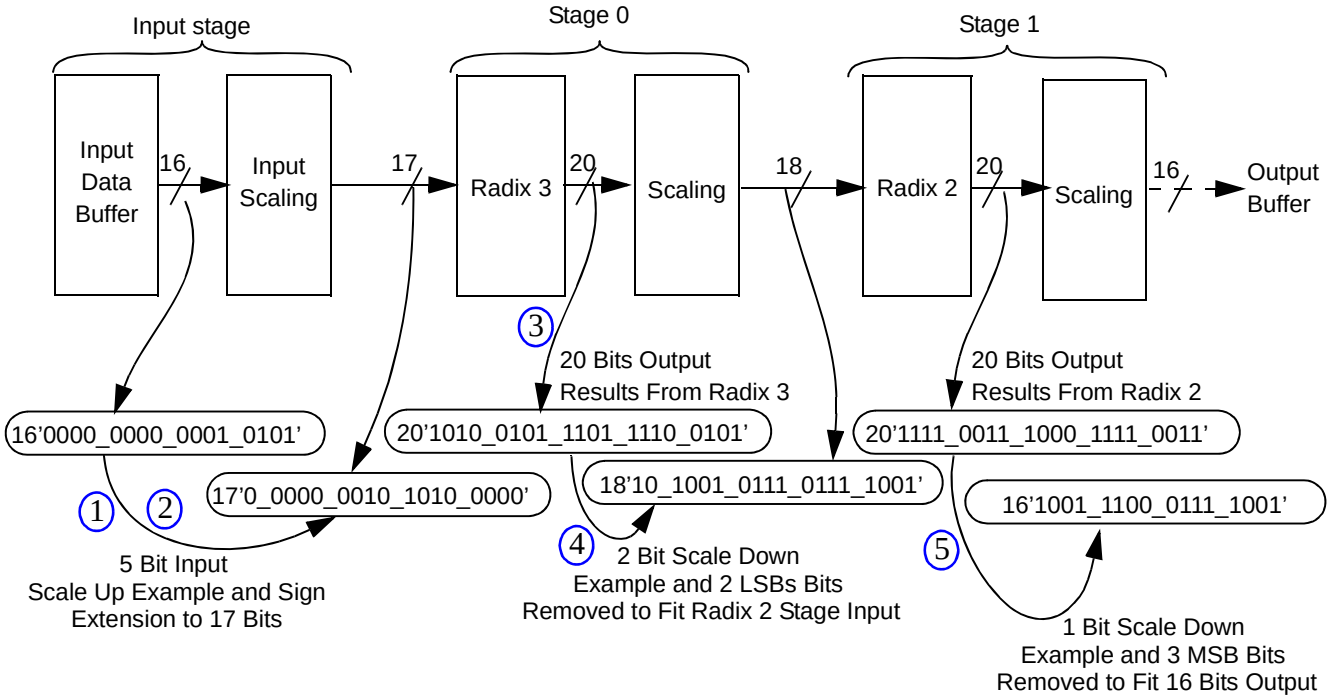


Figure 26-57. Input Scaling Example and Stage Scaling Example (Adaptive or User Defined)

Note: The previous example describes a partitioning of transform length of 6 (3x2), which is not a supported transform length of the eFTPE. The example is for explanation only.

26.4.3.3.9.4.4 Overall Scaling Amount Programming

The host can define the overall amount of scaling to be applied during the transform processing when using Adaptive Scaling mode. If enabled, the eFTPE adjusts its accumulated scaling amount to the Overall Scaling factor programmed by the host. The scaling the eFTPE 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 (down scaling), the eFTPE upscales the final results by 2. If the required Overall Scaling is 8 (down scaling), the eFTPE downscales the final results by 2.

To enable the Overall Scaling factor, the *OVA_SCL* bit in the eFTPE BD must be set, and the *ADP_OVA_SCL* field in the eFTPE BD, must contain the Overall Scaling factor. If Overall Scaling is enabled, the *ADP_OVA_SCL_ST* status indications in the eFTPE BD is equal to the *ADP_OVA_SCL* field of the eFTPE BD, since the later determines the overall scaling which is applied during the transform processing.

26.4.3.3.9.4.5 Input Exponent

In general, the eFTPE assume the block floating point scale value (exponent) of the input data is zero. It is possible to define any other input block floating point scale (exponent) value by configuring the *INP_EXP_PTR* field (bits [31:24]) of the eFTPE BD. This scale value is used as the base scale value for the calculations of the Adaptive Overall Scaling Status calculations as described in **Section 26.4.3.3.9.4.3**, *Adaptive Scaling*, that is, the internal accumulated scaling in the eFTPE is added to the input exponent value configured in the *INP_EXT_PTR* field and reported to the *ADP_OVA_SCL_ST*.

If Overall Scaling Amount Programming is used (see **Section 26.4.3.3.9.4.4**, *Overall Scaling Amount Programming*), the eFTPE add the configurable exponent value (configured in the *INP_EXP_PTR* field of the eFTPE BD) to the total Overall Scaling Amount required to calculate the required alignment of the output results. For example, if the configured Input Exponent is 2, the internal eFTPE scaling result is 4 and the required Overall Scaling amount is 8, then the eFTPE aligns the output results by additional 2 to reach the required results.

If BD Repeat option is enabled (*BD_RPT* > 0), the eFTPE allows different Input Exponent values for each of the input blocks. As the eFTPE BD can not include all the Input Exponent values of all the potential tasks of the BD (up to 128), the *INP_EXP_PTR* field should contain an address pointer to the system memory where the Input Exponent values of all the tasks of the BD should be placed. The MAPLE-B2 expect to find the Input Exponent values (8 bits value) of all the tasks of the BD as described in the following figure:

Figure 26-58. Input Exponent Vector in Case of BD Repeat

By setting the value of the *INP_EXP_PTR* to zero, the MAPLE-B2 assume that the input scale value of all the tasks in this repeat job is zero.

26.4.3.3.9.4.6 User Defined Input Scaling

This feature is relevant for input data to the eFTPE that is characterized as small value samples which, with additional scale up, results in more precise output results.

The eFTPE can be programmed to scale up the input data by up to 2^{15} (15 left shifts). This is done by configuring the *IN_SCL* field of the eFTPE BD with values between 1 to 15. The value of 0 means no input scaling is performed. If overflow occurs due to input scaling, the eFTPE uses the largest/smallest number possible (sign dependent), instead of the overflown sample.

User Defined Input scaling can only up scale (shift left) the input data.

The User Defined Input Scaling value, if enabled, is also included in the final Adaptive Overall Scaling Status indication reported in the *ADP_OVA_SCL_ST* field of the eFTPE BD. For example, if the User Defined Input Scaling value configuration is 3, that is, the input data samples (mantissa) are being shift left by 3 bits (up scale), and the total scaling accumulated in the processing is 1 (down scale), the reported Adaptive Overall Scaling Status is “-2”.

If Overall Scaling Amount is enabled (see **Section 26.4.3.3.9.4.4**, *Overall Scaling Amount Programming*), the User Defined Input Scaling value is also included in the total Scaling calculation required to reach the required Scale value as configured in the *ADP_OVA_SCL* field of the eFTPE BD. The following example describe the internal eFTPE scaling calculation:

Assuming:

- Input Exponent value (*INP_EXP_PTR*) is 4.
- User Defined Input Scaling value (*IN_SCL*) is 2.
- Internal eFTPE processing scaling results is 3.

The Adaptive Overall Scaling Status would be: 4 - 2 + 3 = 5

Note: On the MAPLE-B, used in the MSC8156 devices, the Input Scaling value, if enabled, was not included in the final Adaptive Overall Scaling Status nor in the Overall Scaling Amount (if enabled). To keep backwards compatibility with the MAPLE-B block, an additional bit is added to the eFTPE Configuration Register (see EFTPE_<x> Configuration Register (FTPE<x>CR): *BCIS*. If set, the eFTPE does not include the Input Scaling configuration in either the final Adaptive Overall Scaling Status or the Overall Scaling Amount (if enabled).

26.4.3.3.9.4.7 Adaptive Input Scaling

In additional to the User Defined Input Scaling (see **Section 26.4.3.3.9.4.6**, *User Defined Input Scaling*), the eFTPE supports automatic increment of the scale value of the input data samples to reach maximum precision during the calculations. By setting the *AIUS* bit of the eFTPE BD, the eFTPE adaptively up-scales the input data samples thus gaining maximum precision during calculations. Up scaling of the input samples means finding the maximum allowed shift-left operations to be executed on each of the input samples so that none of the samples is overflown thus saturated. Note that this automatic increment of the scale values is limited to 9 bits only.

The Adaptive Input Scaling, if enabled, is also included in the final Adaptive Overall Scaling Status indication reported in the *ADP_OVA_SCL_ST* field of the eFTPE BD. For example, if the eFTPE has up-scaled the input samples by 3 (due to enabling the Adaptive Input Scaling), that is, the input data samples (mantissa) are shift left by 3 bits (up scale) to reach maximum precision, and the total scaling accumulated in the processing is 1 (down scale), the reported Adaptive Overall Scaling Status is -2.

If Overall Scaling Amount is enabled (see **Section 26.4.3.3.9.4.4**, *Overall Scaling Amount Programming*), the Adaptive Input Scaling value is also included in the total Scaling calculation required to reach the required Scale value as configured in the *ADP_OVA_SCL* field of the eFTPE BD

26.4.3.3.9.4.8 Extra Scaling

In general, when using Adaptive Input Scaling (*AIUS*=1) with Adaptive Scaling (*SCL_TYPE*=0) with Overall Scaling Amount disabled (*OVA_SCL*=0), the eFTPE output data samples are normalized, that is, at least one of the real or imaginary parts of one of the samples begin with 01 (for positive value) or 10 (for negative values). To allow complex scaling schemes (see **Section 26.4.3.3.2**, *BD Repeat Option*), the eFTPE supports increasing the scale value by up to 2 while shifting right the mantissa value by up to 2. To enable the Extra Scaling, configure the *EXS* field of the eFTPE BD with the required number (0 to 2). **Figure 26-59** describe an example of the Extra Scaling functionality:

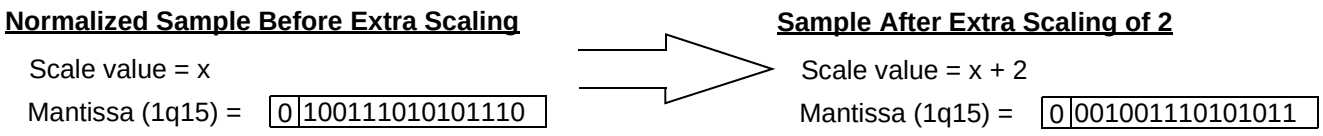


Figure 26-59. Extra Scaling Functionality Example

The Extra Scaling, if enabled, is also included in the final Adaptive Overall Scaling Status indication reported in the *ADP_OVA_SCL_ST* field of the eFTPE BD. For example, if the total accumulate scaling of the internal eFTPE processing is 6 and the Extra Scaling configuration is set to 1, the reported Adaptive Overall Scaling Status in the *ADP_OVA_SCL_ST* field is 7 and the samples mantissas are right shifted by one.

26.4.3.3.9.4.9 Saturation

When using User Defined Input scaling (see **Section 26.4.3.3.9.4.6**, *User Defined Input Scaling*) or User Defined Scaling (see **Section 26.4.3.3.9.4.2**, *User Defined Scaling*) or Overall Scaling (see **Section 26.4.3.3.9.4.4**, *Overall Scaling Amount Programming*), some of the data samples may overflow due to the internal calculations. To avoid the overflow, the eFTPE uses the largest/smallest number possible (depending on the sign bit) instead of the overflown sample. This is called saturation. When such saturation event occur, an indication bit is set by MAPLE-B2 in the *CMP_RSN* status field of the eFTPE BD (see **Section 26.4.3.3.11**, *eFTPE Status Indications*).

26.4.3.3.10 eFTPE Initialization Configuration

After the MAPLE-B2 initialization (see *MAPLE-B Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)), initialize the following eFTPE configuration registers:

26.4.3.3.10.1 eFTPE Configuration Register

The EFTPE_<x> Configuration Register (FTPE<x>CR) includes basic eFTPE configuration fields that must be initialized once for each eFTPE instance (after MAPLE-B2 initialization) and remain unchanged for the whole operation duration. The relevant configuration fields are:

- ***BCIS*** - Backward Compatible Input Scaling indication. Determines whether the User Defined Input Scaling (see **Section 26.4.3.3.9.4.6**, *User Defined Input Scaling*) is included in the Overall Scaling Amount calculations (see **Section 26.4.3.3.9.4.4**, *Overall Scaling Amount Programming*) and in the Adaptive Overall Scaling Status indication. In the MAPLE-B of the MSC8156 family of devices the User Defined Input Scaling was not included in the Overall Scaling Amount calculations and in the Adaptive Overall Scaling Status indication. If backward compatibility is required with respect to these features, set this field.
- ***BC*** - Buffers Configuration Indication. In general, each eFTPE instance includes a pre-multiplier input buffer, Apost-multiplier input buffer and an output buffer. In some applications, where vector pre-multiplication or vector post multiplication are not required, it is possible to configure the eFTPE to use the redundant buffer for other usage to boost eFTPE performance. The following buffer usage configurations are allowed after MAPLE-B2 initialization:
  - (BC = 0): Single Pre-Multiplier buffer, Single Post Multiplier Buffer and Single Output buffer.
  - (BC = 1): Single Pre-Multiplier buffer, no Post Multiplier buffer and Double Output buffer.
  - (BC = 2): No Pre-Multiplier buffer, single Post Multiplier Buffer and Double Output buffer.
  - (BC = 4): No Pre-Multiplier buffer, Double Post Multiplier Buffer and Single Output buffer.

In general, allowing double buffer for the Post Multiplication Input Buffer or for the Output buffer may in some cases boost the eFTPE total performance. It allows more flexibility to the MAPLE-B2 and reduces the time at which the eFTPE is halted due to waiting for the next post multiplication input data or due to waiting for the output buffer to be empty by the MAPLE-B2 DMA. Enabling double buffer does not require any change in the eFTPE programming model as is handled internally by the MAPLE-B2.

Note: If Frequency Correction is scheduled to be enabled as post-multiplication (*FC*=1 and *FCPRE*=0), then the post-multiplication buffer must be allocated in the *BC* configuration (*BC* = 0,2,4) because the FC is using the post-multiplication buffer to maintain the FC samples before multiplying them with the processed data.

Note: Enabling a double post multiplication buffer is highly recommended when frequent change of the post multiplication vector is required.

Note: If either vector pre-multiplication is not required or vector post-multiplication is not required, it is recommended to work with any of the double output buffer configurations thus increasing eFTPE performance.

26.4.3.3.10.2 Data Size Registers

The Data Size Registers (see **Section 26.5.5.3.1** to **Section 26.5.5.3.3**) holds the values of the data for the Guard band Insertion. If working with fixed values per Transform Length then it is recommended to initialize these values during the eFTPE initialization. If different data values are required for each job, it can be implemented by working with the Data Size parameters (see **Section 26.5.3.3.1**, *eFTPE Data Size Set x Parameter 0 (FTPEDSSxP0)* to **Section 26.5.3.3.3**, *eFTPE Data Size Set x Parameter 2(FTPEDSSxP2)*) as described in **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT*. In that case there is no need to initialize these registers.

26.4.3.3.11 eFTPE Status Indications

The eFTPE BD has the following status indicators which are fetched from the eFTPE after each BD completion.

- *CMP_RSN* —This field in the eFTPE BD describes the status of the completed job. It indicates whether the job completed OK or completed with saturation events during the processing.
- *ADT_OVA_SCL_ST* —This field in the eFTPE BD describes the Adaptive Overall Scaling accumulated in the current job. For details see **Section 26.4.3.3.9.4**, *eFTPE Internal Scaling Calculations*
- *NNPIS*—This field in the eFTPE BD is valid only if Code Generation is enabled. It describes the next number of pilot state for pilot generation. For details see **Section 26.4.3.3.8.6**, *Code Generation Status*
- *NSGO*—This field in the eFTPE BD is valid only if Code Generation is enabled. It describes the next slot generation off period for pilot generation. For details see **Section 26.4.3.3.8.6**, *Code Generation Status*.
- *NSN*—This field in the eFTPE BD is valid only if Code Generation is enabled. It describes the next slot number for pilot generation. For details see **Section 26.4.3.3.8.6**, *Code Generation Status*
- *NCO*—This field in the eFTPE BD is valid only if Code Generation is enabled. It describes the next chip offset for pilot generation. For details see **Section 26.4.3.3.8.6**, *Code Generation Status*
- *FCF_REAL_ST*—This field in the eFTPE BD is valid only if Frequency Correction is enabled. It describes the frequency correction real part after job ends. For details see **Section 26.4.3.3.7**, *Frequency Correction Support in eFTPE*

- *FCF_IMAG_ST*—This field in the eFTPE BD is valid only if Frequency Correction is enabled. It describes the frequency correction imaginary part after job ends. For details see **Section 26.4.3.3.7**, *Frequency Correction Support in eFTPE*
- *SCOS*—This field in the eFTPE BD is valid only if Frequency Correction is enabled. It describes the step counter state after job completion.

26.4.3.3.12 eFTPE ECC Support

As part of the MAPLE-B2 ECC support (see **Section 26.4.3.1.1, *Memory Error Correction/Detection Support***), the eFTPE include a status register which specify the exact memory module in which the ECC error has occurred.

If the general ECC error indication of the MAPLE-B2 is asserted, and by reading the PSIF PIC Event Register 2 (PSPICER2), an eFTPE instance ECC indication bit is asserted, it means that the source of the ECC error is in that certain eFTPE instance. Reading the EFTPE_<x> ECC Interrupt Status Register (FTPE<x>ECCISR) gives indication as to which internal eFTPE memory was the source of the ECC error. Resetting an eFTPE ECC error bit indication is done by writing ‘1’ to the relevant bit.

The following steps indicate the required actions to identify the source of the ECC interrupt:

1. Read the PSIF PIC Event Register 2 (PSPICER2) to detect which PE is the source of the ECC interrupt
2. If any of the EFxECC bits is asserted than the source of the ECC interrupt is in one of the eFTPEs.
3. Read the relevant FTPExECCISR register to identify the exact memory module which was the source of the ECC interrupt.

To reset the ECC interrupt, the following action must occur:

1. Clear the ECC error bit indications in the EFTPE_<x> ECC Interrupt Status Register (FTPE<x>ECCISR) by writing ‘1’ to the relevant bits.
2. Clear the *EFxECC* error bit from the PSIF PIC Event Register 2 (PSPICER2) by writing ‘1’ to the bit.

26.4.3.4 DEPE Downlink Turbo Encoding Operation

The MAPLE-B2 supports Turbo Encoding and Rate-Matching for each of its operation modes as described in **Section 26.2**, *MAPLE-B2 Features* using the Downlink Encoder Processing Element (DEPE).

26.4.3.4.1 DEPE Buffer Descriptors and Header Structures

The DEPE uses BDs to configure the general DEPE operation. In addition, the DEPE uses header structures to specialize the operations for 3GLTE, WiMAX, and UMTS operations. DEPE buffer descriptors are described in detail in **Section 26.5.4.4.1**, *DEPE Buffer Descriptor Structure*, on page 26-439. The header structures are described as follows:

- 3GLTE, see **Section 26.5.4.4.3.1**, *DEPE Header Structure for 3GLTE*, on page 26-448.
- WiMAX, see **Section 26.5.4.4.3.2**, *DEPE Header Structure for WiMAX (802.16e)*, on page 26-450. and **Section 26.5.4.4.3.3**, *DEPE Header Structure for WiMAX (802.16m)*, on page 26-452
- UMTS, see **Section 26.5.4.4.3.4**, *DEPE Header Structure for UMTS*, on page 26-454.

26.4.3.4.2 DEPE Multiple Headers (tasks) in single BD support

To reduce host BD write overhead and to enhance its encoding throughputs performance, the MAPLE-B2 is capable of receiving several CB encoding tasks in a single BD. By configuring the [NOH] field of the DEPE BD to any number greater than zero (up to 32 tasks), the MAPLE-B2 assumes multiple CB encoding tasks are assigned for the current BD. Due to BD fixed size limitations, it can not include multiple tasks Headers, therefore the MPALE-2B expect to find the concatenated tasks Headers in the system memory at the address pointed by the [HBA] field of the DEPE BD. **Figure 26-60** describe an example of the expected Headers data structure in the system memory with respect to the standard.

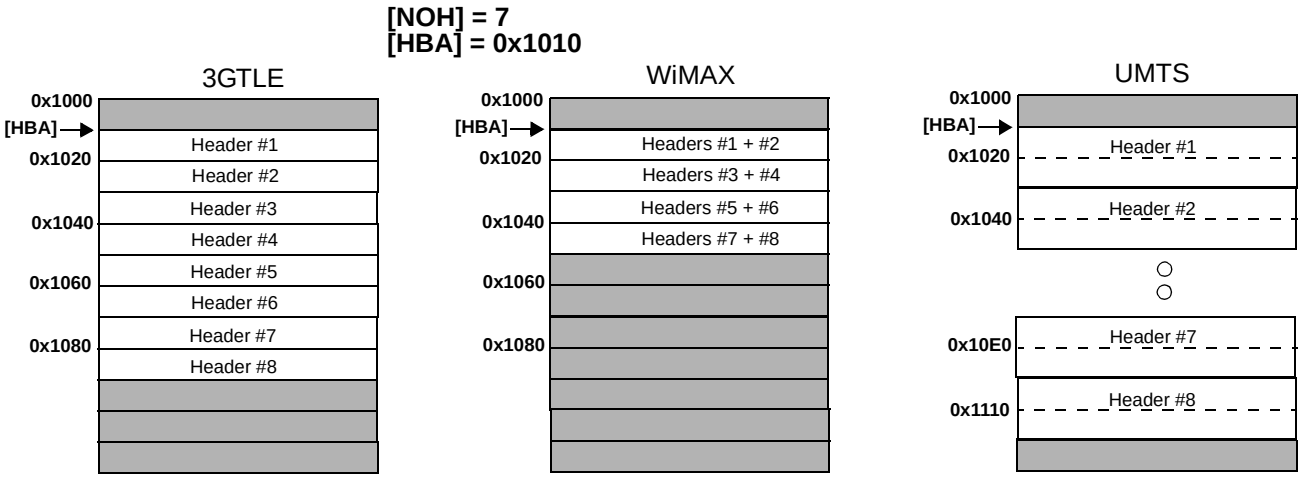


Figure 26-60. Expected Concatenated Headers in System Memory

Due to DEPE input buffer size limitation and to maintain internal pipeline between DEPE jobs, the total size of the tasks (Headers + data) in multiple task BD must not exceed 100 lines of 64 bits (or 800 bytes). Due to the different Header sizes for each standard, this limitation impose

different maximum number of tasks allowed for each of the standards. **Table 26-28** describe the maximum number of allowed tasks for each standard assuming minimal Code Block size.

Table 26-28. Maximum number of allowed tasks (Header + Data)

Technology	Minimum Code Block Size	Maximum Number of tasks in BD
WiMAX	48	32
UMTS	40	20
3G-LTE	40	32

- The expected data structure of the input data in case of multiple task BD is described in **Section 26.4.3.4.4.2**, *Input Data Structure for Multiple Tasks*.
- The data structure of the output data in case of multiple task BD is described in **Section 26.4.3.4.6.2**, *Output Data Structure for Multiple Tasks*
- The data structure of the status fields in case of multiple task BD is described in **Section 26.4.3.4.5.1**, *3GLTE Processing*.

26.4.3.4.3 [LH] Indication in DEPE Header

Each DEPE Header includes a Last Header ([LH]) indication bit. This bit indicates the DEPE whether the current Header is the last Header in the current BD. This bit must be configured as follows: if the current BD includes single DEPE Header ([NOH] = 0), the [LH] bit of this single Header must be set. If the current BD includes multiple Headers ([NOH] > 0), only the [LH] bit of the last Header must be set while the [LH] bit of all previous Headers must be reset.

26.4.3.4.4 DEPE Input Data Structure

The DEPE expected inputs are data bits to be encoded and rate matched. The [IBA] field of DEPE BD points to the address in the system memory where the MAPLE-B2 is to fetch the input data stream into the DEPE. The address pointed by [IBA] must be 4 bytes aligned.

The [IBS] field of the DEPE BD describe the total number of 4 bytes words which should be fetched by the MAPLE-B2 into the DEPE input buffer. As the DEPE input buffer is 64 bit aligned, the straightforward calculation for the [IBS] field is:

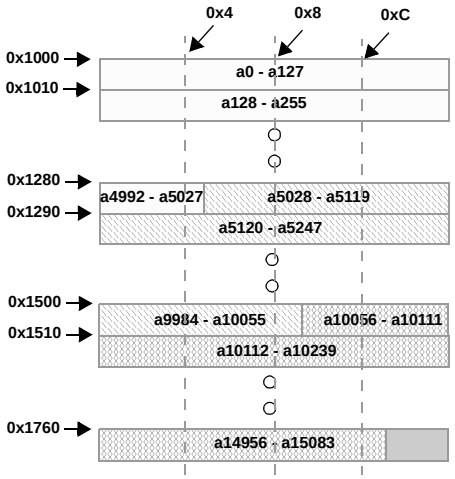
```
[IBS]= 2 * roundup([Code Block Size]/ 64)
```

Due to the multiple tasks support and to the input buffer offset support, the total number of 4 bytes words which should be fetched by MAPLE-B2 may not always be calculated straightforward. For details, see **Section 26.4.3.4.4.2**, *Input Data Structure for Multiple Tasks* and **Section 26.4.3.4.4.1**, *Input Buffer Offset Support (3GLTE and UMTS only)*.

26.4.3.4.4.1 Input Buffer Offset Support (3GLTE and UMTS only)

The DEPE supports up to 31 bits offset of the input stream with respect to the input address pointed by [IBA] field of the DEPE BD. Configuring the [IBO] field of the DEPE Header instruct the DEPE at which offset of the first 4 bytes word the first valid bit of the current CB is located. Supporting this offset along with the 32 bits resolution support of the Input Base Address pointer [IBA] of the DEPE BD allows pointing on the input stream with resolution of a single bit, thus eliminating the need to align the input CBs after segmentation to 4 bytes aligned buffers.

Figure 26-61 describe an example of 15084 bits Transport Block (TB) located in system memory. After segmentation this TB is divided into 3 CB each of 5028 bits. The following figure describe how to configure the [IBA] and [IBO] fields of the DEPE BDs/Headers to direct MAPLE-B2 into each of the CBs.



# BD	[IBA]	[IBO]
1	0x1000	0x0
2	0x1284	0x4
3	0x1508	0x8

Figure 26-61. Input buffer bit resolution pointer example using [IBA] and [IBO] fields

The Input Buffer Offset field is valid only for single task BD. For multiple task BD it is assumed that each task is different TB therefore it is required to be 64 bits aligned in the system memory.

When Input Buffer Offset is set to a value different than 0x0, the [IBS] field should be calculated as follows:

```
[IBS]= 2 * roundup(([Code Block Size]+ [IBO]) / 64)
```

Note: For [IBS] calculations, the *Code Block Size* value should consider additional supplements added by the DEPE in some cases. For example, if CRC is enabled or if the [NOF] field is enabled then the actual code block size would be:

```
Code Block Size = Code Block Size - 24 - [NOF]
```

26.4.3.4.4.2 Input Data Structure for Multiple Tasks

When multiple tasks in a BD is enabled (by configuring [NOH] > 0), the MAPLE-B2 expect to find the input data buffers of all the tasks successively in the system memory. Since the DEPE internal input buffer is 64 bits width, each input data buffer of each task must be 64 bits aligned.

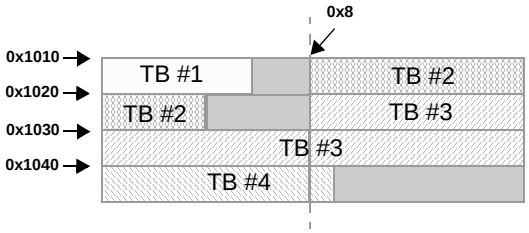
The expected Headers data structure for multiple tasks BD is described in **Section 26.4.3.4.2**, *DEPE Multiple Headers (tasks) in single BD support*.

When [NOH] is set to a value different than 0x0, the [IBS] field should be calculated as follows:

```
[IBS] =0;
for (i=0; i < [NOH]+1; i++)
      [IBS] = [IBS]+ [IBS[i]]
```

where $IBS_{[i]}$ is the Input Buffer Size of task [i].

Figure 26-62 describe an example of the expected input data structure for multiple tasks BD which includes multiple small Transport Blocks.



[NOH] = 3
[IBA] = 0x1010
[IBS] = 16

Task #	Task Size	[IBO]
1	40	0x0
2	96	0x0
3	192	0x0
4	72	0x0

Figure 26-62. Expected Input Buffer Data Structure for Multiple Tasks BD

Note: The above [IBS] calculations and expected input buffer data structure are not relevant when UMTS BD job is given with Transport Block indication ([TB] bit is set). In that case the expected input buffer is different and the [IBS] field is ignored. For details see **Section 26.4.3.4.5.4.6**, *UMTS Transport Block (TB) Support*.

26.4.3.4.5 DEPE Processing

The DEPE encoding and rate matching processing is different for each of the supported technologies. As per the current MAPLE-B2 operation mode, the DEPE uses the [STD] bit of the DEPE BD to determine if the job is of 3GLTE type ([STD] = 1), or of UMTS/WiMAX type ([STD] = 0). The following sections describe the DEPE encoding processing for each of the standards.

26.4.3.4.5.1 3GLTE Processing

The 3GLTE DEPE processing includes the following processing chain:

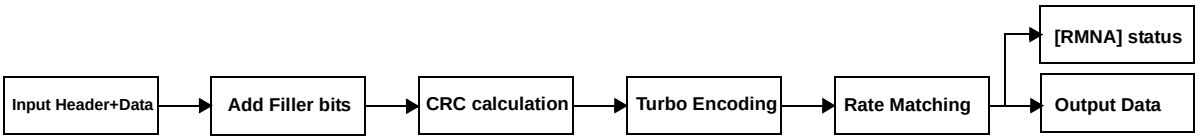


Figure 26-63. 3GLTE DEPE Processing chain

the block sizes allowed in the 3GLTE standards includes 188 possible types and varies from block size of 40 to block size of 6144. The Code Block size should be set in the [CBSI] field of the DEPE LTE Header according to the encoding described in **Table 26-263**.

26.4.3.4.5.1.1 Add Filler Bits

If the [NOF] field of the DEPE 3GLTE Header is greater than 0x0 the DEPE add [NOF] filler bits at the most significant bits of the first data line and concatenates the data block to the filler bits. **Figure 26-64** describe an example of the data buffer in the DEPE after adding the filler bits.

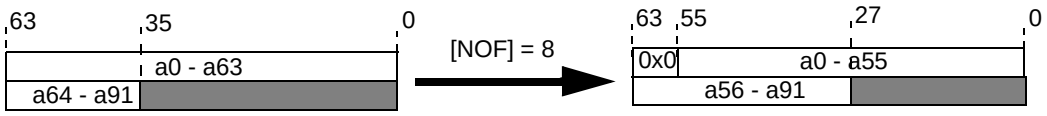


Figure 26-64. DEPE Data Buffer After Adding Filler Bits ([NOF]=8)

26.4.3.4.5.1.2 CRC Calculation

If the CRC Disable ([CD]) bit of the DEPE LTE Header is set, the CRC calculation is bypassed. If the [CD] bit of the DEPE LTE Header is reset, the CRC is calculated and attached to the end of the data block. The CRC machine is reset to zero for each new block encoded. The CRC Mode ([CM]) controls the CRC polynomial type as described in **Table 26-29**:

Table 26-29. 3GLTE CRC Polynomials

[CM] configuration	CRC SIZE	CRC TYPE	Polynomial
0	24	CRCA	$D^{24} + D^{23} + D^6 + D^5 + D + 1$
1	24	CRCB	$D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1$

26.4.3.4.5.1.3 Turbo Encoding

During Turbo Encoding process the data buffer is being encoded and forwarded to the Rate Matching process. The DEPE uses a binary encoder for the 3GLTE encoding as described in the 3GLTE standard. In addition the Turbo Encoding process uses a dedicated interleaver which interleaves the block according to the 3GLTE interleaving scheme. The 3GLTE interleaver scheme is selected for each block according to its size which is described in the [CBSI] field of the DEPE LTE Header.

26.4.3.4.5.1.4 Rate Matching

The 3GLTE Rate Matching process consists of the following steps:

1. Interleave the sub-block of each of the encoder output streams separately (systematic stream, 1^{st} parity stream and 2^{nd} parity stream).
2. Collect the 3 interleaved streams into a single virtual circular buffer
3. Select the output bits out of the virtual circular buffer.

Figure 26-65 describes the 3GLTE Rate Matching scheme:

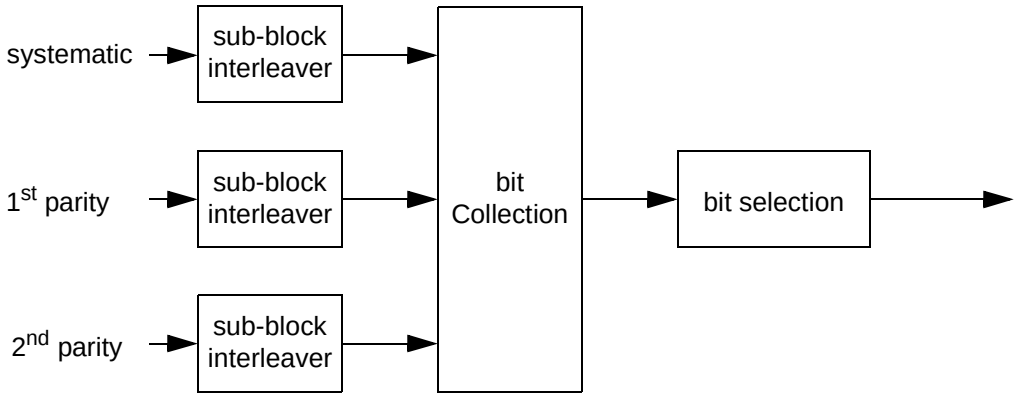


Figure 26-65. 3GLTE Rate Matching Scheme

Prior to sub-block interleaving, the number of filler bits ([NOF] field of the DEPE LTE Header) of the systematic and 1st parity streams are marked as NULL. Then the systematic, parity-1st and parity-2nd streams, are sub-block interleaved according to the 3GLTE sub-block interleaving scheme for each encoded stream, and with specific interleaving functions derived from code block size. The interleaved streams are collected into a virtual circular buffer such that the interleaved parity streams are interlaced and concatenated to the interleaved systematic stream. Bit collection is described in **Figure 26-66**

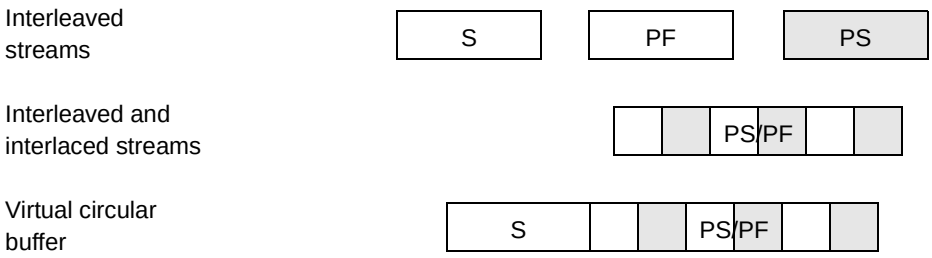


Figure 26-66. LTE Bit Collection

The bit selection scheme is executed using to the following configurations described in the DEPE LTE Header:

- [RMS]. Rate Matching Start bit. Describes the start bit index in the virtual buffer from which the output stream starts. This index equivalent to the RV index of the 3GLTE standard.
- [RMNI]. Rate Matching number of input bits. Limits the number of bits in the virtual buffer. This value is equivalent to the N_{cb} parameter of the 3GLTE standard.
- [RMNO]. Rate Matching Number of Output bits. Describes the required number of output bits after Rate Matching. Due to memory size limitation the 3GLTE Rate Matching

process in the DEPE does not support repetition. If repetition is required, the Rate Matching process should be programmed to produce a reduced circular buffer in which each encoded bit is written once. Repetition may be implemented by reading circularly from the MAPLE-B2 output circular buffer as many bits as required. This restriction limits the allowed size of the [RMNO] field to be equal or less to the value of [RMNI].

The bit selection is done according to the following pseudo code:

```
#given circular buffer
let N = 3*CBS + 12
let n0,n1 to nN-1 be the bits in the virtual circular buffer

#selecting output bits o0,o1 to oRMNO-1
k=0
j=0
while(k<[RMNO])
{
          let i = (j + [RMS])% [RMNI]
          if(ni≠ NULL)
{
          ok = ni
          k=k+1;
          }
          j = j +1
}
```

The Rate Matching circular buffer, which its size is defined by [RMNI], includes, in addition to the valid encoded bits, also dummy bits added during the Sub-block interleaving process and filler bits added prior to the encoding. These bits are not included in the DEPE output buffer. Since calculating the number of the dummy bits and filler bits is not a straightforward calculation, the value of [RMNO], which is limited by the value of [RMNI] only, may be larger than the actual number of valid encoded bits in the circular buffer. In this case the MAPLE-B2 does not output the value described in the [RMNO] field, but the actual number of valid encoded bits. The MAPLE-B2 indicates the actual number of output bits in the [RMNA] status field of the DEPE BD.

If the DEPE BD includes multiple tasks ([NOH] > 0), the [RMNA] status fields of all the tasks included in this BD are placed by MAPLE-B2 in the system memory at the address pointed by

the [BD_STS_PTR] field of the DEPE BD. **Figure 26-67** describe the data structure of the DEPE status fields in case multiple tasks are assigned in single BD:

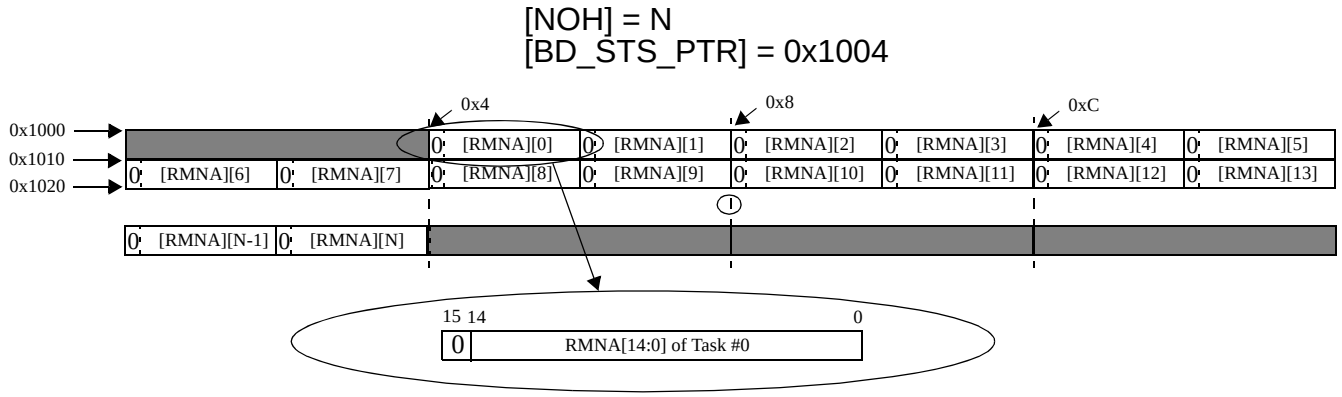


Figure 26-67. DEPE Status Data Structure for Multiple Task BD

26.4.3.4.5.2 WiMAX Processing (802.16e)

The WiMAX DEPE processing includes the following processing chain:

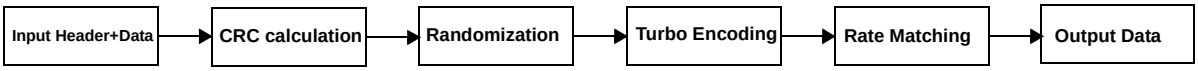


Figure 26-68. WiMAX DEPE Processing chain

The block sizes allowed in the WiMAX standards includes 17 possible types and varies from block size of 48 to block size of 4800. The Code Block size should be set in the [CBSI] field of the DEPE WiMAX Header according to the encoding described in **Table 26-264**.

26.4.3.4.5.2.1 CRC Calculation

If the CRC Disable ([CD]) bit of the DEPE WiMAX Header is set, the CRC calculation is bypassed. If the [CD] bit of the DEPE WiMAX Header is reset, the CRC is calculated and attached to the end of the data block. The CRC machine is initialized to its initial state as described in X.25 ITU-T standard for each new block encoded. The CRC polynomial used for WiMAX CRC processing is: $D^{16} + D^{12} + D^5 + 1$ hence the number of bits attached to the end of the Code Block is 16.

26.4.3.4.5.2.2 Randomization

If the Randomization Disable ([RD]) bit of the DEPE WiMAX Header is set, the Randomization processing is bypassed. If the [RD] bit is reset, the randomization processing is performed as follows:

The Randomizer is initialized to 011011100010101 for each new block encoded. The randomizer randomizes the input stream using the circuit described in **Figure 26-69**.

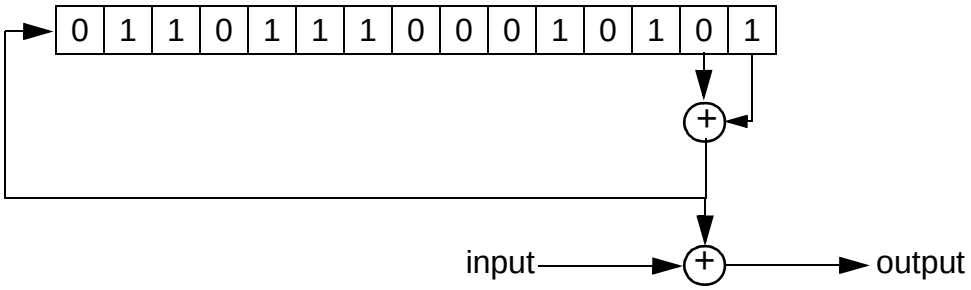


Figure 26-69. WiMax Randomizer

26.4.3.4.5.2.3 Turbo Encoding

During Turbo Encoding process the data buffer is being encoded and forwarded to the Rate Matching process. The DEPE uses a duo- binary encoder for the WiMAX encoding as described in the WiMAX standard. In addition the Turbo Encoding process uses a dedicated interleaver which interleaves the block according to the WiMAX interleaving scheme. The WiMAX interleaver scheme is selected for each block according to its size which is described in the [CBSI] field of the DEPE WiMAX Header.

26.4.3.4.5.2.4 Rate Matching

The WiMAX Rate Matching process consists of the following:

1. Interleave the sub-block of each of the six encoder output streams separately.
2. Collect the 6 interleaved streams into single virtual circular buffer.
3. Select the output bits out of the virtual circular buffer.

Figure 26-70 describes the WiMAX Rate Matching scheme:

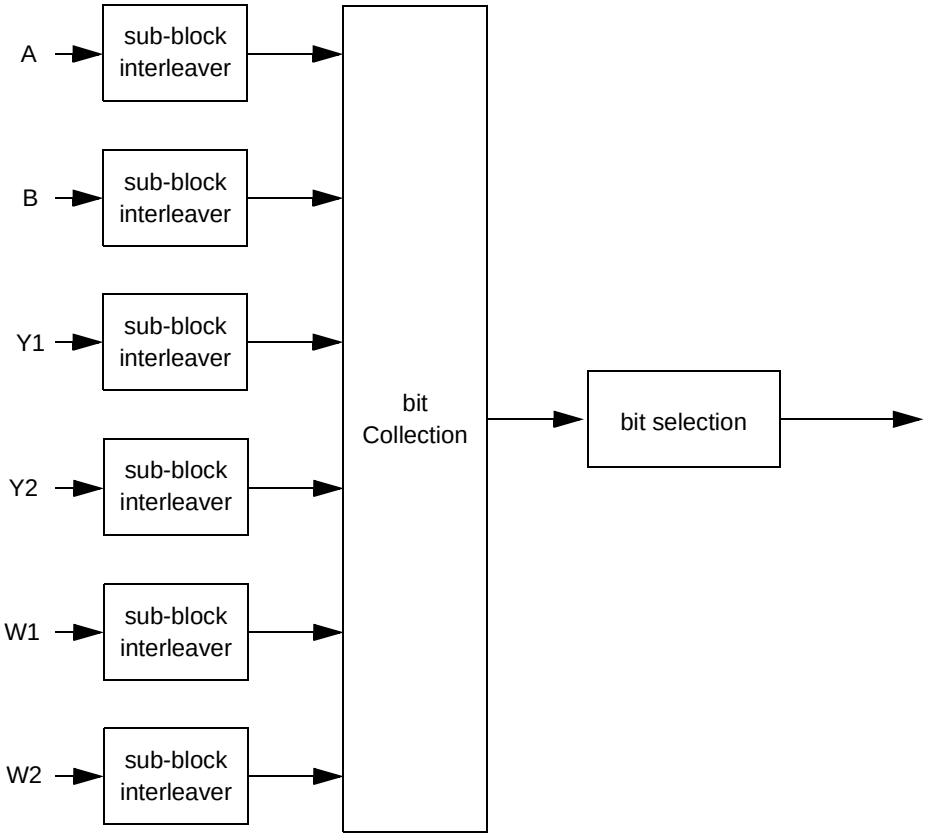


Figure 26-70. WiMAX Rate Matching Scheme

A, B, Y1, Y2, W1 and W2 are sub-block interleaved according to sub-block interleaving scheme of the WiMAX standard for each encoded stream, and with specific interleaving functions derived from code block size described in the [EBCS] field of the DEPE WiMAX Header. The interleaved stream are collected into a virtual circular buffer such that the interleaved B stream is concatenated to the interleaved A stream. The interleaved Y1 and Y2 streams are interlaced and concatenated to the interleaved B stream, and the interleaved W1 and W2 streams are interlaced and concatenated to the interlaced and interleaved Y1 and Y2 stream. Bit collection is described in **Figure 26-71**

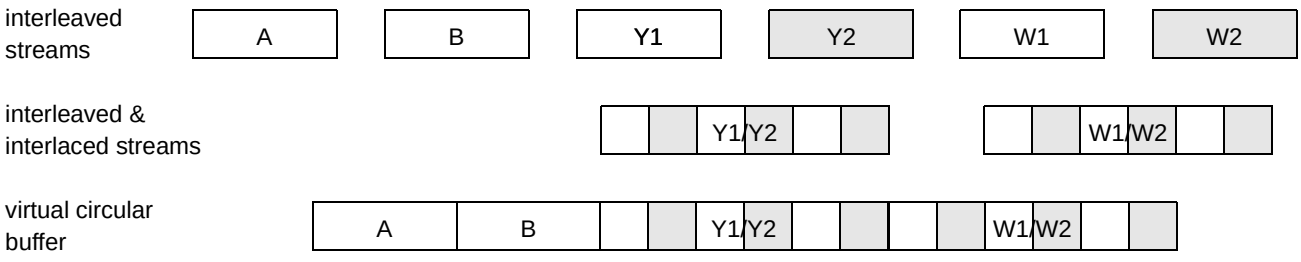


Figure 26-71. Wimax Bit Collection

The bit selection scheme is executed using to the following configurations described in the DEPE WiMAX Header:

- [RMS] - Rate Matching Start bit. Describes the start bit index in the virtual buffer from which the output stream starts.
- [RMNO] - Rate Matching Number of Output bits. Describes the required number of output bits after Rate Matching. Due to memory size limitation the WiMAX Rate Matching process in the DEPE does not support repetition. If repetition is required, the Rate Matching process should be programmed to produce a reduced circular buffer in which each encoded bit is written once. Repetition may be implemented by reading circularly from the MAPLE-B2 output buffer as many bits as required. This restriction limits the allowed size of the [RMNO] field to be equal or less to the value of 3*[Code Block Size] which is represented by the [CBSI] field of the DEPE WiMAX Header.

The output bits are selected according to the following pseudo code

```
#given circular buffer
let N = 3*CBS
let n0,n1 to nN-1 be the bits in the virtual circular buffer

#selecting output bits o0,o1 to oRMNO-1
         k=0
         j=0
while(j<[RMNO])
{
let i = (j + JH[RMS])% N
         oi = ni
         j = j +1
}
```

26.4.3.4.5.3 WiMAX Processing (802.16m)

The WiMAX 802.16m DEPE processing includes the following processing chain:

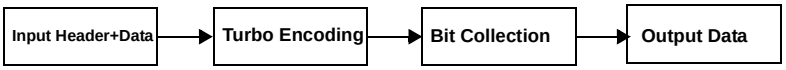


Figure 26-72. WiMAX 802.16m DEPE Processing chain

The block sizes allowed in the WiMAX standards includes 39 possible types and varies from block size of 48 to block size of 4800. The Code Block size should be set in the [CBSI] field of the DEPE WiMAX 802.16m Header according to the encoding described in **Table 26-266**.

26.4.3.4.5.3.1 Turbo Encoding

During Turbo Encoding process the data buffer is being encoded and forwarded to the Bit collection Matching process. The DEPE uses a duo- binary encoder for the WiMAX 802.16m encoding as described in the WiMAX 802.16m standard. In addition the Turbo Encoding process uses a dedicated interleaver which interleaves the block according to the WiMAX 802.16m interleaving scheme. The WiMAX 802.16m interleaver scheme is selected for each block

according to its size which is described in the [CBSI] field of the DEPE WiMAX 802.16m Header.

26.4.3.4.5.3.2 Bit Collection

The WiMAX 802.16m Rate Matching process consists of the following:

1. Interleave the sub-block of each of the six encoder output streams separately.
2. Collect the 6 interleaved streams into single virtual circular buffer.
3. Select the output bits out of the virtual circular buffer.

For WiMAX 802.16m, the DEPE supports only points #1 and #2 of the above. The output size of the bit collection scheme is always ‘Code Block Size’ * 3, hence the RMNO field of the DEPE WiMAX 802.16m field must be set to this value.

Figure 26-70 describes the DEPE WiMAX 802.16m processing scheme:

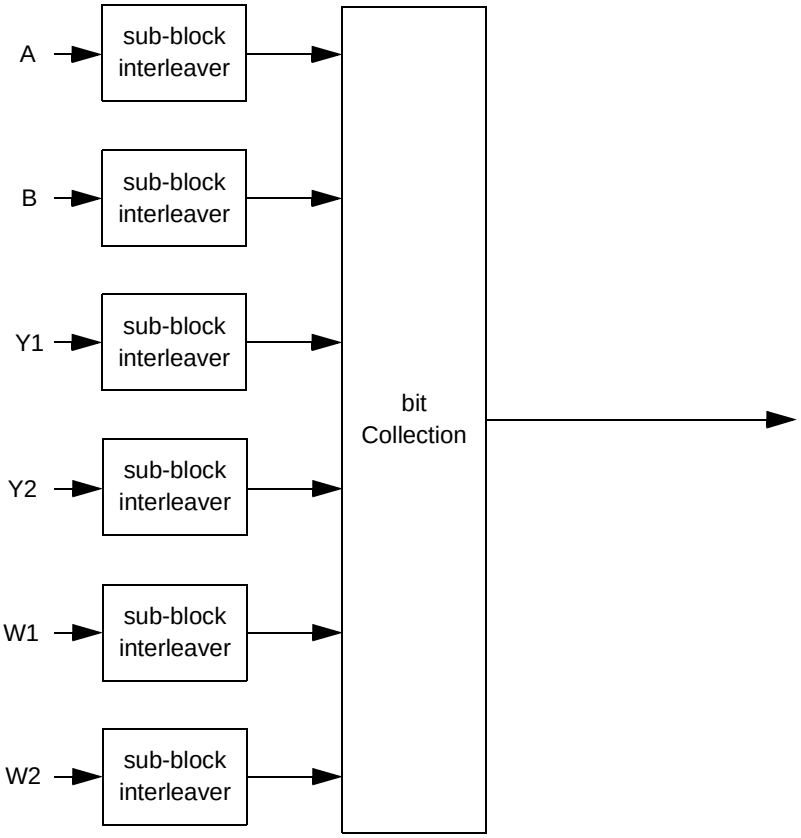


Figure 26-73. DEPE WiMAX 802.16m processing Scheme

A, B, Y1, Y2, W1 and W2 are sub-block interleaved according to sub-block interleaving scheme of the WiMAX 802.16m standard for each encoded stream, and with specific interleaving functions derived from code block size described in the [CBSI] field of the DEPE WiMAX 802.16m Header. The interleaved stream are collected into a virtual circular buffer such that the interleaved B stream is concatenated to the interleaved A stream. The interleaved Y1 and Y2

streams are interlaced and concatenated to the interleaved B stream, and the interleaved W1 and W2 streams are interlaced and concatenated to the interlaced and interleaved Y1 and Y2 stream. Bit collection is described in **Figure 26-71**

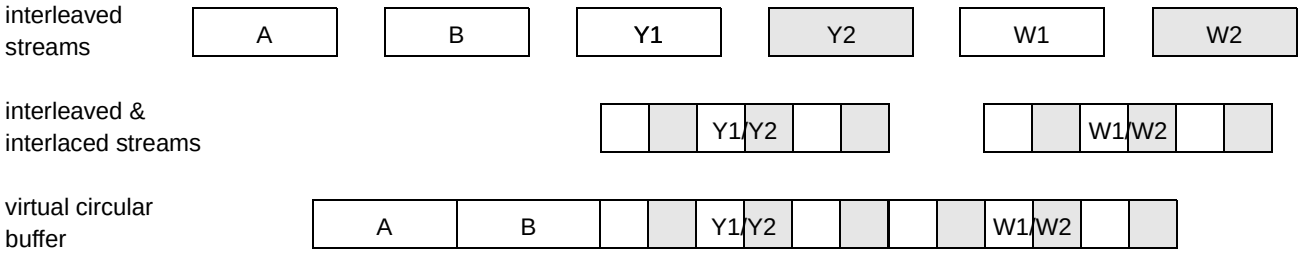


Figure 26-74. Wimax Bit Collection

The bit selection scheme is executed using to the following assumptions :

- The Rate Matching Start bit is assumed to be Zero, indicating that the start bit index of the virtual buffer from which the output stream starts is always zero.
- The Rate Matching Number of Output bits (*RMNO* field of the DEPE WiMAX 802.16m Header) is always assumed as 3*[Code Block Size] which is represented by the [CBSI] field of the DEPE WiMAX 802.16m Header.
- The Sub-Block interleaving scheme includes optional shift left of the interleaved buffers. The DEPE allows left shift execution of the Y, B and W buffers (by setting the *CLSY*, *CLSB* and *CLSW* fields of the DEPE Header respectively). As per described in the WiMAX 802.16m standard the following constraints must apply on these fields:
  - *CLSY* =1
  - *CLSB* = *CLSW*

26.4.3.4.5.4 UMTS Processing

The UMTS DEPE processing includes processing chains for FDD (**Figure 26-75**) and TDD (**Figure 26-76**).

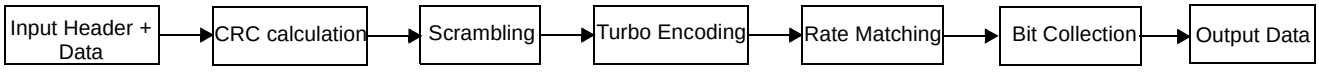


Figure 26-75. UMTS FDD DEPE Processing Chain

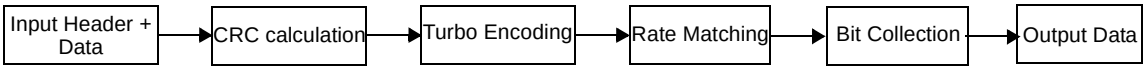


Figure 26-76. UMTS TDD DEPE Processing Chain

The block sizes allowed in the UMTS standards includes all possible sizes varies from block size of 40 to block size of 5114. The Code Block size should be set in the [CBS] field of the DEPE UMTS Header as described in **Table 26-267**.

26.4.3.4.5.4.1 CRC Calculation

If the CRC Disable ([CD]) bit of the DEPE UMTS Header is set, the CRC calculation is bypassed. If the [CD] bit of the DEPE UMTS Header is reset, the CRC is calculated and attached to the end of the data block. The UMTS standard defines CRC calculation for TB only, therefore the [CD] bit should be reset only for the cases that TB include single CB, that is, no segmentation is required. The CRC machine is reset to zero for each new block encoded. The CRC polynomial used for UMTS CRC processing is: $D^{24} + D^{23} + D^6 + D^5 + D + 1$ hence the number of bits attached to the end of the data block is 24.

26.4.3.4.5.4.2 Scrambling (FDD Only)

If the Scrambling Disable ([SD]) bit of the DEPE UMTS Header is set, the Scrambling processing is bypassed. If the [SD] bit is reset, the Scrambling processing is performed as follows:

The Scrambler is initialized to 1000000000000000 for each new block encoded. The Scrambler scrambles the input stream using the circuit described in **Figure 26-77**.

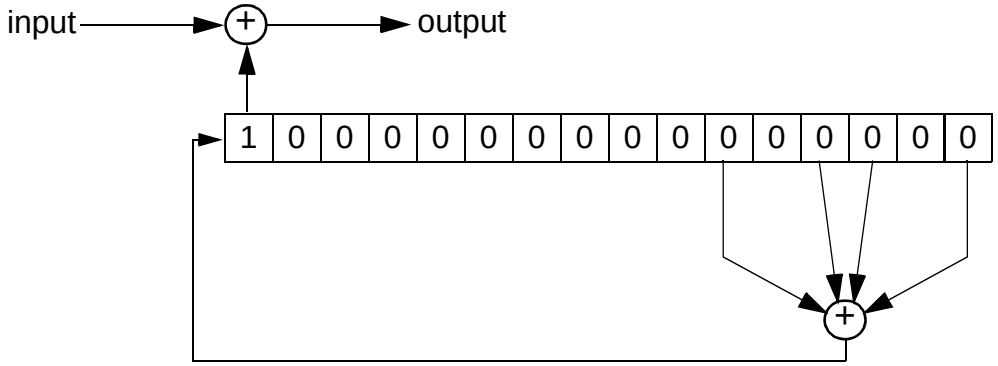


Figure 26-77. UMTS Scrambler

The UMTS standard defines Scrambling for TB only, therefore the [SD] bit should be reset only for the cases that TB include single CB, that is, no segmentation is required.

26.4.3.4.5.4.3 Turbo Encoding

During Turbo Encoding process the data buffer is being encoded and forwarded to the Rate Matching process. The DEPE uses a binary encoder for the UMTS encoding as described in the UMTS standard. In addition the Turbo Encoding process uses a dedicated interleaver which interleaves the block according to the UMTS interleaving scheme. The UMTS interleaver scheme is selected for each block according to its size which is described in the [CBS] field of the DEPE UMTS Header.

26.4.3.4.5.4.4 Rate Matching

The UMTS Rate Matching is done in parallel on the three encoded streams: systematic stream, first parity stream and second parity stream. The Rate Matching is divided into two phases: RM1 and RM2. RM1 performs puncturing on the parity streams only (should be disabled during E-DCH processing). RM2 performs either puncturing or repetition on all streams. When the [REP] bit of DEPE UMTS Header is reset, RM2 does puncturing on all streams and when the [REP] is set RM2 does repetition on all streams. The output of RM2 is written to the DEPE output buffer. UMTS Rate Matching processing is described in **Figure 26-78**:

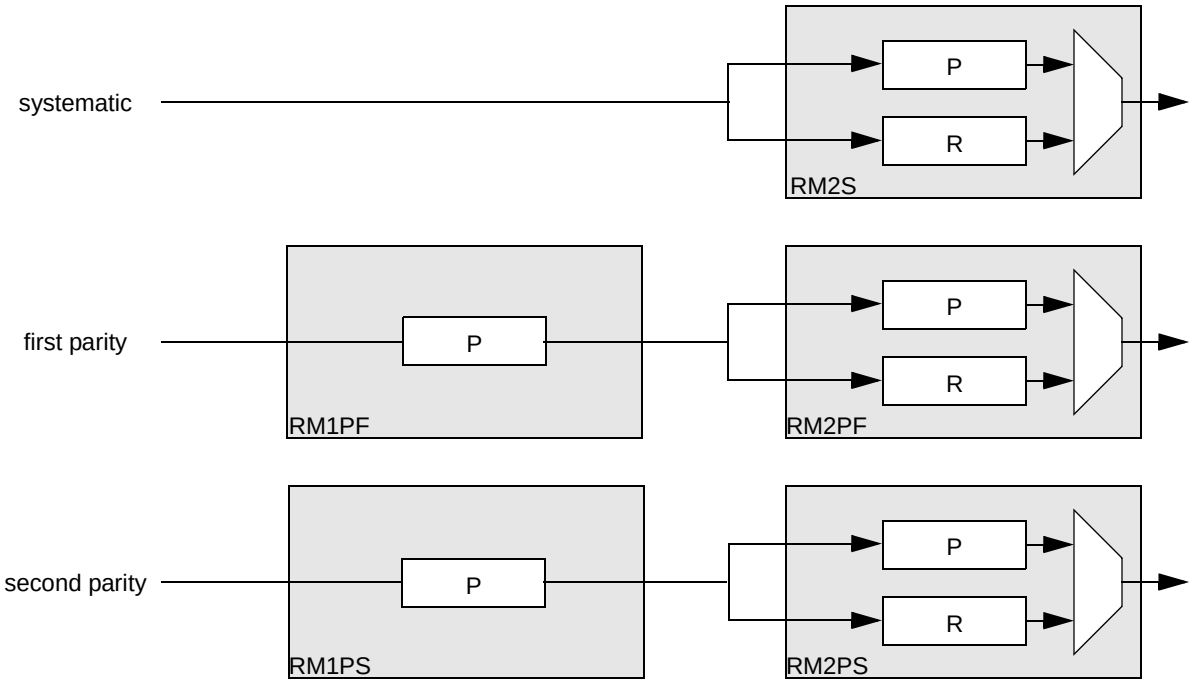


Figure 26-78. UMTS Rate Matching Processing

Note: For E-DCH processing the RM1 machine must be disabled by clearing its e_{minus} parameters to zero and configuring its e_{ini} parameters to any value greater than zero

Note: The size of the DEPE output buffer allocated for encoded data is 43200 bits. If the number of encoded bits after repetition of a single BD job (either of a single task BD or multiple tasks BD) is greater than this number, the RM2 processing should be disabled by clearing its e_{minus} parameters to 0 and configuring its e_{ini} parameters to any value greater than zero.

The following pseudo code describes the puncturing and repetition algorithms:

```
#Puncturing:

e = e_int #Init error
foreach input bit do
{
```

```
            e = e - e_minus                 #Update error
            if(e<=0)
            {
            e = e + e_plus              #Update error
            }
            else
            {
               Output current bit
            }
}

#Repetition:

e = e_ini #Init error
foreach input bit do
{
            e = e - e_minus                 #Update error
            Output current bit
            while(e <= 0)
            {
                 Output current bit
                 e = e + e_plus              #Update error
            }
}
```

Each of the Puncturing and Repetition machines are programmed by three parameters: e_{ini}, e_{plus} and e_{minus}. **Table 26-30** describes the e_{ini}, e_{plus} and e_{minus} calculations executed in the DEPE for RM1 and RM2 for each of the data streams (systematic, first parity and second parity)

Table 26-30. UMTS RM Programming

Machine[1]	e_{plus}	e_{minus}	e_{ini}
RM1PF	2*X	2**Floor*(PEM1/2)	PFEI1
RM1PS	X	*Ceil*(PEM1/2)	PSEI1
RM2S	X	SEM2	SEI2
RM2PF	2*(X - *Floor*(PEM1/2))	PFEM2	PFEI2
RM2PS	X - *C*eil(PEM1/2)	PSEM2	PSEI2

1. see **Figure 26-78**

Where:

```
X = ([CBS] + 4) * [NCB]          #[CBS] and [NCB] are defined in DEPE UMTS Header
SEM2= {[ESEM],[SEM]}                          #Defined in DEPE UMTS Header
SEI2= {[ESEI],[SEI]}                          #Defined in DEPE UMTS Header
PEM1 = {[EPEM1],[PEM1]}                       #Defined in DEPE UMTS Header
PFEM2= {[EPFEM2],[PFEM2]}                     #Defined in DEPE UMTS Header
PSEM2= {[EPSEM2],[PSEM2]}                     #Defined in DEPE UMTS Header
PFEI1= {[EPFEI1],[PFEI1]}                     #Defined in DEPE UMTS Header
PSEI1= {[EPSEI1],[PSEI1]}                     #Defined in DEPE UMTS Header
PFEI2= {[EPFEI2],[PFEI2]}                     #Defined in DEPE UMTS Header
PSEI2= {[EPSEI2],[PSEI2]}                     #Defined in DEPE UMTS Header
```

26.4.3.4.5.4.5 Bit Collection

The bit collection processing in the DEPE is controlled by the following parameters in the DEPE UMTS Header:

- *Bit Collection Mode ([BCM])*

  0 = Indicated the DEPE to collect the bits in the following order: Systematic, first parity and second parity are interlaced. This configuration should be used for E-DCH (FDD only).

  1 = Bit collection mode is done according to [BCT] configuration. This configuration should be used for E-DCH (TDD only) and HS-DSCH (FDD+TDD).
- *Bit Collection Type ([BCT])*—Valid only if [BCM] = 1

  00 = Bit collection is disabled. The DEPE outputs each of the Systematic, first parity and second parity in separate vectors. For details, see **Section 26.4.3.4.6.3**, *Separate Vectors Output Data Structure for UMTS*. This configuration should be used only if the required output buffer (after repetition) is bigger than the maximum DEPE output buffer (43200 bits).

  01, 10 or 11 = Systematic, first parity and second parity are interleaved by a rectangular interleaver with 2,4, or 6 rows respectively.
- *Number of Physical channels ([P])*—Valid only if [BCM] = 1 and [BCT] > 0.

  0 to 15 = Number of physical channels allocated for this task.

26.4.3.4.5.4.6 UMTS Transport Block (TB) Support

The DEPE is capable of supporting TB based processing and bit collection with the following limitations:

- TB size does not exceed the size of 43200 bits.
- One of the following configurations is required:
  - The required Bit Collection scheme is: Systematic, first parity and second parity are interlaced ([BCM] = 0) and the required Rate matching scheme is: Repetition ([REP] = 1)
  - The required Bit Collection scheme is: Systematic, first parity and second parity bits are interleaved by a rectangular interleaver ([BCM] = 1 and [BCT] ≠ 0).

Accumulating a whole TB in its output buffer requires inputting the DEPE successively and in order all the CB jobs related to that TB. To support that and to ease required BD configuration, the MAPLE-B2 allows configuring a single BD with single Header for full TB processing.

Configure a BD for TB processing using the standard layout with the following exceptions:

- The Input Buffer Size ([IBS]) field is ignored and should be set to zero. The MAPLE-B2 uses the [CBS] field of the DEPE Header to calculate the amount of data required for each CB in the TB.

- The Output Buffer Size ([OBS]) should be set to the size of the TB to be output from the DEPE output buffer.
- The Input Buffer Address ([IBA]) should point to the location of the TB in the system memory.
- The Transport Block Indication ([TB]) bit in the BD must be set indicating the MAPLE-B2 that the current BD is for a whole TB.

Configure a Header for TB processing using the standard layout with the following emphasis:

- The Code Block Size ([CBS]) field should be set the CB size of all segmented CBs in the TB.
- The Output Buffer Offset ([OBO]) should be set to zero assuming the output buffer of the whole TB does not require any offset.
- The Continuous Bit Collection indication should be set to zero ([CBC]=0)
- The Reset Rate Matching indication should be set to zero ([RRM]=1), indicating the DEPE to reset its internal Rate Matching machine.
- The Last Header indication should be set to zero ([LH]=0).
- The CRC Disable must be set ([CD]=1). The DEPE does not support continuous CRC calculation of a TB over several CBs.
- The Scrambling Disable must be set ([SD]=1). The DEPE does not support continuous Scrambling calculation of a TB over several CBs.
- The Number of Headers indication ([NOH]) should be set to zero indicating the DEPE that this Header represent single CB job.

The MAPLE-B2 uses the Header attached to the DEPE Transport Block BD as the Header of the first CB to launch into the DEPE. For the following CBs, use the specified modification for the CB Header into the DEPE:

1. Calculate the required Input Buffer Offset ([IBO]) according to the initial TB Input Buffer Offset, the CB size and the current index of the CB in the TB.
2. Calculate the required Output Buffer Offset ([OBO]) to concatenate the current CB output to the previous CB output of that TB.
3. Set the Continuous Bit Collection bit ([CBC]=1), indicating the DEPE that these CBs belong to the same TB.
4. Reset the ‘Reset Rate Matching’ bit ([RRM]=0), indicating the DEPE not to reset its RM machines as the current CBs belong to the same TB.

The MAPLE-B2 clears the BD [OWNER] bit only after the completion of the whole TB.

26.4.3.4.5.4.7 UMTS Processing Summary

Table 26-31 summarizes the optional configurations for the DEPE UMTS Header parameters with respect to the job type

Table 26-31. UMTS (FDD) Processing Functions Summary

Function	HS-DSCH TBS<=5090	HS-DSCH TBS>5090	E-DCH TBS<=5090 Puncturing ([REP] = 0)	E-DCH TBS<=5090 Repetition ([REP] = 1)	E-DCH TBS>5090 Puncturing ([REP] = 0)	E-DCH TBS>5090 Repetition (REP] = 1)
CRC	Optional ([CD] = 0/1)	Disabled ([CD] = 1)	Optional ([CD] = 0/1)	Optional ([CD] = 0/1)	Disabled ([CD] = 1)	Disabled ([CD] = 1)
Scrambling	Optional ([SD]= 0/1)	Disabled ([SD] = 1)	Disabled ([SD] = 1)	Disabled ([SD] = 1)	Disabled ([SD] = 1)	Disabled ([SD] = 1)
RM1	Enabled	Enabled	Disabled (E_{minus}=0 E_{ini}>0)	Disabled (E_{minus}=0 E_{ini}>0)	Disabled (E_{minus}=0 E_{ini}>0)	Disabled (E_{minus}=0 E_{ini}>0)
RM2	Enabled	Enabled	Enabled	Enabled	Enabled	Enabled
Bit Collection	Enabled ([BCM]=1 & [BCT]>0) OR Disabled (separate vectors) ([BCM]=1 & [BCT]=0)	TB BC Enabled ([BCM]=1 & [BCT]>0) OR Disabled (separate vectors) ([BCM]=1 & [BCT]=0)	Enabled ([BCM] = 0)	Enabled ([BCM] = 0) OR Disabled (separate vectors) ([BCM]=1 & [BCT]=0)	Enabled (BCM = 0)	TB BC Enabled ([BCM] = 0) OR Disabled (separate vectors) ([BCM]=1 & [BCT]=0)

26.4.3.4.5.4.8 Code Blocks Encoding Order

When the number of bits in the input stream (after CRC attachment) is greater than 5114, the input stream should be segmented to few Code Blocks (CB). In case segmentation was required, there are two ways to process the CBs using the MAPLE-B2:

1. Program the MAPLE-B2 with single Transport Block BD as described in **Section 26.4.3.4.5.4.6**, *UMTS Transport Block (TB) Support*.
2. Work with the MAPLE-B2 on a Code Block base where each BD represent a single Code Block

In case option #2 is chosen, two possible ways of working with the MAPLE-B2 are described, as follows:

- Successive And In Order Encoding. Encoding of the CB segments can be executed in the DEPE successively and in order. For such case the first segmented CB should reset the Rate Matching machine (by setting [RRM] bit of the DEPE UMTS Header) thus the initial error variables are loaded into the error variables. For the following CBs (non-first segments) related to the same Transport Block (TB), the Rate Matching machine is not reset (by resetting the [RRM] bit of the DEPE UMTS Header) such that the error variable

calculated by encoding the previous CB segment is used as initial error for the current segment.

Figure 26-79 describes an example of encoding three TBs each segmented to three CBs. When encoding TB_0CB_0 the error variables are initialized by the initial error variables programmed in the DEPE UMTS Header ([RRM] bit is set). When encoding TB_0CB_1 the initial error variables fields in the DEPE UMTS Header are ignored ([RRM] is reset) and the Error variables keep the values calculated while encoding TB_0CB_0. When encoding TB_0CB_2 the initial error variables field in the Header are ignored ([RRM] is reset) and the Error variables keep the value calculated while encoding TB_0CB_1. When encoding TB_1CB_0 the error variables are again initialized by the initial error variables programmed in the UMTS Header ([RRM] is set), and so on.

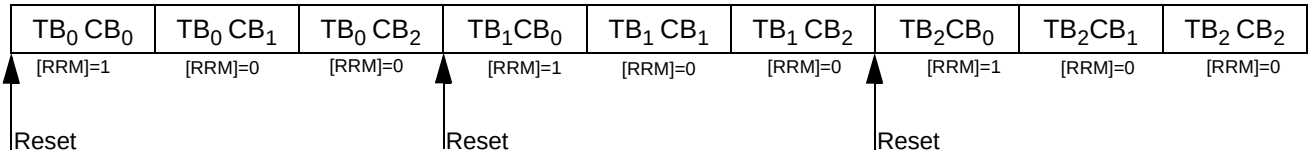


Figure 26-79. Rate Matching of Successive and In Order Code Blocks

Note: Working successively and in order in MAPLE-B2 can be promised only when working with single DEPE Buffer Descriptor's Ring thus assuring that the order of the BDs is executed as their input order into the BD Ring.

- Non-Successive or Out-of-Order Encoding. Encoding of the CB segments is executed in the DEPE either non successively or out of order. For such case the [RRM] bit should be set for all the segmented CB, and it is up to the host to calculate and initialize the error variables of the DEPE UMTS Header for each of the CB segments. **Figure 26-80** describe the required [RRM] configuration for such case:

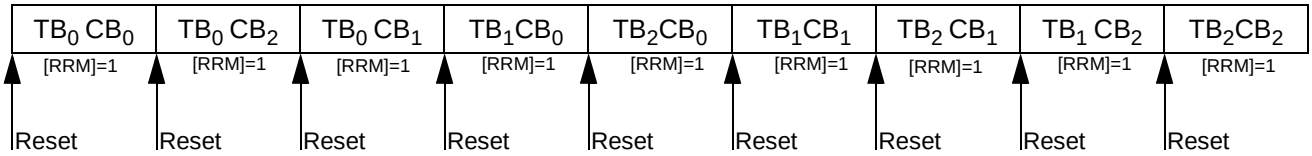


Figure 26-80. Rate Matching of Non-Successive or Out Of Order Code Blocks

Calculating the non-first segmented CB initial error variables (denoted by e^i_{ini}) may be calculated according to the following pseudo code:

```
Let e_ini be the TB initial error
Let e_minus be the TB error decrement variable
Let e_plus be the TB error increment variable
Let CBS^i be the i^th code block size

if(i==0)
{
        e^0_ini = e_ini
}
else
```

```
{
        e^i_ini = (e^(i-1)_ini -(CBS^(i-1) + 4)*e_minus)%e_plus
}
if(e^i_ini = 0) then e^i_ini = e_plus
```

26.4.3.4.6 DEPE Output Data Structure

The DEPE outputs are encoded bits after Rate Matching. The [OBA] field of DEPE BD points to the address in the system memory where the MAPLE-B2 is to write the output data from the DEPE output buffer. The address pointed by [OBA] must be 4 bytes aligned.

The [OBS] field of the DEPE BD describe the total number of 4 bytes words which should be written by the MAPLE-B2 to the system memory. Its straight forward calculation depends on the current technology:

- 3GLTE. [OBS] field is ignored as the MAPLE-B2 uses the [RMNA] status field and the [OBO] field to determine the output size. The number of 4 bytes word which the MAPLE-B2 outputs is calculated as follows:

  ```
  Output Data Size = roundup(([RMNA] + [OBO]) / 32)
  ```

- WiMAX. [OBS] field should be derived from the [RMNO] field of the DEPE WiMAX Header as follows:

  ```
  [OBS] = roundup([RMNO] / 32)
  ```

- UMTS. The output buffer stream size should be calculated using the error variables calculated according to **Table 26-30**. Following is the output buffer stream size calculation flow:

1. Calculate the number of bits which were removed or added (depend if puncturing or repetition were required) in each Rate Matching machine according to the following equation:

   $$Delta = rounddown((N * e_{minus} - e_{init} + e_{plus}) / e_{plus})$$

   Where:

   N = The number of input stream to the RM machine.

   — Following is the Delta calculation for each RM1 machine:

   $$RM1PF\_Delta = rounddown((([CBS] + 4) * e_{minus\_rm1pf} - e_{init\_rm1pf} + e_{plus\_rm1pf}) / e_{plus\_rm1pf})$$
   $$RM1PS\_Delta = rounddown((([CBS] + 4) * e_{minus\_rm1ps} - e_{init\_rm1ps} + e_{plus\_rm1ps}) / e_{plus\_rm1ps})$$

   — Following is the Delta calculation for each RM2 machine:

   $$RM2S\_Delta = rounddown((([CBS] + 4) * e_{minus\_rm2s} - e_{init\_rm2s} + e_{plus\_rm2s}) / e_{plus\_rm2s})$$
   $$RM2PF\_Delta = rounddown((([CBS] + 4 - RM1PF\_Delta) * e_{minus\_rm2pf} - e_{init\_rm2pf} + e_{plus\_rm2pf}) / e_{plus\_rm2pf})$$
   $$RM2PS\_Delta = rounddown((([CBS] + 4 - RM1PS\_Delta) * e_{minus\_rm2ps} - e_{init\_rm2ps} + e_{plus\_rm2ps}) / e_{plus\_rm2ps})$$

2. Calculate the size of each stream at the output of RM2 machine:

   — If puncturing is required ([REP] = 0):

RM2S_Out = [CBS] + 4 - RM2S_Delta
RM2PF_Out = [CBS] + 4 - RM1PF_Delta - RM2PF_Delta
RM2PS_Out = [CBS] + 4 - RM1PS_Delta - RM2PS_Delta

— If repetition is required ([REP] = 1):

RM2S_Out = [CBS] + 4 + RM2S_Delta
RM2PF_Out = [CBS] + 4 - RM1PF_Delta + RM2PF_Delta
RM2PS_Out = [CBS] + 4 - RM1PS_Delta + RM2PS_Delta

3. The total output stream size is:

Out_Stream_Size = RM2S_Out + RM2PF_Out + RM2PS_Out

Once calculated, calculating the [OBS] is done as follows:

```
[OBS] = roundup([Out_Stream_Size] / 32)
```

Due to the multiple tasks support, the output buffer offset support and the separate vectors support the total number of 4 bytes words which should be written by MAPLE-B2 may not always be calculated straightforward. For details, see **Section 26.4.3.4.6.2**, *Output Data Structure for Multiple Tasks*, **Section 26.4.3.4.6.1**, *Output Buffer Offset and Concatenate Output Support (3GLTE and UMTS)* and **Section 26.4.3.4.6.3**, *Separate Vectors Output Data Structure for UMTS*.

26.4.3.4.6.1 Output Buffer Offset and Concatenate Output Support (3GLTE and UMTS)

The DEPE supports up to 31 bits offset of the output stream with respect to the output address pointed by the Output Buffer Address ([OBA]) field of the DEPE BD. Configuring the Output Buffer Offset ([OBO]) field of the DEPE Header instructs the DEPE at which offset of the first 4 bytes word the first valid bit of the current output data buffer is to be written to. Setting the Concatenate Output Enable ([COE]) bit of the DEPE BD indicates the MAPLE-B2 that the [OBO] field of the DEPE Header has been initialized with a value greater than 0x0, hence bit combination operation is required on the first 4 bytes word of the DEPE output buffer in the system memory. **Figure 26-81** describe a 3GLTE example of the MAPLE-B2 operation when using the [OBO] and [COE] fields to concatenate in the system memory two CBs executed in different BDs.

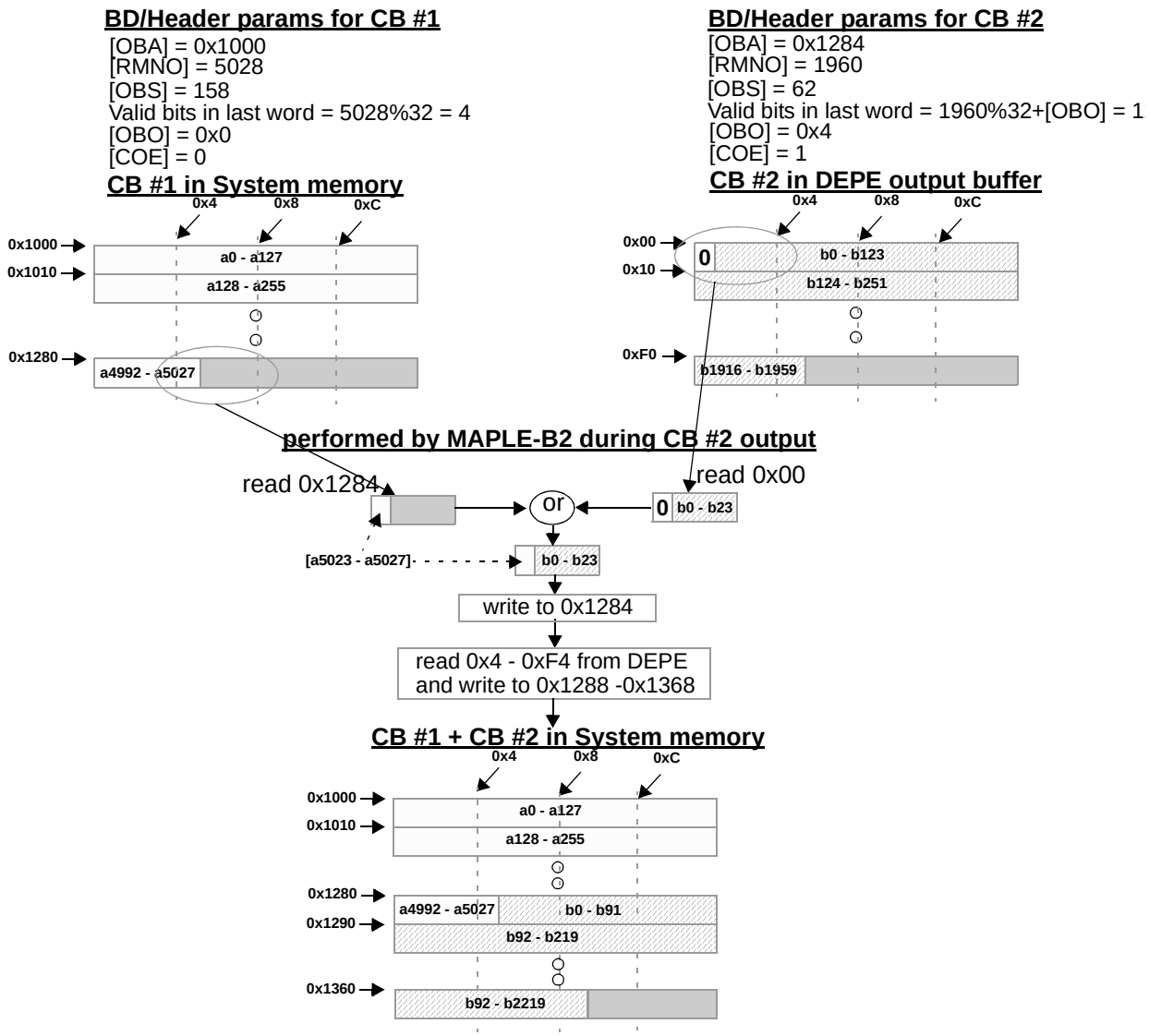


Figure 26-81. Using [OBO] and [COE] to Concatenate CB in System Memory

Supporting the Output Buffer Offset ([OBO]) along with the 32 bits resolution support of the Output Base Address pointer ([IBA]) of the DEPE BD allows writing the output stream with resolution of a single bit, thus eliminating the need to concatenate the output CBs after encoding.

When Output Buffer Offset is set to a value different than 0x0, the [OBS] field should be calculated as follows:

- 3GLTE. [OBS] field is ignored as the MAPLE-B2 uses the [RMNA] status field and the [OBO] field to determine the output size. The number of 4 bytes word which the MAPLE-B2 outputs is calculated as follows:

```
Output Data Size = roundup(([RMNA] + [OBO]) / 32)
```

- WiMAX. [OBS] field should be derived from the [RMNO] field of the DEPE WiMAX Header (see **Section 26.4.3.4.5.2**, *WiMAX Processing (802.16e)*) as follows:

```
[OBS] = roundup([RMNO]+ [OBO]/ 32)
```

- UMTS. The output buffer stream size should be calculated from the error variables calculations as described in **Section 26.4.3.4.5.4**, *UMTS Processing*. Once calculated, calculating the [OBS] is done as follows:

```
[OBS] = roundup([Out_Stream_Size] + [OBO] / 32)
```

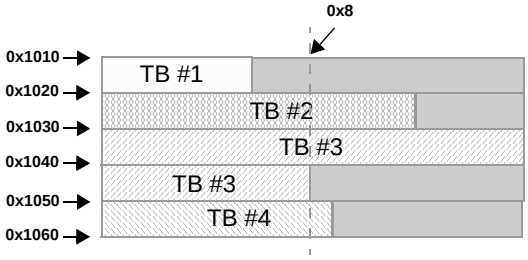
26.4.3.4.6.2 Output Data Structure for Multiple Tasks

When multiple tasks in a BD is enabled (by configuring [NOH] > 0), the MAPLE-B2 writes the output data buffers of all the tasks successively in the system memory, where the beginning each task is aligned to 128 bits line.

When [NOH] is set to a value different than 0x0, the [OBS] field should be calculated as follows:

```
[OBS] =0;
for (i=0; i < [NOH]+1; i++)
        [OBS] = [OBS]+ 4* roundup([Task[i]Output Size] / 128);
```

Figure 26-82 describe an example of the output data structure for multiple tasks BD.



[NOH] = 3
[OBA] = 0x1010
[OBS] = 20

Task #	Task Size	[OBO]
1	40	0x0
2	96	0x0
3	192	0x0
4	72	0x0

Figure 26-82. Expected Input Buffer Data Structure for Multiple Tasks BD

26.4.3.4.6.3 Separate Vectors Output Data Structure for UMTS

When processing a UMTS job in the DEPE and the Bit Collection processing is disabled (see **Section 26.4.3.4.5.4.5**, *Bit Collection*), the DEPE outputs a separate vector for each of the Rate Matching output streams: systematic, first parity and second parity. In order for the MAPLE-B2 to work in separate vectors output data structure the following parameters must be initialized:

- *[BCM], [BCT]*. DEPE UMTS Header parameters which indicate the DEPE the required Bit Collection scheme. For separate vectors output data structure (bit collection disable) they must be set to 1 and 00 respectively.
- *[SVE]*. Separate Vectors Enable indication, which indicate the MAPLE-B2 that the DEPE is working in separate vectors mode thus 3 separate vectors must be read from its output buffer and into the system memory. This bit must be set to 1 if separate vectors in DEPE is enabled.
- *[ESVO]*. Encoded Separate Vectors Offset. Indicates the MAPLE-B2 which offset to use between the vectors in the system memory. When [SVE] bit is set, the MAPLE-B2 places the systematic vector at the address pointed by the [OBA] field of the DEPE BD. The first parity vector is placed at address which equals ([OBA] + [Separate Vector Offset]) and the second parity is placed and address which equals ([OBA] + (2 * [Separate Vector Offset])). This parameter should be initialized according to the total size of each output vector so that no overlapping between the vectors occur.
- *[OBS],[PFS],[PSS]*. DEPE BD parameters which indicates the MAPLE-B2 the size of the systematic, parity first and parity second vectors size respectively. Calculating the size of each vector is described in **Section 26.4.3.4.5.4**, *UMTS Processing*. When calculating each vector size, its offset as initialized in [OBO],[PFO] and [PSO] must be considered.
- *[OBO],[PFO],[PSO]*. DEPE UMTS Header parameters which indicate the DEPE how to arrange the vectors in its output buffer. The [OBO] indicate the DEPE the offset of the first valid bit in the systematic vector with respect to the beginning of the first 32 bits word in the DEPE output buffer. The [PFO] indicate the DEPE the offset of the first valid bit of the first parity vector with respect to the first valid bit of the systematic vector. This offset must be greater than or equal to the number of systematic bits of the current code block after rate matching. The [PSO] indicate the DEPE the offset of the first valid bit of the second parity vector with respect to the first valid bit of the systematic vector. This offset must be greater than or equal to the number of systematic bits plus the number of the first parity bits.

Figure 26-83 and **Table 26-32** illustrate an example of a CB processing with Separate vectors enabled. In this example, the number of bits in each one of the systematic, first parity, and second parity vectors after Rate Matching is 144 bits. **Table 26-32** describe an example of possible configuration of the parameters of the BD:

Table 26-32. Separate Vectors Parameter Configuration Example

Parameter	BD configuration	Remarks
[BCM], BCT]	[BCM] = 1, [BCT] = 0	Bit collection is disabled
[SVE]	[SVE] = 1	Separate vectors is enabled
[ESVO]	[ESVO] = 1	Each vector is of 144 bits hence 256 bits offset prevent overlapping
[OBA]	[OBA] = 0x1000	
[OBS],[PFS],[PSS]	[OBS] = [PFS] = [PSS] = 5	Size of each vector is: roundup(144/32) = 5
[OBO],[PFO],[PSO]	[OBO] = 0 [PFO] = 256 [PSO] = 512	Each vector starts in new 128 bits line.

Figure 26-83 describe the DEPE output buffer content and the system memory content after BD completion:

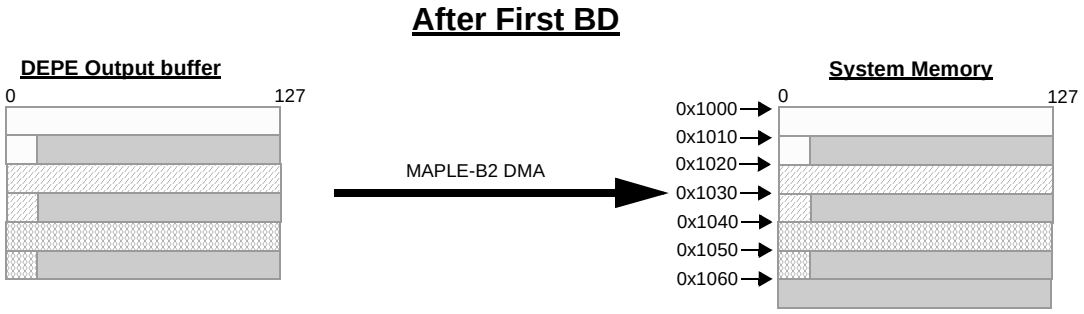


Figure 26-83. Example of DEPE Output Buffer and System Memory Content in Separate Vectors

Note: The Separate Vectors Output data structure is supported for single Headers job only, that is, is *NOH* must be set to '0' if *SVE* is set to '1'.

26.4.3.4.7 BD Status Indications

On BD completion, the MAPLE-B2 returns the RMNA status fields to the DEPE BD (relevant for 3GLTE BDs only). This field indicates the actual number of output bits in the output buffer. For details, refer to **Section 26.4.3.4.5.1**, *3GLTE Processing*

26.4.3.5 EQPE Equalization Processing Operation

The MAPLE-B2 supports Linear Equalization (LNEQ) processing (MMSE/ZF), Maximum Likelihood Equalization (MLEQ) using QRD-M algorithm, Tree Search processing (QR Decomposition is performed externally), and General Matrix Inversion operations (MIV) as described in **Section 26.2**, *MAPLE-B2 Features* using the Equalization Processing Element (EQPE).

The EQPE also includes a Frequency Domain Unit (FDU) targeted for vectors multiplication and accumulation (MAC). This unit is involved with the CONVPE processing (see **Section 26.4.3.9**, *CONVPE Convolution and Correlation Processing Operation*) by executing all the vector MAC operation required by the CONVPE. As the Equalization and Matrix Inversion related logic and the FDU related logic share the same resources, at a given time the EQPE can either work in FDU mode or in EQPE mode (Equalization and Matrix Inversion).

In general, this section describes the EQPE mode only (Equalization and Matrix Inversion), excluding **Section 26.4.3.5.7**, *CONVPE Support—FDU Processing* which describe the FDU processing as part of the CONVPE processing (see **Section 26.4.3.9**, *CONVPE Convolution and Correlation Processing Operation*).

26.4.3.5.1 EQPE Initialization Configuration

After the MAPLE-B2 initialization routine (see *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)) and before the first Buffer Descriptor is applied to the EQPE, the EQ_THRESH configuration register should be initialized: This register holds the threshold value that indicates matrix singularity. For a detailed register description see **Section 26.5.5.4.1**, *EQPE Threshold Register (EQ_THRESH)*. For additional details on matrix singularity reports see **Section 26.4.3.5.10**, *EQPE Status Indications*.

The EQPE can generate false ECC double error indications that may mask real double ECC errors from the EQPE. This can occur if the ECC logic is enabled during MAPLE-B2 initialization or if the EQPE is set to Automatic Power Switch Mode (EQPE_PWR=0) during API. To prevent this occurrence, use the following steps:

1. Disable the ECC logic of the MAPLE-B2 during API initialization (logic is disabled for the MAPLE-B2 memories).
2. Set the EQPE power mode to "power on" (EQPE_PWR=1) during the API initialization (EQPE does not power down during idle periods; impact depends on EQPE utilization).
3. Ignore the ECC double error sourced from the EQPE only by masking its enable bit in PSPICER2. This allows enabling of both the ECC logic (single errors are fixed) and the automatic power switching mode (however, real double ECC errors from the EQPE are also masked, but single ECC errors are fixed because the ECC logic is enabled).

26.4.3.5.2 EQPE BD Structure

EQPE buffer descriptors are described in detail in **Section 26.5.4.5**, *EQPE BD Structure*, on page 26-457.

26.4.3.5.3 EQPE Sub-Carrier Grid

The EQPE sub-carrier grid defines the geometry of the equalization tasks. The grid is defined by 2 parameters: *ROWS* and *COLS* configured in the EQPE BD (see **Section 26.5.4.5**, *EQPE BD Structure*), which specify the number of rows and columns in the grid. The sub-carrier grid is defined for equalization tasks only, that is, it does not exist for MIV mode.

The grid allows controlling the following:

1. The size of the processing task - *(ROWS+1)* x *(COLS+1)*. **Figure 26-84** shows an example of an EQPE job containing 36 x 12 = 432 sub-carriers. The maximal size of the grid is *ROWS*=2047 and *COLS*=15 meaning the largest job size is 32,768 sub-carriers.

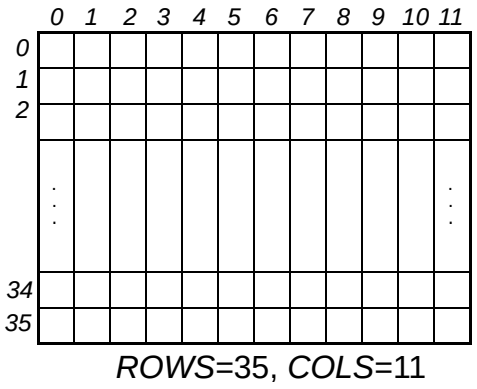


Figure 26-84. The EQPE Sub-Carrier Grid

2. Allows defining the connection between the ***H*** samples and interpolation weights (***W***s) - controlling to which sub-carrier each belong.
3. The processing order and the order of the output. The EQPE processing order is *"vertical order"* (column by column). **Figure 26-85** shows an example of an EQPE job of 36x12=432 elements and the order in which they are processed by the EQPE. The

processing order for each mode is chosen to reduce the total latency in case further processing inside the MAPLE-B2 is required.

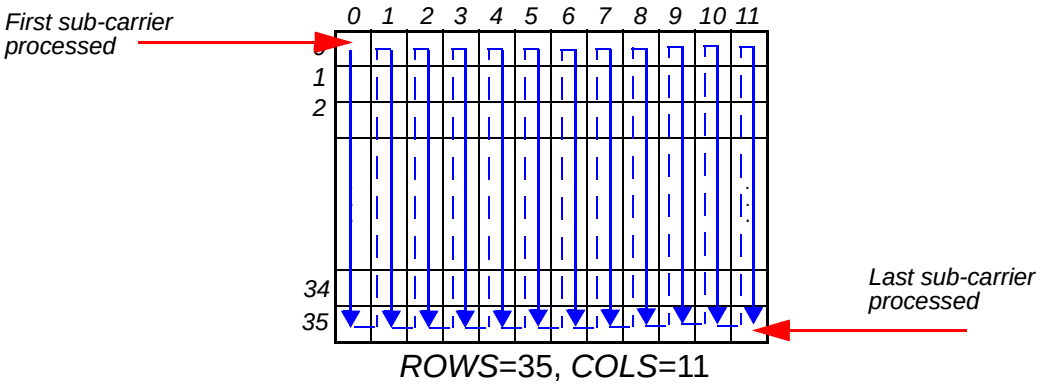


Figure 26-85. EQPE Processing Order

4. Allows defining the granularity of the S matrices and W samples see **Section 26.4.3.5.4.5**, *S Matrix* and **Section 26.4.3.5.4.7**, *W Samples*.

Note: In TS mode, if *RM_SCL_TYPE*=1 then the size of the sub-carrier grid is limited to 4096, that is (*ROWS* + 1) x (*COLS* + 1) <=4096.

26.4.3.5.4 EQPE Input Samples and Data Structures

The input data buffers required for each new EQPE job varies with respect to the current processing mode (LNEQ, MLEQ, TS and MIV) and enabling/disabling additional features related to each processing mode. **Table 26-33** summarize some of the expected input buffers type for each processing mode with respect to related features that affects input data buffers. The following sections describe the expected data structure for each of the input buffer types.

Table 26-33. Expected Input Buffers with Respect to EQPE Processing mode

EQPE Mode	Mode Parameters	Y Buffer	H0/R/M Buffer	H1 Buffer	H2 Buffer	H3 Buffer	S Buffer	W Buffer	F Buffer	C Buffer
LNEQ	INTRP=0, F_EN=1, C_EN=1	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes
	INTRP=1, F_EN=1, C_EN=0	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
	INTRP=2, F_EN=0, C_EN=1	Yes	Yes	No	No	No	Yes	No	No	Yes
	INTRP=3, F_EN=0, C_EN=0	Yes	Yes	No	No	No	Yes	No	No	No
MLEQ	INTRP=0	Yes	Yes	Yes	No	No	Yes	Yes	NA	NA
	INTRP=1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	NA	NA
	INTRP=2,3	Yes	Yes	No	No	No	Yes	No	NA	NA
TS		Yes	Yes	NA	NA	NA	Yes	NA	NA	NA
MIV		NA	Yes	NA	NA	NA	NA	NA	NA	NA
Note: This table only includes major combination examples and not all possible combinations.										

The following sections describe the expected data buffers structure of the EQPE related input buffers in each of its supported modes.

26.4.3.5.4.1 Y Samples

The following subsection define the Y sample requirements.

26.4.3.5.4.1.1 Y Samples Input Data Structure

The ***y*** samples in the Y buffers are the baseband samples from the receive antennas after FFT operation and are used only for equalization jobs (ALG=0,1,2). Each ***y*** sample is an *((Rx)* x *1)* vector for each sub-carrier in the grid or *((Lx)* x *1)* vector for TS mode[1]. Each ***y*** sample is represented by a 16-bit fractional fixed point representation of 1q15 (*V*) real and imaginary parts (16I, 16Q) and a scale factor (*Y_SCL*) such that the actual value of the sample is: $V \times 2^{Y-SCL}$.

The expected amount of Y samples for the entire job is:

— LNEQ, MLEQ: *(Rx+1)* x (*COLS+1)* x (*ROWS+1)* x *4 Bytes*
— TS: *(Lx+1)* x *(COLS +1*)x(*ROWS+1)* x *4 Bytes*.

The MAPLE-B2 expect to find *((COLS+1)* x *(Rx+1))* input Y buffers successively placed in the system memory and ordered by one of the following two ways according to the *Y_MOD* field of the MEQYBMP parameter (see also **Figure 26-86**):

1. If *(Y_MOD=0)*: Columns 0 to *COLS* of Receive antenna #0 (after FFT processing) following by columns 0 to *COLS* of Receive antenna #1...
2. If *(Y_MOD=1)*: Columns 0 of receive antennas #0 to *Rx* following by columns 1 of receive antennas #0 to *Rx*...

1. In TS mode the Y samples are actually the vector $Q^H y$ where [Q,R]=qr(H).

Figure 26-86 describe the two optional modes:

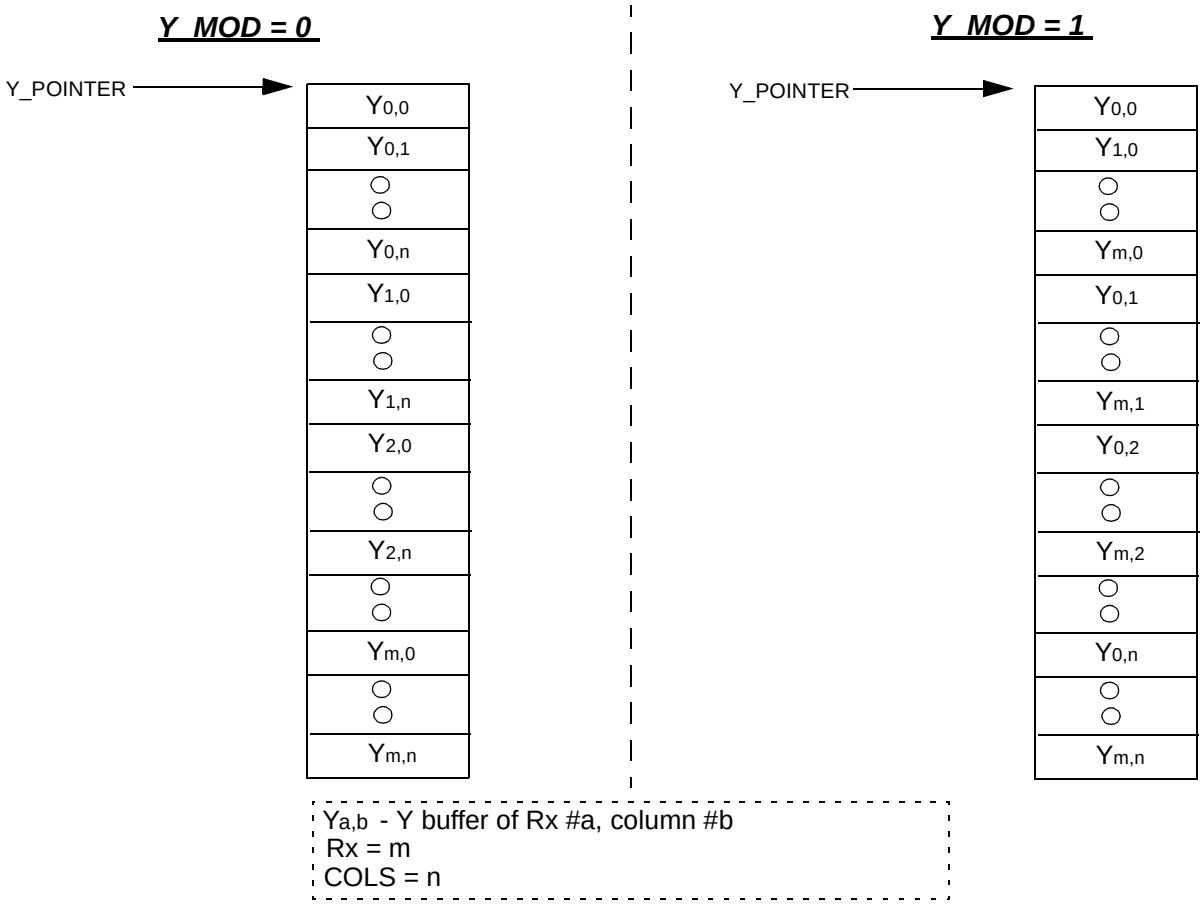


Figure 26-86. Y Buffers Order in System Memory with Respect to *MEQYBMP[Y_MOD]*

The MAPLE-B2 uses the following EQPE BD parameters to fetch the correct data for the current job:

1. *Y_POINTER* parameter - Points to the base address of all the Y buffers which includes data relevant for the current job.
2. *Y_GAP* parameter - Describe the size of each of the Y buffers located starting from *Y_POINTER.*
3. *Y_OFFSET* parameter - Describe the address offset of the relevant ***y*** samples within each Y buffer.

Following is an example which illustrates the usage of the above parameters. Assume an EQPE BD with the following parameters configuration (LNEQ): *Rx* = 3, *COLS* = 11, *ROWS* = 23, *Y_POINTER* = 0x10000000, *Y_GAP* = 1200, *Y_OFFSET* = 120. The current *Y_MOD* configuration is 0. From this configuration the MAPLE-B2 assume the following:

Table 26-34. Y Buffers address calculation example

$Y_{a,b}$	Absolute Address calculation	Size calculation (bytes)
$Y_{0,0}$	*Y_POINTER* + (4 bytes x *Y_OFFSET*) = 0x100001E0	(ROWS+1) x 4 = 96
$Y_{0,1}$	*Y_POINTER* + (4 bytes x Y_GAP) + (4 bytes x Y_OFFSET) = 0x100014A0	(ROWS+1) x 4 = 96
$Y_{0,2}$	*Y_POINTER* + (2 x 4 bytes x Y_GAP) + (4 bytes x Y_OFFSET) = 0x10002760	(ROWS+1) x 4 = 96
$Y_{0,11}$	*Y_POINTER* + (11 x 4 bytes x Y_GAP) + (4 bytes x Y_OFFSET) = 0x1000D020	(ROWS+1) x 4 = 96
$Y_{1,0}$	*Y_POINTER* + (12 x 4 bytes x Y_GAP) + (4 bytes x Y_OFFSET) = 0x1000E2E0	(ROWS+1) x 4 = 96
$Y_{1,11}$	*Y_POINTER* + (23 x 4 bytes x Y_GAP) + (4 bytes x Y_OFFSET) = 0x1001B120	(ROWS+1) x 4 = 96
$Y_{3,0}$	*Y_POINTER* + (36 x 4 bytes x Y_GAP) + (4 bytes x Y_OFFSET) = 0x1002A4E0	(ROWS+1) x 4 = 96
$Y_{3,11}$	*Y_POINTER* + (47 x 4 bytes x Y_GAP) + (4 bytes x Y_OFFSET) = 0x10037320	(ROWS+1) x 4 = 96

The expected internal data structure of each such Y buffer is described in **Figure 26-87**:

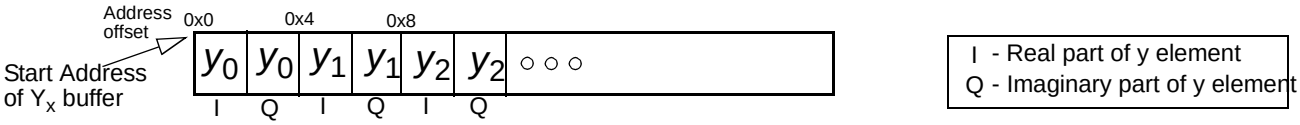


Figure 26-87. y Samples Data Structure

26.4.3.5.4.1.2 Y Scale Samples Input Data Structure

The EQPE supports a distinct Y scaling factor for each column of each Rx antenna. Therefore, there are (*Rx* + 1) x (*COLS* + 1) Y scale samples, each 1 Byte.

The expected order of the Y scale samples is as the following example (*Rx* = 3):

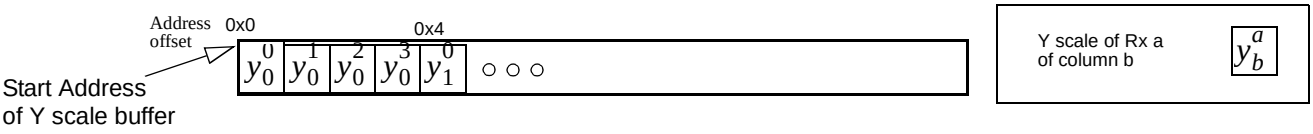


Figure 26-3. *Y* scale Samples Data Structure in Y Scale buffer for *LNEQ, MLEQ and TS Modes.*

The MAPLE-B2 expects to find the Y scale samples vector at the following address as calculated by the following EQPE BD parameters: Address = *SCL_PTR_BA* + (16 * *Y_SCL_OFFSET*).

26.4.3.5.4.2 H Samples

The ***H*** samples (for LNEQ and MLEQ) are the CE reference matrices (internal interpolation mode) or actual CE matrix (Non/External interpolation modes). Each of the ***H*** samples is

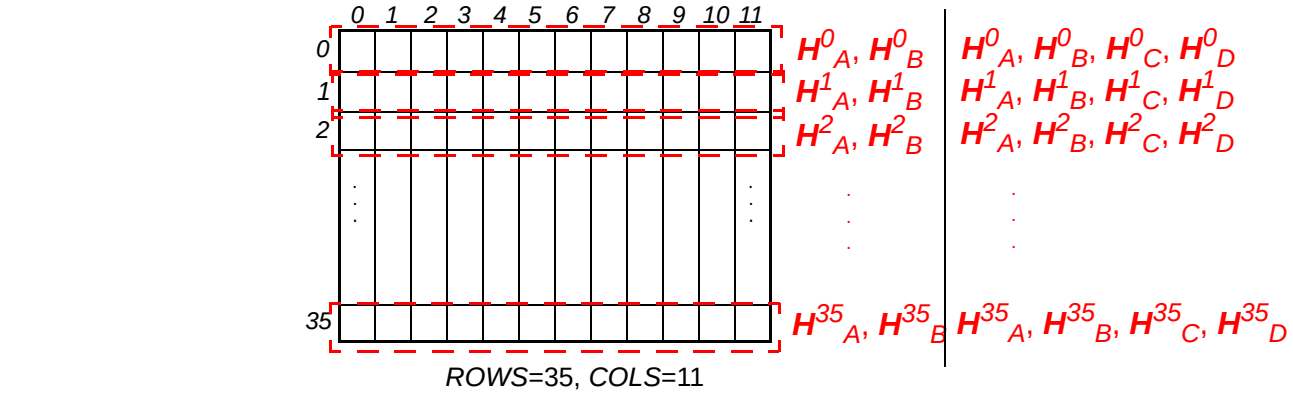
represented by a 16-bit fractional fixed point representation of 1q15 (*V*) real and imaginary parts (16I,16Q) and a scale factor (*SCL*) such that the actual value of the sample is: $V \times 2^{SCL}$.

26.4.3.5.4.2.1 H Matrices Arrangement on the Sub-Carrier Grid

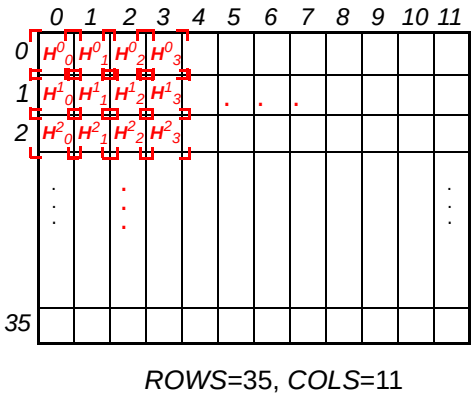
There are 4 interpolation modes supported by the EQPE that define the arrangement of the ***H*** matrices on the sub carrier grid. The interpolation mode is set in *INTRP* parameter of the EQPE BD:

- Internal Interpolation mode (II2 and II4) - In this mode there are two reference ***H*** matrices - $\boldsymbol{H}^{k}_{A}$ and $\boldsymbol{H}^{k}_{B}$ (II2) or 4 matrices $\boldsymbol{h}^{k}_{A}$,$\boldsymbol{h}^{k}_{B}$,$\boldsymbol{h}^{k}_{C}$,$\boldsymbol{h}^{k}_{D}$ (II4) shared by the entire row in the grid (*k* is the row index). The EQPE uses the matrices to calculate the interpolated CE matrices according to the interpolation weights (W samples). A distinct weight is applied for each layer for each column on the grid and may even be changed in the row dimension, thus achieving the desired interpolation strategy. For details see **Section 26.4.3.5.5.7**, *CE Matrix Interpolation*. A total of (*ROWS*+1)x2 and (*ROWS*+1)x4 ***H*** matrices are needed for II2 and II4 modes respectively.
- External Interpolation mode - In this mode there is one ***H*** matrix for each sub-carrier (it is assumed CE interpolation is done externally by the host), and no interpolation is performed by the EQPE. A total of (*ROWS*+1)x(*COLS*+1) ***H*** matrices are needed (one for each sub-carrier in the grid). The order of the matrices is according to the processing of the EQPE that is, *vertical order* (column by column).
- Non-Interpolated mode - In this mode a single ***H*** matrix is used for each row, no interpolation is performed by the EQPE. A total of (*ROWS*+1) ***H*** matrices are needed (one for each row).

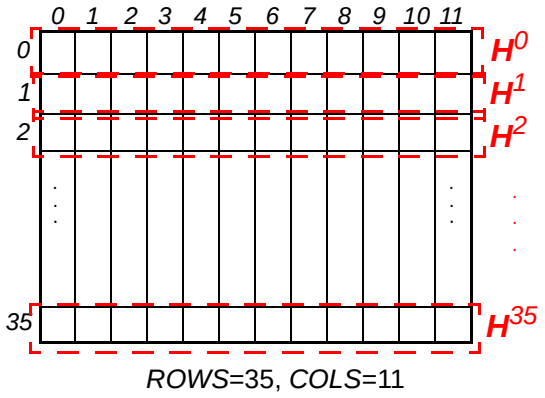
Figure 26-88 shows an example of the ***H*** matrices arrangement for all the interpolation modes.



(a) Internal Interpolation mode 2 | 4



(b) External Interpolation mode



(c) Non-Interpolated mode

Figure 26-88. *H* Matrices Arrangement on Sub-Carrier Grid

26.4.3.5.4.2.2 H Samples Input Data Structure

The H matrices input vectors number and size depends on the EQPE mode (LNEQ or MLEQ) and on the interpolation modes as described in **Table 26-33**. The H matrices arrangement on the Sub-Carrier grid is described in **Figure 26-88**.

Each H matrix dimensions is of ((*Rx+1)* * (*Lx+1)*) elements and the size of each element in the matrix is 32 bits (16I,16Q). The following table describe the total data size of ***H*** matrices input data expected with respect to the interpolation mode and other job's parameters:

Table 26-35. H Matrix Input Data Size

Interpolation Mode	H Matrix total input size calculations
Internal Interpolation 2 (II2)	((*Rx*+1) * (*Lx*+1) * 4_{bytes}) * (*ROWS*+1) * $2_{(2\ reference\ symbols\ per\ row)}$
Internal Interpolation 4 (II4)	((*Rx*+1) * (*Lx*+1) * 4_{bytes}) * (*ROWS*+1) * $4_{(4\ reference\ symbols\ per\ row)}$
External Interpolation	((*Rx*+1) * (*Lx*+1) * 4_{bytes}) * (*ROWS*+1) * (*COLS*+1)$_{(each\ column\ in\ row\ has\ CE\ symbol)}$
Non Interpolation	((*Rx*+1) * (*Lx*+1) * 4_{bytes}) * (*ROWS*+1) * $1_{(single\ reference\ symbols\ per\ row)}$

Note: In the case of *Lx*=2, the H matrix input data structure is identical to the case where *Lx*=3, therefore the above calculations should assume *Lx*=3 even if *Lx*=2.

The MAPLE-B2 expect to find each H matrix reference symbol in different vectors for the II2 and II4 interpolation modes (*H_POINTER0* to *H_POINTER3*), or in a single vector in case of external interpolation or non-interpolation modes (*H_POINTER0*).

The MAPLE-B2 uses the *CONS_H*, *H_OFFSET* and *H_GAP* fields of the EQPE BD to fetch the relevant input vectors into the EQPE: assuming $H_x{}^{i,j}$ is an element of H matrix #*x* (x = 0...*ROWS*) located in row *i* (*i* = 0...*Rx)* and column *j* (*j*=0...*Lx*) of the H matrix, the following describe the fetching order of the H matrix elements for the II2 and II4 interpolation modes (for II2 y=0,1 and for II4 y=0,1,2,3):

Table 26-36. H Elements Order in the H Vectors for II2 and II4 Interpolation Modes.

H pointer	Description	Index priority	H elements order
Hy_POINTER	Reference Symbol #y	for (*j* = 0, *j* <= *Lx*, *j*++) --for (*i* = 0, *i* <= *Rx*, *i*++) ----for (*x* = 0, *x* <= *ROWS*, *x*++) ------place $H_x{}^{i,j}$	$H_0{}^{0,0}$,$H_1{}^{0,0}$,...,$H_{ROWS}{}^{0,0}$,$H_0{}^{1,0}$,$H_1{}^{1,0}$,...,$H_{ROWS}{}^{1,0}$,...,$H_0{}^{Rx,0}$,$H_1{}^{Rx,0}$,...,$H_{ROWS}{}^{Rx,0}$,$H_0{}^{0,1}$,$H_1{}^{0,1}$,...,$H_{ROWS}{}^{0,1}$,$H_0{}^{1,1}$,$H_1{}^{1,1}$,...$H_{ROWS}{}^{1,1}$,...,$H_0{}^{Rx,Lx}$,$H_1{}^{Rx,Lx}$,...$H_{ROWS}{}^{Rx,Lx}$,

Figure 26-89 describe the expected location of each H element vector with respect to *H_CONS*, *H_GAP* and *H_OFFSET* values.

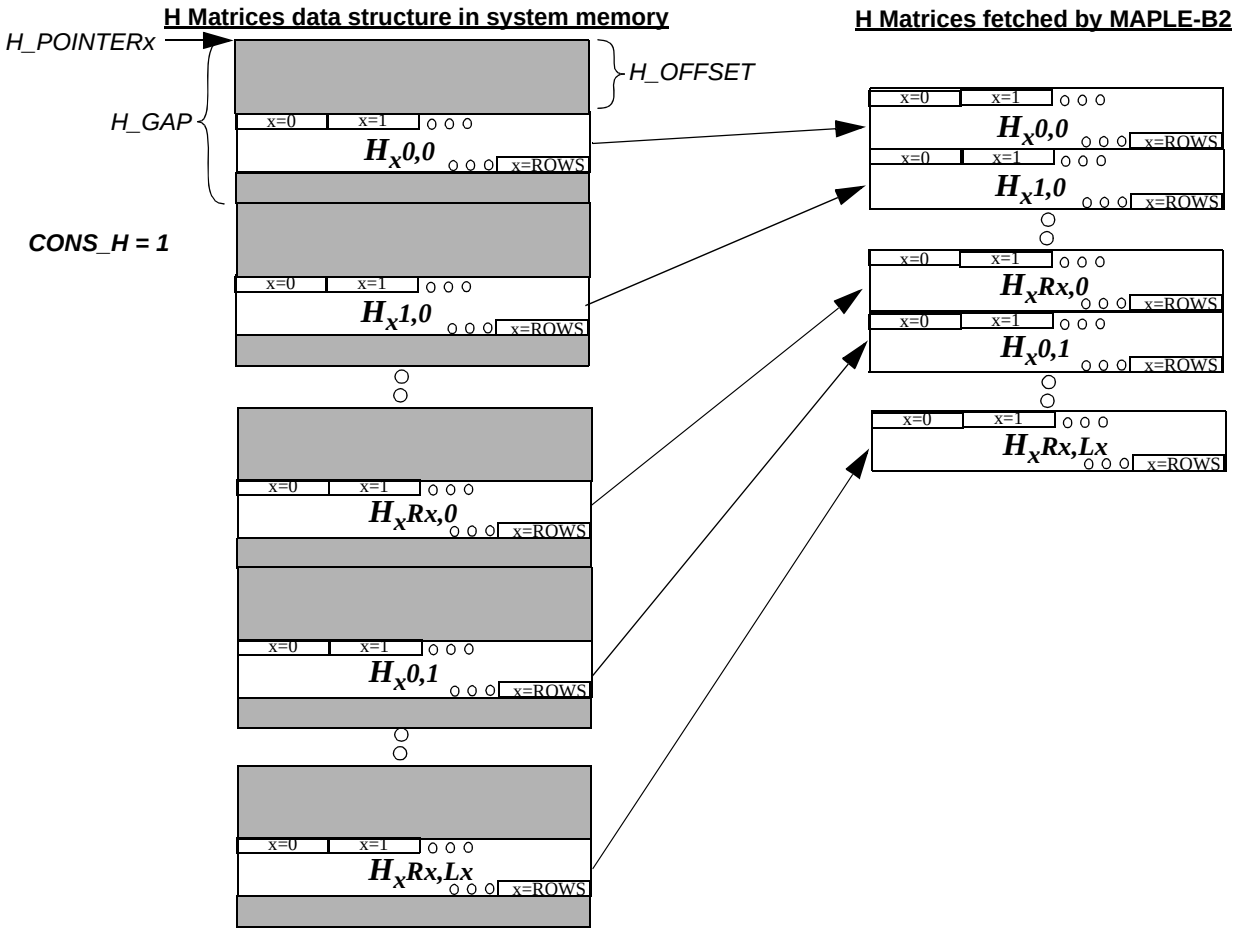


Figure 26-89. H input vectors address calculation using *H_OFFSET* and *H_GAP (CONS_H = 1)*

If the H input vectors are arranged in a consecutive manner for a given job (no gaps between the vectors), the *CONS_H* bit of the EQPE BD should be reset. In this case the MAPLE-B2 internally configures the *H_OFFSET* and *H_GAP* fields as follows:

— *H_OFFSET* = 0;
— *H_GAP = ROWS*;

Figure 26-90 describe the expected data structure in the system memory in case *CONS_H* = 0:

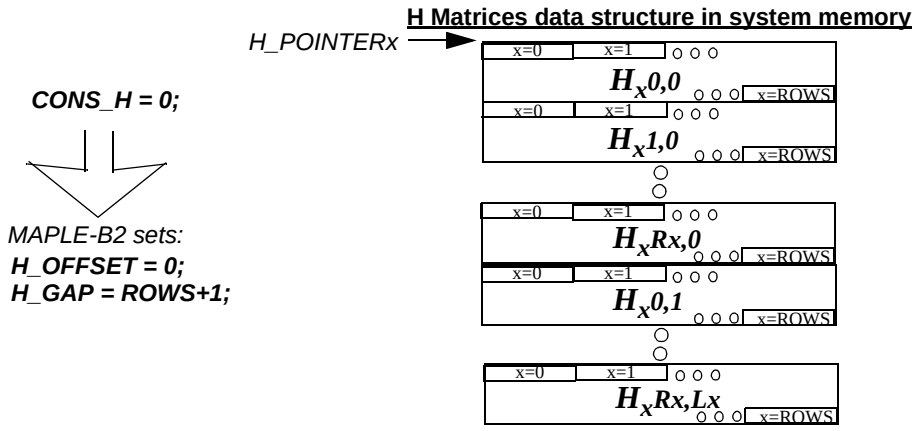


Figure 26-90. H input vectors in system memory in case *CONS_H* = 0

For the non interpolation mode there is only a single reference symbol of H matrices for the whole job. In that case the single input vector should be pointed by *H_POINTER0* and the order of the H matrix elements vectors are as described in **Table 26-36** and the expected location of each vector is the same as described in **Figure 26-89** or **Figure 26-90** (depends on *CONS_H*, *H_OFFSET* and *H_GAP* configuration).

For the external interpolation mode each column of the sub-carrier grid has a column of CE matrices. In that case, assuming $H^{y}_{x\,i,j}$ is an element of H matrix #*x* (x = 0...*ROWS*) of column #y (y= 0...*COLS*) located in row *i* (*i* = 0...*Rx)* and column *j* (*j*=0...*Lx*) of the H matrix, the following describe the fetching order of the H matrix elements vectors located in the single expected vector (*H_POINTER0*). The *CONS_H*, *H_OFFSET* and *H_GAP* fields are used to calculate the offset of the elements vectors as described in **Figure 26-89** or (depends on *CONS_H*, *H_OFFSET* and *H_GAP* configuration)

Table 26-37. H Elements Order in the H Vector for External Interpolation Mode.

H pointer	Description	Index priority	H elements order
H_POINTER0	Reference Symbols 0...*COLS*	for (*j* = 0, *j* <= *Lx*, *j*++) --for (*i* = 0, *i* <= *Rx*, *i*++) ----for (*y* = 0, *y* <= *COLS*, *y*++) ------for (*x* = 0, *x* <= *ROWS*, *x*++) --------place $H^{y}_{x\,i,j}$	$H^{0}_{0}{}^{0,0},H^{0}_{1}{}^{0,0},\ldots,H^{0}_{ROWS}{}^{0,0},H^{1}_{0}{}^{0,0},\ldots,H^{1}_{ROWS}{}^{0,0},\ldots,$ $H^{COLS}_{ROWS}{}^{0,0},H^{0}_{0}{}^{1,0},\ldots,H^{COLS}_{ROWS}{}^{Rx,0},H^{0}_{0}{}^{0,1},\ldots,$ $H^{COLS}_{ROWS}{}^{Rx,1},\ldots,H^{COLS}_{ROWS}{}^{Rx,Lx}$

Note: For external interpolation mode each new column of ***H*** samples must begin with a 16 bytes aligned address - therefore, if the current column do not end at the end of a 16 bytes line, the beginning address of the next column should jump to the next 16 bytes aligned address.

26.4.3.5.4.2.3 H Scales Input Data Structure

The number of H scale samples depends on the interpolation mode. In general, there is a distinct scaling factor for each element of the H matrix (*(Rx+1)* * (*Lx+1)* dimensions) and for each column/reference:

- For II2 there are 2*(*Rx*+1)*(*Lx*+1) H scale samples
- For II4 there are 4*(*Rx*+1)*(*Lx*+1) H scale samples
- For External Interpolation: (*COLS*+1)*(*Rx*+1)*(*Lx*+1) H scale samples
- For No-Interpolation: (*Rx*+1)*(*Lx*+1) H scale samples

All the ***H*** scale samples are of size of 1 byte each and their order in the input vector depends on the job's sub-carrier grid dimensions (*Rx*, *Lx*) and the interpolation mode. The following sections describe the vector structure for each of the relevant configurations.It is assumed that $h^{x}_{i,j}$ is the scale value of element *i,j* (*i* = 0...*Rx*, *j* = 0...*Lx*) in the H Matrix of reference symbol *x* (*x*=0,1 for *II2*, *x*=0...3 for *II4*, *x*=0...*COLS* for External Interpolation and *x* = 0 for Non-interpolation mode).

26.4.3.5.4.2.4 Internal Interpolation 2 (II2) H Scale Vector Structure

Table 26-38. *H* Scale Vector for *II2* and *Rx* x *Lx* Configuration Is 8x4[1]

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^{0}_{0,0}$	$h^{1}_{0,0}$	$h^{0}_{0,1}$	$h^{1}_{0,1}$	$h^{0}_{1,0}$	$h^{1}_{1,0}$	$h^{0}_{1,1}$	$h^{1}_{1,1}$	$h^{0}_{2,0}$	$h^{1}_{2,0}$	$h^{0}_{2,1}$	$h^{1}_{2,1}$	$h^{0}_{3,0}$	$h^{1}_{3,0}$	$h^{0}_{3,1}$	$h^{1}_{3,1}$
#2	$h^{0}_{4,0}$	$h^{1}_{4,0}$	$h^{0}_{4,1}$	$h^{1}_{4,1}$	$h^{0}_{5,0}$	$h^{1}_{5,0}$	$h^{0}_{5,1}$	$h^{1}_{5,1}$	$h^{0}_{6,0}$	$h^{1}_{6,0}$	$h^{0}_{6,1}$	$h^{1}_{6,1}$	$h^{0}_{7,0}$	$h^{1}_{7,0}$	$h^{0}_{7,1}$	$h^{1}_{7,1}$
#3	$h^{0}_{0,2}$	$h^{1}_{0,2}$	$h^{0}_{0,3}$	$h^{1}_{0,3}$	$h^{0}_{1,2}$	$h^{1}_{1,2}$	$h^{0}_{1,3}$	$h^{1}_{1,3}$	$h^{0}_{2,2}$	$h^{1}_{2,2}$	$h^{0}_{2,3}$	$h^{1}_{2,3}$	$h^{0}_{3,2}$	$h^{1}_{3,2}$	$h^{0}_{3,3}$	$h^{1}_{3,3}$
#4	$h^{0}_{4,2}$	$h^{1}_{4,2}$	$h^{0}_{4,3}$	$h^{1}_{4,3}$	$h^{0}_{5,2}$	$h^{1}_{5,2}$	$h^{0}_{5,3}$	$h^{1}_{5,3}$	$h^{0}_{6,2}$	$h^{1}_{6,2}$	$h^{0}_{6,3}$	$h^{1}_{6,3}$	$h^{0}_{7,2}$	$h^{1}_{7,2}$	$h^{0}_{7,3}$	$h^{1}_{7,3}$

1. This table is relevant for the 8x3 configuration with the only difference that the 4th layer elements become "don't-care"

Table 26-39. *H* Scale Vector for *II2* and *Rx* x *Lx* Configuration Is 8x2

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^{0}_{0,0}$	$h^{1}_{0,0}$	$h^{0}_{0,1}$	$h^{1}_{0,1}$	$h^{0}_{1,0}$	$h^{1}_{1,0}$	$h^{0}_{1,1}$	$h^{1}_{1,1}$	$h^{0}_{2,0}$	$h^{1}_{2,0}$	$h^{0}_{2,1}$	$h^{1}_{2,1}$	$h^{0}_{3,0}$	$h^{1}_{3,0}$	$h^{0}_{3,1}$	$h^{1}_{3,1}$
#2	$h^{0}_{4,0}$	$h^{1}_{4,0}$	$h^{0}_{4,1}$	$h^{1}_{4,1}$	$h^{0}_{5,0}$	$h^{1}_{5,0}$	$h^{0}_{5,1}$	$h^{1}_{5,1}$	$h^{0}_{6,0}$	$h^{1}_{6,0}$	$h^{0}_{6,1}$	$h^{1}_{6,1}$	$h^{0}_{7,0}$	$h^{1}_{7,0}$	$h^{0}_{7,1}$	$h^{1}_{7,1}$

Table 26-40. *H* Scale Vector for *II2* and *Rx* x *Lx* Configuration Is 8x1

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^{0}_{0,0}$	$h^{1}_{0,0}$	$h^{0}_{1,0}$	$h^{1}_{1,0}$	$h^{0}_{2,0}$	$h^{1}_{2,0}$	$h^{0}_{3,0}$	$h^{1}_{3,0}$	$h^{0}_{4,0}$	$h^{1}_{4,0}$	$h^{0}_{5,0}$	$h^{1}_{5,0}$	$h^{0}_{6,0}$	$h^{1}_{6,0}$	$h^{0}_{7,0}$	$h^{1}_{7,0}$

Table 26-41. *H* Scale Vector for *II2* and *Rx* x *Lx* Configuration Is 4x4

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^{0}_{0,0}$	$h^{1}_{0,0}$	$h^{0}_{0,1}$	$h^{1}_{0,1}$	$h^{0}_{1,0}$	$h^{1}_{1,0}$	$h^{0}_{1,1}$	$h^{1}_{1,1}$	$h^{0}_{2,0}$	$h^{1}_{2,0}$	$h^{0}_{2,1}$	$h^{1}_{2,1}$	$h^{0}_{3,0}$	$h^{1}_{3,0}$	$h^{0}_{3,1}$	$h^{1}_{3,1}$
#2	$h^{0}_{0,2}$	$h^{1}_{0,2}$	$h^{0}_{0,3}$	$h^{1}_{0,3}$	$h^{0}_{1,2}$	$h^{1}_{1,2}$	$h^{0}_{1,3}$	$h^{1}_{1,3}$	$h^{0}_{2,2}$	$h^{1}_{2,2}$	$h^{0}_{2,3}$	$h^{1}_{2,3}$	$h^{0}_{3,2}$	$h^{1}_{3,2}$	$h^{0}_{3,3}$	$h^{1}_{3,3}$

Table 26-42. *H* Scale Vector for *II2* and *Rx* x *Lx* Configuration Is 4x2

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^1_{0,0}$	$h^0_{0,1}$	$h^1_{0,1}$	$h^0_{1,0}$	$h^1_{1,0}$	$h^0_{1,1}$	$h^1_{1,1}$	$h^0_{2,0}$	$h^1_{2,0}$	$h^0_{2,1}$	$h^1_{2,1}$	$h^0_{3,0}$	$h^1_{3,0}$	$h^0_{3,1}$	$h^1_{3,1}$

Table 26-43. *H* Scale Vector for *II2* and *Rx* x *Lx* Configuration Is 4x1

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^1_{0,0}$	$h^0_{1,0}$	$h^1_{1,0}$	$h^0_{2,0}$	$h^1_{2,0}$	$h^0_{3,0}$	$h^1_{3,0}$								

Table 26-44. *H* Scale Vector for *II2* and *Rx* x *Lx* Configuration Is 2x2

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^1_{0,0}$	$h^0_{0,1}$	$h^1_{0,1}$	$h^0_{1,0}$	$h^1_{1,0}$	$h^0_{1,1}$	$h^1_{1,1}$								

Table 26-45. *H* Scale Vector for *II2* and *Rx* x *Lx* Configuration Is 2x1

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^1_{0,0}$	$h^0_{1,0}$	$h^1_{1,0}$												

Table 26-46. *H* Scale Vector for *II2* and *Rx* x *Lx* Configuration Is 1x1

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^1_{0,0}$														

26.4.3.5.4.2.5 Internal Interpolation 4 (II4) H Scale Vector Structure

Table 26-47. *H* Scale Vector for *II4* and *Rx* x *Lx* Configuration Is 8x2

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^1_{0,0}$	$h^2_{0,0}$	$h^3_{0,0}$	$h^0_{1,0}$	$h^1_{1,0}$	$h^2_{1,0}$	$h^3_{1,0}$	$h^0_{2,0}$	$h^1_{2,0}$	$h^2_{2,0}$	$h^3_{2,0}$	$h^0_{3,0}$	$h^1_{3,0}$	$h^2_{3,0}$	$h^3_{3,0}$
#2	$h^0_{4,0}$	$h^1_{4,0}$	$h^2_{4,0}$	$h^3_{4,0}$	$h^0_{5,0}$	$h^1_{5,0}$	$h^2_{5,0}$	$h^3_{5,0}$	$h^0_{6,0}$	$h^1_{6,0}$	$h^2_{6,0}$	$h^3_{6,0}$	$h^0_{7,0}$	$h^1_{7,0}$	$h^2_{7,0}$	$h^3_{7,0}$
#3	$h^0_{0,1}$	$h^1_{0,1}$	$h^2_{0,1}$	$h^3_{0,1}$	$h^0_{1,1}$	$h^1_{1,1}$	$h^2_{1,1}$	$h^3_{1,1}$	$h^0_{2,1}$	$h^1_{2,1}$	$h^2_{2,1}$	$h^3_{2,1}$	$h^0_{3,1}$	$h^1_{3,1}$	$h^2_{3,1}$	$h^3_{3,1}$
#4	$h^0_{4,1}$	$h^1_{4,1}$	$h^2_{4,1}$	$h^3_{4,1}$	$h^0_{5,1}$	$h^1_{5,1}$	$h^2_{5,1}$	$h^3_{5,1}$	$h^0_{6,1}$	$h^1_{6,1}$	$h^2_{6,1}$	$h^3_{6,1}$	$h^0_{7,1}$	$h^1_{7,1}$	$h^2_{7,1}$	$h^3_{7,1}$

Table 26-48. *H* Scale Vector for *II4* and *Rx* x *Lx* Configuration Is 8x1

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^1_{0,0}$	$h^2_{0,0}$	$h^3_{0,0}$	$h^0_{1,0}$	$h^1_{1,0}$	$h^2_{1,0}$	$h^3_{1,0}$	$h^0_{2,0}$	$h^1_{2,0}$	$h^2_{2,0}$	$h^3_{2,0}$	$h^0_{3,0}$	$h^1_{3,0}$	$h^2_{3,0}$	$h^3_{3,0}$
#2	$h^0_{4,0}$	$h^1_{4,0}$	$h^2_{4,0}$	$h^3_{4,0}$	$h^0_{5,0}$	$h^1_{5,0}$	$h^2_{5,0}$	$h^3_{5,0}$	$h^0_{6,0}$	$h^1_{6,0}$	$h^2_{6,0}$	$h^3_{6,0}$	$h^0_{7,0}$	$h^1_{7,0}$	$h^2_{7,0}$	$h^3_{7,0}$

Table 26-49. *H* Scale Vector for *II4* and *Rx* x *Lx* Configuration Is 4x4

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^1_{0,0}$	$h^2_{0,0}$	$h^3_{0,0}$	$h^0_{1,0}$	$h^1_{1,0}$	$h^2_{1,0}$	$h^3_{1,0}$	$h^0_{2,0}$	$h^1_{2,0}$	$h^2_{2,0}$	$h^3_{2,0}$	$h^0_{3,0}$	$h^1_{3,0}$	$h^2_{3,0}$	$h^3_{3,0}$
#2	$h^0_{0,1}$	$h^1_{0,1}$	$h^2_{0,1}$	$h^3_{0,1}$	$h^0_{1,1}$	$h^1_{1,1}$	$h^2_{1,1}$	$h^3_{1,1}$	$h^0_{2,1}$	$h^1_{2,1}$	$h^2_{2,1}$	$h^3_{2,1}$	$h^0_{3,1}$	$h^1_{3,1}$	$h^2_{3,1}$	$h^3_{3,1}$
#3	$h^0_{0,2}$	$h^1_{0,2}$	$h^2_{0,2}$	$h^3_{0,2}$	$h^0_{1,2}$	$h^1_{1,2}$	$h^2_{1,2}$	$h^3_{1,2}$	$h^0_{2,2}$	$h^1_{2,2}$	$h^2_{2,2}$	$h^3_{2,2}$	$h^0_{3,2}$	$h^1_{3,2}$	$h^2_{3,2}$	$h^3_{3,2}$
#4	$h^0_{0,3}$	$h^1_{0,3}$	$h^2_{0,3}$	$h^3_{0,3}$	$h^0_{1,3}$	$h^1_{1,3}$	$h^2_{1,3}$	$h^3_{1,3}$	$h^0_{2,3}$	$h^1_{2,3}$	$h^2_{2,3}$	$h^3_{2,3}$	$h^0_{3,3}$	$h^1_{3,3}$	$h^2_{3,3}$	$h^3_{3,3}$

Table 26-50. *H* scale vector for *II4* and *Rx* x *Lx* configuration is 4x2

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^1_{0,0}$	$h^2_{0,0}$	$h^3_{0,0}$	$h^0_{1,0}$	$h^1_{1,0}$	$h^2_{1,0}$	$h^3_{1,0}$	$h^0_{2,0}$	$h^1_{2,0}$	$h^2_{2,0}$	$h^3_{2,0}$	$h^0_{3,0}$	$h^1_{3,0}$	$h^2_{3,0}$	$h^3_{3,0}$
#2	$h^0_{0,1}$	$h^1_{0,1}$	$h^2_{0,1}$	$h^3_{0,1}$	$h^0_{1,1}$	$h^1_{1,1}$	$h^2_{1,1}$	$h^3_{1,1}$	$h^0_{2,1}$	$h^1_{2,1}$	$h^2_{2,1}$	$h^3_{2,1}$	$h^0_{3,1}$	$h^1_{3,1}$	$h^2_{3,1}$	$h^3_{3,1}$

Table 26-51. *H* Scale Vector for *II4* and *Rx* x *Lx* Configuration Is 4x1

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^1_{0,0}$	$h^2_{0,0}$	$h^3_{0,0}$	$h^0_{1,0}$	$h^1_{1,0}$	$h^2_{1,0}$	$h^3_{1,0}$	$h^0_{2,0}$	$h^1_{2,0}$	$h^2_{2,0}$	$h^3_{2,0}$	$h^0_{3,0}$	$h^1_{3,0}$	$h^2_{3,0}$	$h^3_{3,0}$

Table 26-52. *H* Scale Vector for *II4* and *Rx* x *Lx* Configuration Is 2x2

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^1_{0,0}$	$h^2_{0,0}$	$h^3_{0,0}$	$h^0_{1,0}$	$h^1_{1,0}$	$h^2_{1,0}$	$h^3_{1,0}$	$h^0_{0,1}$	$h^1_{0,1}$	$h^2_{0,1}$	$h^3_{0,1}$	$h^0_{1,1}$	$h^1_{1,1}$	$h^2_{1,1}$	$h^3_{1,1}$

Table 26-53. *H* Scale Vector for *II4* and *Rx* x *Lx* Configuration Is 2x1

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^1_{0,0}$	$h^2_{0,0}$	$h^3_{0,0}$	$h^0_{1,0}$	$h^1_{1,0}$	$h^2_{1,0}$	$h^3_{1,0}$								

Table 26-54. *H* Scale Vector for *II4* and *Rx* x *Lx* Configuration Is 1x1

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^1_{0,0}$	$h^2_{0,0}$	$h^3_{0,0}$												

26.4.3.5.4.2.6 External Interpolation H Scale Vector Structure

Table 26-55. *H* Scale Vector for *External Interpolation* and *Rx* x *Lx* Configuration Is 8x4

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^0_{0,1}$	$h^0_{1,0}$	$h^0_{1,1}$	$h^0_{2,0}$	$h^0_{2,1}$	$h^0_{3,0}$	$h^0_{3,1}$	$h^0_{4,0}$	$h^0_{4,1}$	$h^0_{5,0}$	$h^0_{5,1}$	$h^0_{6,0}$	$h^0_{6,1}$	$h^0_{7,0}$	$h^0_{7,1}$
#2	$h^0_{0,2}$	$h^0_{0,3}$	$h^0_{1,2}$	$h^0_{1,3}$	$h^0_{2,2}$	$h^0_{2,3}$	$h^0_{3,2}$	$h^0_{3,3}$	$h^0_{4,2}$	$h^0_{4,3}$	$h^0_{5,2}$	$h^0_{5,3}$	$h^0_{6,2}$	$h^0_{6,3}$	$h^0_{7,2}$	$h^0_{7,3}$
#3	$h^1_{0,0}$	$h^1_{0,1}$	$h^1_{1,0}$	$h^1_{1,1}$	$h^1_{2,0}$	$h^1_{2,1}$	$h^1_{3,0}$	$h^1_{3,1}$	$h^1_{4,0}$	$h^1_{4,1}$	$h^1_{5,0}$	$h^1_{5,1}$	$h^1_{6,0}$	$h^1_{6,1}$	$h^1_{7,0}$	$h^1_{7,1}$
#4	$h^1_{0,2}$	$h^1_{0,3}$	$h^1_{1,2}$	$h^1_{1,3}$	$h^1_{2,2}$	$h^1_{2,3}$	$h^1_{3,2}$	$h^1_{3,3}$	$h^1_{4,2}$	$h^1_{4,3}$	$h^1_{5,2}$	$h^1_{5,3}$	$h^1_{6,2}$	$h^1_{6,3}$	$h^1_{7,2}$	$h^1_{7,3}$
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
#(2*COLS*+1)	$h^{COLS}_{0,0}$	$h^{COLS}_{0,1}$	$h^{COLS}_{1,0}$	$h^{COLS}_{1,1}$	$h^{COLS}_{2,0}$	$h^{COLS}_{2,1}$	$h^{COLS}_{3,0}$	$h^{COLS}_{3,1}$	$h^{COLS}_{4,0}$	$h^{COLS}_{4,1}$	$h^{COLS}_{5,0}$	$h^{COLS}_{5,1}$	$h^{COLS}_{6,0}$	$h^{COLS}_{6,1}$	$h^{COLS}_{7,0}$	$h^{COLS}_{7,1}$
#(2*COLS*+2)	$h^{COLS}_{0,2}$	$h^{COLS}_{0,3}$	$h^{COLS}_{1,2}$	$h^{COLS}_{1,3}$	$h^{COLS}_{2,2}$	$h^{COLS}_{2,3}$	$h^{COLS}_{3,2}$	$h^{COLS}_{3,3}$	$h^{COLS}_{4,2}$	$h^{COLS}_{4,3}$	$h^{COLS}_{5,2}$	$h^{COLS}_{5,3}$	$h^{COLS}_{6,2}$	$h^{COLS}_{6,3}$	$h^{COLS}_{7,2}$	$h^{COLS}_{7,3}$

Table 26-56. *H* Scale Vector for *External Interpolation* and *Rx* x *Lx* Configuration Is 8x2

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^0_{0,1}$	$h^0_{1,0}$	$h^0_{1,1}$	$h^0_{2,0}$	$h^0_{2,1}$	$h^0_{3,0}$	$h^0_{3,1}$	$h^0_{4,0}$	$h^0_{4,1}$	$h^0_{5,0}$	$h^0_{5,1}$	$h^0_{6,0}$	$h^0_{6,1}$	$h^0_{7,0}$	$h^0_{7,1}$
#2	$h^1_{0,0}$	$h^1_{0,1}$	$h^1_{1,0}$	$h^1_{1,1}$	$h^1_{2,0}$	$h^1_{2,1}$	$h^1_{3,0}$	$h^1_{3,1}$	$h^1_{4,0}$	$h^1_{4,1}$	$h^1_{5,0}$	$h^1_{5,1}$	$h^1_{6,0}$	$h^1_{6,1}$	$h^1_{7,0}$	$h^1_{7,1}$
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
#*COLS*+1	$h^{COLS}_{0,0}$	$h^{COLS}_{0,1}$	$h^{COLS}_{1,0}$	$h^{COLS}_{1,1}$	$h^{COLS}_{2,0}$	$h^{COLS}_{2,1}$	$h^{COLS}_{3,0}$	$h^{COLS}_{3,1}$	$h^{COLS}_{4,0}$	$h^{COLS}_{4,1}$	$h^{COLS}_{5,0}$	$h^{COLS}_{5,1}$	$h^{COLS}_{6,0}$	$h^{COLS}_{6,1}$	$h^{COLS}_{7,0}$	$h^{COLS}_{7,1}$

Table 26-57. *H* Scale Vector for *External Interpolation* and *Rx* x *Lx* Configuration Is 8x1

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^0_{1,0}$	$h^0_{2,0}$	$h^0_{3,0}$	$h^0_{4,0}$	$h^0_{5,0}$	$h^0_{6,0}$	$h^0_{7,0}$	$h^1_{0,0}$	$h^1_{1,0}$	$h^1_{2,0}$	$h^1_{3,0}$	$h^1_{4,0}$	$h^1_{5,0}$	$h^1_{6,0}$	$h^1_{7,0}$
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
$\#\lceil (COLS+1)/2 \rceil$ [1]	…	…	…	…	…	…	…	…	$h^{COLS}_{0,0}$	$h^{COLS}_{1,0}$	$h^{COLS}_{2,0}$	$h^{COLS}_{3,0}$	$h^{COLS}_{4,0}$	$h^{COLS}_{5,0}$	$h^{COLS}_{6,0}$	$h^{COLS}_{7,0}$

1.The location of the last H scale vector in the last line depends on the value of *COLS*+1

Table 26-58. *H* Scale Vector for *External Interpolation* and *Rx* x *Lx* Configuration Is 4x4

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^0_{0,1}$	$h^0_{1,0}$	$h^0_{1,1}$	$h^0_{2,0}$	$h^0_{2,1}$	$h^0_{3,0}$	$h^0_{3,1}$	$h^0_{0,2}$	$h^0_{0,3}$	$h^0_{1,2}$	$h^0_{1,3}$	$h^0_{2,2}$	$h^0_{2,3}$	$h^0_{3,2}$	$h^0_{3,3}$
#2	$h^1_{0,0}$	$h^1_{0,1}$	$h^1_{1,0}$	$h^1_{1,1}$	$h^1_{2,0}$	$h^1_{2,1}$	$h^1_{3,0}$	$h^1_{3,1}$	$h^1_{0,2}$	$h^1_{0,3}$	$h^1_{1,2}$	$h^1_{1,3}$	$h^1_{2,2}$	$h^1_{2,3}$	$h^1_{3,2}$	$h^1_{3,3}$

Table 26-58. *H* Scale Vector for *External Interpolation* and *Rx* x *Lx* Configuration Is 4x4

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
#*COLS*+1	$h^{COLS}_{0,0}$	$h^{COLS}_{0,1}$	$h^{COLS}_{1,0}$	$h^{COLS}_{1,1}$	$h^{COLS}_{2,0}$	$h^{COLS}_{2,1}$	$h^{COLS}_{3,0}$	$h^{COLS}_{3,1}$	$h^{COLS}_{0,2}$	$h^{COLS}_{0,3}$	$h^{COLS}_{1,2}$	$h^{COLS}_{1,3}$	$h^{COLS}_{2,2}$	$h^{COLS}_{2,3}$	$h^{COLS}_{3,2}$	$h^{COLS}_{3,3}$

Table 26-59. *H* Scale Vector for *External Interpolation* and *Rx* x *Lx* Configuration Is 4x2

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^0_{0,1}$	$h^0_{1,0}$	$h^0_{1,1}$	$h^0_{2,0}$	$h^0_{2,1}$	$h^0_{3,0}$	$h^0_{3,1}$	$h^1_{0,0}$	$h^1_{0,1}$	$h^1_{1,0}$	$h^1_{1,1}$	$h^1_{2,0}$	$h^1_{2,1}$	$h^1_{3,0}$	$h^1_{3,1}$
$\#\lceil (COLS+1)/2 \rceil$ [1]	…	…	…	…	…	…	…	…	$h^{COLS}_{0,0}$	$h^{COLS}_{0,1}$	$h^{COLS}_{1,0}$	$h^{COLS}_{1,1}$	$h^{COLS}_{2,0}$	$h^{COLS}_{2,1}$	$h^{COLS}_{3,0}$	$h^{COLS}_{3,1}$

1.The location of the last H scale vector in the last line depends on the value of *COLS*+1

Table 26-60. *H* Scale Vector for *External Interpolation* and *Rx* x *Lx* Configuration Is 4x1

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^0_{1,0}$	$h^0_{2,0}$	$h^0_{3,0}$	$h^1_{0,0}$	$h^1_{1,0}$	$h^1_{2,0}$	$h^1_{3,0}$	…	…	…	…	…	…	…	…
$\#\lceil (COLS+1)/4 \rceil$ [1]	…	…	…	…	…	…	…	…	…	…	…	…	$h^{COLS}_{0,0}$	$h^{COLS}_{1,0}$	$h^{COLS}_{2,0}$	$h^{COLS}_{3,0}$

1.The location of the last H scale vector in the last line depends on the value of *COLS*+1

Table 26-61. *H* Scale Vector for *External Interpolation* and *Rx* x *Lx* Configuration Is 2x2

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^0_{0,1}$	$h^0_{1,0}$	$h^0_{1,1}$	$h^1_{0,0}$	$h^1_{0,1}$	$h^1_{1,0}$	$h^1_{1,1}$	…	…	…	…	…	…	…	…
$\#\lceil (COLS+1)/4 \rceil$ [1]	…	…	…	…	…	…	…	…	…	…	…	…	$h^{COLS}_{0,0}$	$h^{COLS}_{0,1}$	$h^{COLS}_{1,0}$	$h^{COLS}_{1,1}$

1.The location of the last H scale vector in the last line depends on the value of *COLS*+1

Table 26-62. *H* Scale Vector for *External Interpolation* and *Rx* x *Lx* Configuration Is 2x1

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1[1]	$h^0_{0,0}$	$h^0_{1,0}$	$h^1_{0,0}$	$h^1_{1,0}$	…	…	…	…	…	…	…	…	…	…	$h^{COLS}_{0,0}$	$h^{COLS}_{1,0}$

1.The total amount of scale values can exceed a single line and the location of the last H scale vector in the last line depends on the value of *COLS*+1

Table 26-63. *H* Scale Vector for *External Interpolation* and *Rx* x *Lx* Configuration Is 1x1

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$h^0_{0,0}$	$h^1_{0,0}$	…	…	…	…	…	…	…	…	…	…	…	…	…	$h^{COLS}_{0,0}$

26.4.3.5.4.2.7 Non Interpolation H Scale Vector Structure

The *H* scale vector structure for the non interpolation mode is identical to the data structure of the H scale vector for External interpolation mode (**Section 26.4.3.5.4.2.6**) with the only difference that for this case there is only a single column of CE matrices ($h^0_{i,j}$).

26.4.3.5.4.2.8 H Scale Vector Address

The MAPLE-B2 expects to find the H scale samples vector at the address as calculated by MAPLE-B2 using the following EQPE BD parameters:

— Address = *SCL_PTR_BA* + (16 * *H_SCL_OFFSET*).

26.4.3.5.4.3 R Samples

The ***R*** samples are the QR decomposition of H in TS mode. Each of the ***R*** samples is represented by a 16-bit fractional fixed point representation of 1q15 (*V*) real and imaginary parts (16I,16Q) and a scale factor (*SCL*) such that the actual value of the sample is: $V \times 2^{SCL}$.

R samples is relevant for the Tree Search (TS) mode only where the QR Decomposition is done externally by the host and QR results are the input for the EQPE. The MAPLE-B2 expects a single ***R*** matrix for each sub-carrier and a total of (*ROWS*+1)*(*COLS*+1) matrices are needed (one for each sub-carrier in the grid) for a TS mode job.

26.4.3.5.4.3.1 R Matrix Input Data Structure

For EQPE TS job, the MAPLE-B2 expect to find a single ***R*** matrix input vector consist of the ***R*** matrices ordered column by column starting from column #0 to column #*COLS (*defined in EQPE BD). The beginning of each ***R*** matrix column should start at 16 bytes aligned address. It means that if previous column of current job did not end at the end of line, the next column should begin at the end of next line. **Figure 26-91** describe an example of ***R*** matrix input:

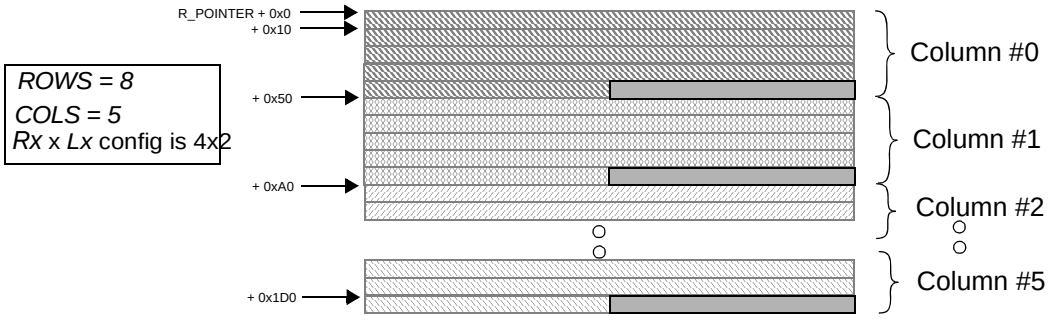


Figure 26-91. *R* Matrix Input Example

The size of each ***R*** matrix is: $(Lx+1)*(Lx+1)*4_{(bytes)}/2_{(upper\ right\ triangular\ matrix)}$.

The structure of the input ***R*** matrix column varies with respect to the *Lx* parameter of the EQPE BD. Assuming $Ry^{I}_{i,j}$ and $Ry^{Q}_{i,j}$ are the real value (*I*) and imaginary value (*Q*) of an ***R*** matrix element where:

- *y* is the index of the ***R*** matrix in the current column (y = 0...*ROWS*)
- *i/j* are the row/column indexes of the ***R*** matrix vertical/horizontal dimensions (*i* = 0...*Lx*, *j* = 0...*Lx*)

the following tables describe the ***R*** matrix structure of a single column in the vector with respect to the *Lx* parameter value:

Table 26-64. *R* Matrix Input Vector for *TS mode when Lx* = 3 (*R* Matrix Is of Size 4x4)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$R_0^I{}_{00}$		$R_0^I{}_{11}$		$R_0^I{}_{22}$		$R_0^I{}_{33}$		$R_0^I{}_{01}$		$R_0^Q{}_{01}$		$R_0^I{}_{02}$		$R_0^Q{}_{02}$	
#2	$R_0^I{}_{03}$		$R_0^Q{}_{03}$		$R_0^I{}_{12}$		$R_0^Q{}_{12}$		$R_0^I{}_{13}$		$R_0^Q{}_{13}$		$R_0^I{}_{23}$		$R_0^Q{}_{23}$	
#3	$R_1^I{}_{00}$		$R_1^I{}_{11}$		$R_1^I{}_{22}$		$R_1^I{}_{33}$		$R_1^I{}_{01}$		$R_1^Q{}_{01}$		$R_1^I{}_{02}$		$R_1^Q{}_{02}$	
#4	$R_1^I{}_{03}$		$R_1^Q{}_{03}$		$R_1^I{}_{12}$		$R_1^Q{}_{12}$		$R_1^I{}_{13}$		$R_1^Q{}_{13}$		$R_1^I{}_{23}$		$R_1^Q{}_{23}$	
...	...		...		...		...		...		...		...		...	
#(2*ROW+1)	$R_{ROWS}^I{}_{00}$		$R_{ROWS}^I{}_{11}$		$R_{ROWS}^I{}_{22}$		$R_{ROWS}^I{}_{33}$		$R_{ROWS}^I{}_{01}$		$R_{ROWS}^Q{}_{01}$		$R_{ROWS}^I{}_{02}$		$R_{ROWS}^Q{}_{02}$	
#(2*ROW+2)	$R_{ROWS}^I{}_{03}$		$R_{ROWS}^Q{}_{03}$		$R_{ROWS}^I{}_{12}$		$R_{ROWS}^Q{}_{12}$		$R_{ROWS}^I{}_{13}$		$R_{ROWS}^Q{}_{13}$		$R_{ROWS}^I{}_{23}$		$R_{ROWS}^Q{}_{23}$	

Table 26-65. *R* Matrix Input Vector for *TS Mode When Lx* = 2 (*R* Matrix Is of Size 3x3)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$R_0^I{}_{00}$		$R_0^I{}_{11}$		$R_0^I{}_{22}$		*don't care*		$R_0^I{}_{01}$		$R_0^Q{}_{01}$		$R_0^I{}_{02}$		$R_0^Q{}_{02}$	
#2	$R_0^I{}_{12}$		$R_0^Q{}_{12}$		*don't care*		*don't care*		*don't care*		*don't care*		*don't care*		*don't care*	
#3	$R_1^I{}_{00}$		$R_1^I{}_{11}$		$R_1^I{}_{22}$		*don't care*		$R_1^I{}_{01}$		$R_1^Q{}_{01}$		$R_1^I{}_{02}$		$R_1^Q{}_{02}$	
#4	$R_1^I{}_{12}$		$R_1^Q{}_{12}$		*don't care*		*don't care*		*don't care*		*don't care*		*don't care*		*don't care*	
...	...		...		...		...		...		...		...		...	
#(2*ROW+1)	$R_{ROWS}^I{}_{00}$		$R_{ROWS}^I{}_{11}$		$R_{ROWS}^I{}_{22}$		*don't care*		$R_{ROWS}^I{}_{01}$		$R_{ROWS}^Q{}_{01}$		$R_{ROWS}^I{}_{02}$		$R_{ROWS}^Q{}_{02}$	
#(2*ROW+2)	$R_{ROWS}^I{}_{12}$		$R_{ROWS}^Q{}_{12}$		*don't care*		*don't care*		*don't care*		*don't care*		*don't care*		*don't care*	

Table 26-66. *R* Matrix Input Vector for *TS Mode When Lx* = 1 (*R* Matrix Is of Size 2x2)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$R_0^I{}_{00}$		$R_0^I{}_{11}$		$R_0^I{}_{01}$		$R_0^Q{}_{01}$		$R_1^I{}_{00}$		$R_1^I{}_{11}$		$R_1^I{}_{01}$		$R_1^Q{}_{01}$	
...	...		...		...		...		...		...		...		...	
#$\lceil (ROWS+1)/2 \rceil$ [1]	...		...		...		...		$R_{ROWS}^I{}_{00}$		$R_{ROWS}^I{}_{11}$		$R_{ROWS}^I{}_{01}$		$R_{ROWS}^Q{}_{01}$	

1.If the number of rows (*ROWS*+1) in the sub-carrier grid is odd, the last R matrix is placed in the left 8 bytes of the memory line

Table 26-67. *R* Matrix Input Vector for *TS Mode When Lx* = 0 (*R* Matrix Is of Size 1x1)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$R_0^I{}_{00}$		$R_1^I{}_{00}$		$R_2^I{}_{00}$		$R_3^I{}_{00}$		$R_4^I{}_{00}$		$R_5^I{}_{00}$		$R_6^I{}_{00}$		$R_7^I{}_{00}$	
...	...		...		...		...		...		...		...		...	
#$\lceil (ROWS+1)/8 \rceil$ [1]	...		...		...		$R_{ROWS}^I{}_{00}$									

1.The location of the last R matrix in the column depends on the number of rows in the sub-carrier grid (*ROWS*)

26.4.3.5.4.3.2 R Scale Input Data Structure

There are 2 supported options for the R scaling factors, controlled by *RM_SCL_TYPE* field of the EQPE BD:

1. One shared scaling factor for all the ***R*** matrices (*RM_SCL_TYPE*=0).
2. A distinct scaling factor for each matrix. Therefore, there are (*ROWS*+1)*(*COLS*+1) ***R*** scale samples (*RM_SCL_TYPE*=1)[1].

The R scale samples are 1 Byte each and the order of the samples is as follows:

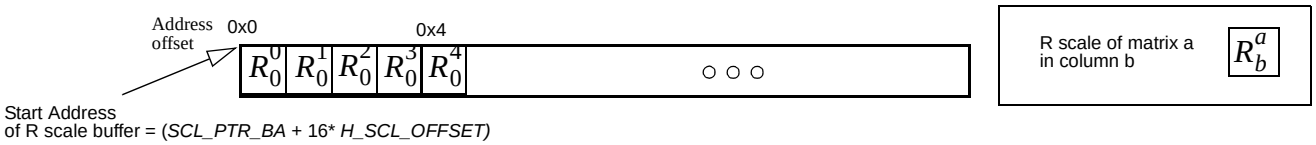


Figure 26-92. *R* Scale Samples Data Structure

The MAPLE-B2 expects to find the R scale samples vector at the following address as calculated by the following EQPE BD parameters: Address = *SCL_PTR_BA* + (16 * *H_SCL_OFFSET*).

26.4.3.5.4.4 M Samples

The ***M*** samples are the input matrix samples in MIV mode. Each of the ***M*** samples is represented by a 16-bit fractional fixed point representation of 1q15 (*V*) real and imaginary parts (16I,16Q) and a scale factor (*SCL*) such that the actual value of the sample is: $V \times 2^{SCL}$.

26.4.3.5.4.4.1 M Matrix Input Data Structure

The ***M*** matrix is relevant for the Matrix Inversion (MIV) mode only. The MAPLE-B2 expects a single vector consist of ***M*** matrices to be inverted by the EQPE. The number of matrices for a matrix inversion job is configured by the *ROWS* parameter of the EQPE BD. The dimensions of the input matrices are configured by the *Lx* parameter of the EQPE BD and the supported dimensions are: 4x4, 3x3, 2x2 and 1x1. The EQPE supports the following two types of input matrices inversion:

- Full Matrix Inversion processing - Both input and output matrices are full.
- Hermitian Matrix Inversion processing - Both input and output matrices are Hermitian

1. In this mode, the max grid size is limited to (ROW+1)*(COLS+1) ≤ 4096.

The control of the Matrix inversion type is done by the *FH* field of the EQPE BD. The size of each element in the matrix is 32 bits (16I, 16Q)[1] and the total size (in bytes) of each matrix depends on both the matrix type (full or hermitian) and its dimensions (as configured by the *Lx* parameter):

- Hermitian matrix (*FH*=0): Size = $(Lx + 1)^2 * 4_{bytes}/2$
- Full matrix (FH=1): Size = $(Lx + 1)^2 * 4_{bytes}$

The data structure of the input vector varies with respect to the matrix type and its dimensions. Assuming $M^I{}_x i,j$ and $M^Q{}_x i,j$ are the real (*I*) and imaginary (*Q*) parts of an element in the M matrix where:

- *x* is the index of the matrix (x = 0...*ROWS*).
- *i/j* are the row/column indexes of the ***M*** matrix vertical/horizontal dimensions (*i* = 0...*Lx*, *j* = 0...*Lx*)

The input data structure of the M matrix input vector is described as follows.

- Full Matrix Inversion. See **Table 26-68** through **Table 26-71**.

Table 26-68. *M* Matrix Input Vector for Full MIV *When Lx* = 3 (*M* matrix is of Size 4x4)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$M_0{}^I{}_{00}$		$M_0{}^Q{}_{00}$		$M_0{}^I{}_{01}$		$M_0{}^Q{}_{01}$		$M_0{}^I{}_{02}$		$M_0{}^Q{}_{02}$		$M_0{}^I{}_{03}$		$M_0{}^Q{}_{03}$	
#2	$M_0{}^I{}_{10}$		$M_0{}^Q{}_{10}$		$M_0{}^I{}_{11}$		$M_0{}^Q{}_{11}$		$M_0{}^I{}_{12}$		$M_0{}^Q{}_{12}$		$M_0{}^I{}_{13}$		$M_0{}^Q{}_{13}$	
#3	$M_0{}^I{}_{20}$		$M_0{}^Q{}_{20}$		$M_0{}^I{}_{21}$		$M_0{}^Q{}_{21}$		$M_0{}^I{}_{22}$		$M_0{}^Q{}_{22}$		$M_0{}^I{}_{23}$		$M_0{}^Q{}_{23}$	
#4	$M_0{}^I{}_{30}$		$M_0{}^Q{}_{30}$		$M_0{}^I{}_{31}$		$M_0{}^Q{}_{31}$		$M_0{}^I{}_{32}$		$M_0{}^Q{}_{32}$		$M_0{}^I{}_{33}$		$M_0{}^Q{}_{33}$	
#5	$M_1{}^I{}_{00}$		$M_1{}^Q{}_{00}$		$M_1{}^I{}_{01}$		$M_1{}^Q{}_{01}$		...		...		...		...	
...	...		...		...		...		...		...		...		...	
#(4*ROWS+1)	$M_{ROWS}{}^I{}_{00}$		$M_{ROWS}{}^Q{}_{00}$		$M_{ROWS}{}^I{}_{01}$		$M_{ROWS}{}^Q{}_{01}$		$M_{ROWS}{}^I{}_{02}$		$M_{ROWS}{}^Q{}_{02}$		$M_{ROWS}{}^I{}_{03}$		$M_{ROWS}{}^Q{}_{03}$	
...	...		...		...		...		...		...		...		...	
#(4*ROWS+4)	$M_{ROWS}{}^I{}_{30}$		$M_{ROWS}{}^Q{}_{30}$		$M_{ROWS}{}^I{}_{31}$		$M_{ROWS}{}^Q{}_{31}$		$M_{ROWS}{}^I{}_{32}$		$M_{ROWS}{}^Q{}_{32}$		$M_{ROWS}{}^I{}_{33}$		$M_{ROWS}{}^Q{}_{33}$	

Table 26-69. *M* Matrix Input Vector for Full MIV *When Lx* = 2 (*M* Matrix Is of Size 3x3)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$M_0{}^I{}_{00}$		$M_0{}^Q{}_{00}$		$M_0{}^I{}_{01}$		$M_0{}^Q{}_{01}$		$M_0{}^I{}_{02}$		$M_0{}^Q{}_{02}$		*don't care*		*don't care*	
#2	$M_0{}^I{}_{10}$		$M_0{}^Q{}_{10}$		$M_0{}^I{}_{11}$		$M_0{}^Q{}_{11}$		$M_0{}^I{}_{12}$		$M_0{}^Q{}_{12}$		*don't care*		*don't care*	
#3	$M_0{}^I{}_{20}$		$M_0{}^Q{}_{20}$		$M_0{}^I{}_{21}$		$M_0{}^Q{}_{21}$		$M_0{}^I{}_{22}$		$M_0{}^Q{}_{22}$		*don't care*		*don't care*	
#4	*don't care*		*don't care*		*don't care*		*don't care*		*don't care*		*don't care*		*don't care*		*don't care*	
#5	$M_1{}^I{}_{00}$		$M_1{}^Q{}_{00}$		$M_1{}^I{}_{01}$		$M_1{}^Q{}_{01}$		...		...		...		...	
...	...		...		...		...		...		...		...		...	
#(4*ROWS+1)	$M_{ROWS}{}^I{}_{00}$		$M_{ROWS}{}^Q{}_{00}$		$M_{ROWS}{}^I{}_{01}$		$M_{ROWS}{}^Q{}_{01}$		$M_{ROWS}{}^I{}_{02}$		$M_{ROWS}{}^Q{}_{02}$		*don't care*		*don't care*	
#(4*ROWS+2)	$M_{ROWS}{}^I{}_{10}$		$M_{ROWS}{}^Q{}_{10}$		$M_{ROWS}{}^I{}_{11}$		$M_{ROWS}{}^Q{}_{11}$		$M_{ROWS}{}^I{}_{12}$		$M_{ROWS}{}^Q{}_{12}$		*don't care*		*don't care*	

1. For Hermitian matrices (*FH*=0), the main diagonal is real value elements (16 bits) only.

Table 26-69. *M* Matrix Input Vector for Full MIV *When Lx* = 2 (*M* Matrix Is of Size 3x3)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#(4*ROWS+3)	$M_{ROWS}{}^{I}{}_{20}$		$M_{ROWS}{}^{Q}{}_{20}$		$M_{ROWS}{}^{I}{}_{21}$		$M_{ROWS}{}^{Q}{}_{21}$		$M_{ROWS}{}^{I}{}_{22}$		$M_{ROWS}{}^{Q}{}_{22}$		don't care		don't care	
#(4*ROWS+4)	don't care		don't care		don't care		don't care		don't care		don't care		don't care		don't care	

Table 26-70. *M* Matrix Input Vector for Full MIV *When Lx* = 1 (*M* Matrix Is of Size 2x2)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$M_0{}^{I}{}_{00}$		$M_0{}^{Q}{}_{00}$		$M_0{}^{I}{}_{01}$		$M_0{}^{Q}{}_{01}$		$M_0{}^{I}{}_{10}$		$M_0{}^{Q}{}_{10}$		$M_0{}^{I}{}_{11}$		$M_0{}^{Q}{}_{11}$	
#2	$M_1{}^{I}{}_{00}$		$M_1{}^{Q}{}_{00}$		$M_1{}^{I}{}_{01}$		$M_1{}^{Q}{}_{01}$		$M_1{}^{I}{}_{10}$		$M_1{}^{Q}{}_{10}$		$M_1{}^{I}{}_{11}$		$M_1{}^{Q}{}_{11}$	
…	…		…		…		…		…		…		…		…	
#(ROWS+1)	$M_{ROWS}{}^{I}{}_{00}$		$M_{ROWS}{}^{Q}{}_{00}$		$M_{ROWS}{}^{I}{}_{01}$		$M_{ROWS}{}^{Q}{}_{01}$		$M_{ROWS}{}^{I}{}_{10}$		$M_{ROWS}{}^{Q}{}_{10}$		$M_{ROWS}{}^{I}{}_{11}$		$M_{ROWS}{}^{Q}{}_{11}$	

Table 26-71. *M* Matrix Input Vector for Full MIV *When Lx* = 0 (*M* Matrix Is of Size 1x1)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$M_0{}^{I}{}_{00}$		$M_0{}^{Q}{}_{00}$		$M_1{}^{I}{}_{00}$		$M_1{}^{Q}{}_{00}$		$M_2{}^{I}{}_{00}$		$M_2{}^{Q}{}_{00}$		$M_3{}^{I}{}_{00}$		$M_3{}^{Q}{}_{00}$	
…	…		…		…		…		…		…		…		…	
$\#\lceil (ROWS+1)/4 \rceil$ [1]	$M_{ROWS}{}^{I}{}_{00}$		$M_{ROWS}{}^{Q}{}_{00}$													

1.The location of the last ***M*** matrix can be located in any of the 4 possible 32 bits structure of the last line (depend on value of *ROWS*.

- Hermitian Matrix Inversion. See **Table 26-72** through **Table 26-75**.

Table 26-72. *M* Matrix Input Vector for Hermitian MIV *When Lx* = 4 (*M* Matrix Is of Size 4x4)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$M_0{}^{I}{}_{00}$		$M_0{}^{I}{}_{11}$		$M_0{}^{I}{}_{22}$		$M_0{}^{I}{}_{33}$		$M_0{}^{I}{}_{01}$		$M_0{}^{Q}{}_{01}$		$M_0{}^{I}{}_{02}$		$M_0{}^{Q}{}_{02}$	
#2	$M_0{}^{I}{}_{03}$		$M_0{}^{Q}{}_{03}$		$M_0{}^{I}{}_{12}$		$M_0{}^{Q}{}_{12}$		$M_0{}^{I}{}_{13}$		$M_0{}^{Q}{}_{13}$		$M_0{}^{I}{}_{23}$		$M_0{}^{Q}{}_{23}$	
#3	$M_1{}^{I}{}_{00}$		$M_1{}^{I}{}_{11}$		$M_1{}^{I}{}_{22}$		$M_1{}^{I}{}_{33}$		$M_1{}^{I}{}_{01}$		$M_1{}^{Q}{}_{01}$		$M_1{}^{I}{}_{02}$		$M_1{}^{Q}{}_{02}$	
#4	$M_1{}^{I}{}_{03}$		$M_1{}^{Q}{}_{03}$		$M_1{}^{I}{}_{12}$		$M_1{}^{Q}{}_{12}$		$M_1{}^{I}{}_{13}$		$M_1{}^{Q}{}_{13}$		$M_1{}^{I}{}_{23}$		$M_1{}^{Q}{}_{23}$	
…	…		…		…		…		…		…		…		…	
#(2*ROWS+1)	$M_{ROWS}{}^{I}{}_{00}$		$M_{ROWS}{}^{I}{}_{11}$		$M_{ROWS}{}^{I}{}_{22}$		$M_{ROWS}{}^{I}{}_{33}$		$M_{ROWS}{}^{I}{}_{01}$		$M_{ROWS}{}^{Q}{}_{01}$		$M_{ROWS}{}^{I}{}_{02}$		$M_{ROWS}{}^{Q}{}_{02}$	
#(2*ROWS+2)	$M_{ROWS}{}^{I}{}_{03}$		$M_{ROWS}{}^{Q}{}_{03}$		$M_{ROWS}{}^{I}{}_{12}$		$M_{ROWS}{}^{Q}{}_{12}$		$M_{ROWS}{}^{I}{}_{13}$		$M_{ROWS}{}^{Q}{}_{13}$		$M_{ROWS}{}^{I}{}_{23}$		$M_{ROWS}{}^{Q}{}_{23}$	

Table 26-73. *M* Matrix Input Vector for Hermitian MIV *When Lx* = 3 (*M* Matrix Is of Size 3x3)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$M_0{}^{I}{}_{00}$		$M_0{}^{I}{}_{11}$		$M_0{}^{I}{}_{22}$		don't care		$M_0{}^{I}{}_{01}$		$M_0{}^{Q}{}_{01}$		$M_0{}^{I}{}_{02}$		$M_0{}^{Q}{}_{02}$	
#2	$M_0{}^{I}{}_{12}$		$M_0{}^{Q}{}_{12}$		don't care		don't care		don't care		don't care		don't care		don't care	
#3	$M_1{}^{I}{}_{00}$		$M_1{}^{I}{}_{11}$		$M_1{}^{I}{}_{22}$		don't care		$M_1{}^{I}{}_{01}$		$M_1{}^{Q}{}_{01}$		$M_1{}^{I}{}_{02}$		$M_1{}^{Q}{}_{02}$	
#4	$M_1{}^{I}{}_{12}$		$M_1{}^{Q}{}_{12}$		don't care		don't care		don't care		don't care		don't care		don't care	
…	…		…		…		…		…		…		…		…	
#(2*ROWS+1)	$M_{ROWS}{}^{I}{}_{00}$		$M_{ROWS}{}^{I}{}_{11}$		$M_{ROWS}{}^{I}{}_{22}$		don't care		$M_{ROWS}{}^{I}{}_{01}$		$M_{ROWS}{}^{Q}{}_{01}$		$M_{ROWS}{}^{I}{}_{02}$		$M_{ROWS}{}^{Q}{}_{02}$	
#(2*ROWS+2)	$M_{ROWS}{}^{I}{}_{12}$		$M_{ROWS}{}^{Q}{}_{12}$		don't care		don't care		don't care		don't care		don't care		don't care	

Table 26-74. *M* Matrix Input Vector for Hermitian MIV *When Lx* = 2 (*M* Matrix Is of Size 2x2)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$M_0{}^I{}_{00}$		$M_0{}^I{}_{11}$		$M_0{}^I{}_{01}$		$M_0{}^Q{}_{01}$		$M_1{}^I{}_{00}$		$M_1{}^I{}_{11}$		$M_1{}^I{}_{01}$		$M_1{}^Q{}_{01}$	
…	…		…		…		…		…		…		…		…	
$\#\lceil (ROWS+1)/2 \rceil$ [1]	…		…		…		…		$M_{ROWS}{}^I{}_{00}$		$M_{ROWS}{}^I{}_{11}$		$M_{ROWS}{}^I{}_{01}$		$M_{ROWS}{}^Q{}_{01}$	

1.If the number of rows (*ROWS*+1) in the sub-carrier grid is even, the last ***M*** matrix is placed in the left 8 bytes of the memory line

Table 26-75. *M* Matrix Input Vector for Hermitian MIV *When Lx* = 1 (*M* Matrix Is of Size 1x1)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	$M_0{}^I{}_{00}$		$M_1{}^I{}_{00}$		$M_2{}^I{}_{00}$		$M_3{}^I{}_{00}$		…		…		…		…	
$\#\lceil (ROWS+1)/8 \rceil$ [1]	…		…		…		…		…		…		…		$M_{ROWS}{}^I{}_{00}$	

1.The last ***M*** matrix can be located in any of the 8 possible 16-bit structure of the last line (depending on the value of *ROWS*).

The MAPLE-B2 expects to find ***M*** matrix vector at the address pointed by *M_POINTER* during EQPE MIV jobs.

26.4.3.5.4.4.2 M Scale Input Data Structure

There are two supported options for the ***M*** scaling factors controlled by *RM_SCL_TYPE* field of the EQPE BD:

- One shared scaling factor for all the ***M*** matrices (*RM_SCL_TYPE*=0).
- A distinct scaling factor for each matrix. Therefore, there are (*ROWS*+1) ***M*** scale samples (*RM_SCL_TYPE*=1).

The ***M*** scale samples are 1 Byte each and the order of the samples is as follows:

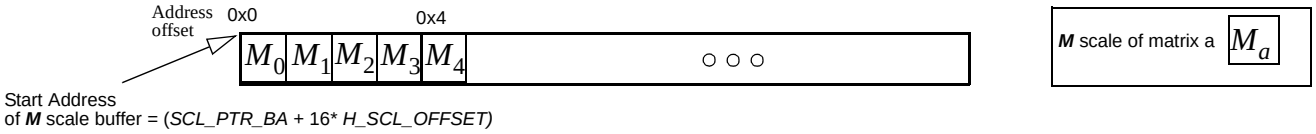


Figure 26-93. *M* Scale Samples Data Structure

If (*RM_SCL_TYPE*=0) then the M scale vector includes single scaling value only.

The MAPLE-B2 expects to find the ***M*** scale samples vector at the following address as calculated by the following EQPE BD parameters: Address = *SCL_PTR_BA* + (16 * *H_SCL_OFFSET*).

26.4.3.5.4.5 S Matrix

The S buffer contains the ***S*** matrices samples (***S*** = $\boldsymbol{C}_n^{-1}$). Each ***S*** sample is represented by a 16-bit fractional fixed point representation of 1q15 (*V*) either real (16I) or complex (16I,16Q) and a scale factor (*S_SCL*) such that the actual value of the sample is: $V \times 2^{S-SCL}$.

26.4.3.5.4.5.1 S matrices Arrangement on the Sub-Carrier Grid

2 parameters control the arrangement of the ***S*** matrices in the S buffer:

- *S_ROW*. The number of rows that share the same ***S*** matrix. If *S_ROW*=0 the all the rows in the job share the same ***S*** matrix.
- *S_COL*. The number of columns that share the same ***S*** matrix. If *S_COL*=0, then all the columns share the same ***S*** matrix.

Figure 26-94 illustrates few examples of using the S_ROW and S_COL on the sub-carrier grid:

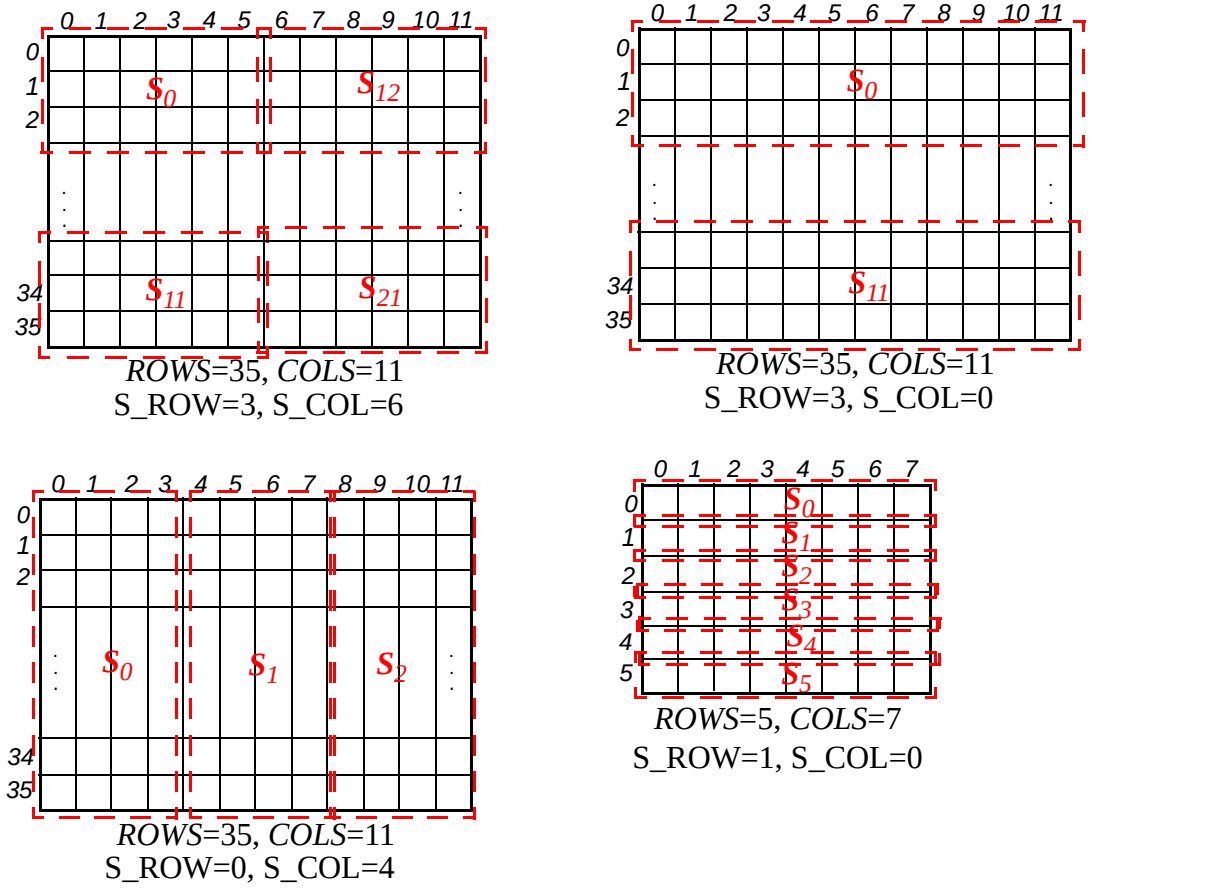


Figure 26-94. ***S*** Matrices Arrangement on the Sub-Carrier Grid

26.4.3.5.4.5.2 S Matrices Input Data Structure

For LNEQ mode, each ***S*** matrix is of *(Rx*+1) x (*Rx* +1) dimensions. There are two supported types of ***S*** matrices which are controlled by the *S_TYPE* parameter of the EQPE BD and affect the ***S*** matrix size:

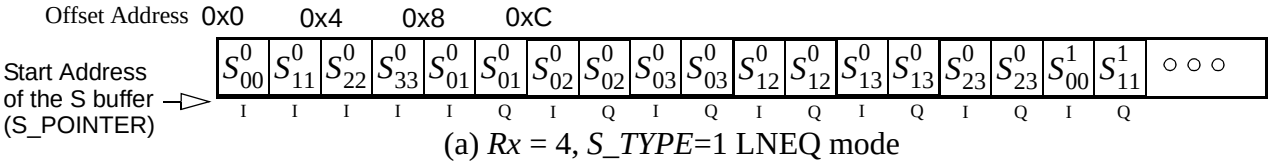
- *S_TYPE*=0: Real diagonal ***S*** matrix. In this case each S matrix is of size of (2 x *Rx)* Bytes

- *S_TYPE*=1: Hermitian ***S*** matrix. In this case each ***S*** matrix is of size of ((4 x *Rx* x *Rx)*/ 2*)* Bytes. The diagonal elements are real and written first followed by the upper-side off-diagonal elements which are complex and written row by row.

For MLEQ and TS mode the EQPE supports single scalar value (16I) per each ***S*** matrix (2 Bytes). For details see **Figure 26-95**.

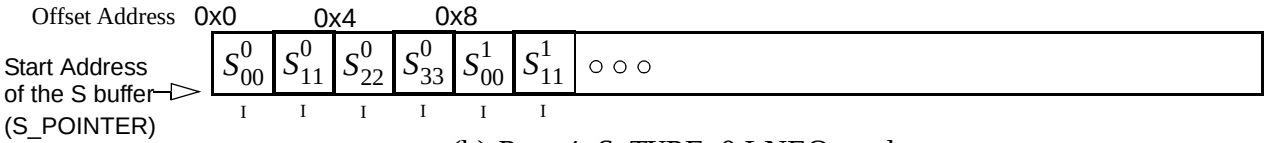
Figure 26-95 describes an example of the expected S matrix structure and the input vector structure for the different modes and types of S matrix:

$$S^0 = \begin{bmatrix} S_{00}^0 & S_{01}^0 & S_{02}^0 & S_{03}^0 \\ S_{01}^{0*} & S_{11}^0 & S_{12}^0 & S_{13}^0 \\ S_{02}^{0*} & S_{12}^{0*} & S_{22}^0 & S_{23}^0 \\ S_{03}^{0*} & S_{13}^{0*} & S_{23}^{0*} & S_{33}^0 \end{bmatrix} \quad \circ\circ\circ \quad S^N = \begin{bmatrix} S_{00}^0 & S_{01}^0 & S_{02}^0 & S_{03}^0 \\ S_{01}^{0*} & S_{11}^0 & S_{12}^0 & S_{13}^0 \\ S_{02}^{0*} & S_{12}^{0*} & S_{22}^0 & S_{23}^0 \\ S_{03}^{0*} & S_{13}^{0*} & S_{23}^{0*} & S_{33}^0 \end{bmatrix}$$



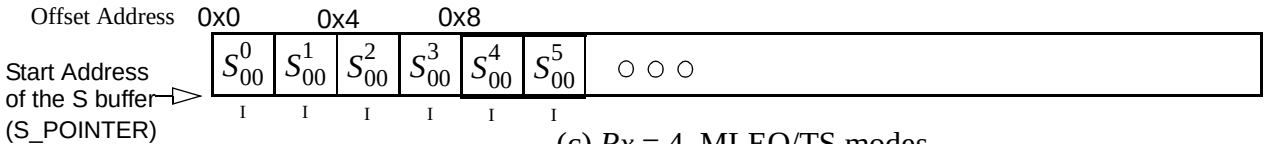
(a) *Rx* = 4, *S_TYPE*=1 LNEQ mode

$$S^0 = \begin{bmatrix} S_{00}^0 & 0 & 0 & 0 \\ 0 & S_{11}^0 & 0 & 0 \\ 0 & 0 & S_{22}^0 & 0 \\ 0 & 0 & 0 & S_{33}^0 \end{bmatrix} \quad \circ\circ\circ \quad S^N = \begin{bmatrix} S_{00}^0 & 0 & 0 & 0 \\ 0 & S_{11}^0 & 0 & 0 \\ 0 & 0 & S_{22}^0 & 0 \\ 0 & 0 & 0 & S_{33}^0 \end{bmatrix}$$



(b) *Rx* = 4, *S_TYPE*=0 LNEQ mode

$$S^0 = S_{00}^0 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \circ\circ\circ \quad S^N = S_{00}^N \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



(c) *Rx* = 4, MLEQ/TS modes

Figure 26-95. S matrices Data Structure

The number of ***S*** matrices in the job is calculated as follows and also should be configured in the *S_SIZE* field of the EQPE BD:

if (*S_ROWS*=0 & *S_COLS* =0)

S_SIZE = 1

if (*S_ROW* ≠ 0 & *S_COLS*=0)

$$S\_SIZE = \left\lceil \frac{ROWS + 1}{SROWS} \right\rceil$$

if (*S_ROW* = 0 & *S_COLS* ≠0)

$$S\_SIZE = \left\lceil \frac{COLS + 1}{SCOLS} \right\rceil$$

if (*S_ROW* ≠ 0 & *S_COLS* ≠0)

$$S\_SIZE = \left\lceil \frac{COLS + 1}{SCOLS} \right\rceil \times \left\lceil \frac{ROWS + 1}{SROWS} \right\rceil$$

The MAPLE-B2 expect to find the ***S*** matrices vector at the address pointed by the *S_POINTER* field of the EQPE BD.

26.4.3.5.4.5.3 S Matrices Granularity Limitation

The S buffer size is 16KByte for LNEQ mode and 8KBytes for MLEQ and TS modes. It is prohibited to give a command that exceeds the S buffer size that is, the job includes more S matrices than the S buffer can contain. Calculating the S buffer size is done as follows:

if (*ALG* = 0 & *S_TYPE* = 0)

S_SIZE x (*Rx* + 1) x 2B ≤ 16KB

if (*ALG* = 0 & *S_TYPE* = 1)

S_SIZE x $(Rx + 1)^2$ x 2B ≤ 16KB

if (*ALG* = 1 or *ALG* = 2)

S_SIZE x 2B ≤ 16KB

For example, for an EQPE job in LNEQ mode of 1200x12 sub-carriers (*ROWS* = 1199, *COLS*=11) with 4 receive antennas (*Rx* = 3) and *S_TYPE*=1:

- if *S_COL*=0 then *S_ROW*>=3 are the legal values.
- if *S_COL*=6 then *S_ROW*>=5 are the legal values.

26.4.3.5.4.5.4 S Scale Samples Input Data Structure

The EQPE supports two types of scaling methods for the ***S*** matrix samples:

- Single shared scale value for all input matrices.
- Different scale value for each input S matrix.

Controlling which ***S*** scale method is used is done using the *S_SCL_TYPE* parameter of the EQPE BD. The size of each scaling sample is one byte and the size of the scale vector is either one byte

if single shared scale value is used (*S_SCL_TYPE*=0), or *S_SIZE* Bytes (see **Section 26.4.3.5.4.5.2**) if different scale value for each S matrix is used (*S_SCL_TYPE*=1).

The MAPLE-B2 expects to find the ***S*** scale samples vector at the following address as calculated by the following EQPE BD parameters: Address = *SCL_PTR_BA* + (16 * *S_SCL_OFFSET*).

Note: If *S_SCL_TYPE*=1, the total amount of ***S*** matrices is limited to 1600 for the job.

26.4.3.5.4.6 C Samples

The C vector contains the samples of $\boldsymbol{C}=\boldsymbol{C}_x^{-1}$. It is assumed that ***C*** is a real, diagonal *Lx* x *Lx* matrix. Each element in the ***C*** matrix is represented by 16-bit fractional (real-part only). The C samples are represented by a 16-bit mantissa in 1q15 (*V*). All ***C*** samples share a common scale factor written to the *C_SCL* parameter of the EQPE BD such that the actual value of each of the C samples is: $V \times 2^{C\_SCL}$.

The C vector is used only for LNEQ mode and only if the *C_EN* parameter of the EQPE BD is set. If *C_EN*=0 then the C vector is not used and the EQPE uses ***C* = *Identity* *Matrix*** in the calculation.

26.4.3.5.4.6.1 C Matrices Arrangement on the Sub-Carrier Grid

It is assumed that the ***C*** matrix is identical for all the sub-carriers in the same column in the grid. Therefore only one ***C*** matrix is inserted for each column. **Figure 26-96** shows an example of ***C*** matrices arrangement on the sub-carrier grid.

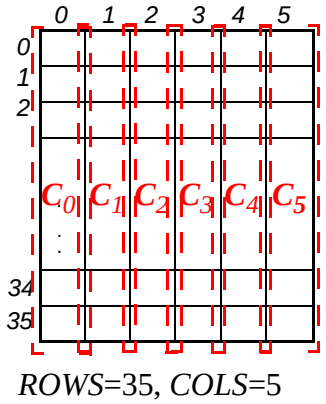


Figure 26-96. C Matrices Arrangement on Sub-Carrier Grid

26.4.3.5.4.6.2 C Samples Input Data Structure

Table 26-76 through **Table 26-79** describe the ***C*** samples vector input data structure. Assuming $C^x_{i,i}$ is the C matrix element where:

- *x* is the column index on the sub-carrier grid (*x* = 0...*COLS*)
- *i* is the column and row index of the element in the C matrix (*i* = 0...*Lx*)

Table 26-76. *C* Matrix Input Vector *When Lx* = 3 (*C* Matrix Is of Size 4x4)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	C^0_{00}		C^0_{11}		C^0_{22}		C^0_{33}		C^1_{00}		C^1_{11}		C^1_{22}		C^1_{33}	
...	...		...		...		...		...		...		...		...	
$\#\lceil (COLS+1)/2 \rceil$ [1]	...		...		...		...		C^{COLS}_{00}		C^{COLS}_{11}		C^{COLS}_{22}		C^{COLS}_{33}	

1.The location of the last ***C*** matrix can be located on the left 8 bytes of the memory line (if number of columns is odd).

Table 26-77. *C* Matrix Input Vector *When Lx* = 2 (*C* Matrix Is of Size 3x3)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	C^0_{00}		C^0_{11}		C^0_{22}		*don't care*		C^1_{00}		C^1_{11}		C^1_{22}		*don't care*	
...	...		...		...		...		...		...		...		...	
$\#\lceil (COLS+1)/2 \rceil$ [1]	...		...		...		...		C^{COLS}_{00}		C^{COLS}_{11}		C^{COLS}_{22}		*don't care*	

1.The location of the last ***C*** matrix can be located on the left 8 bytes of the memory line (if number of columns is odd).

Table 26-78. *C* Matrix Input Vector *When Lx* = 1 (*C* Matrix Is of Size 2x2)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	C^0_{00}		C^0_{11}		C^1_{00}		C^1_{11}		C^2_{00}		C^2_{11}		C^3_{00}		C^3_{11}	
...	...		...		...		...		...		...		...		...	
$\#\lceil (COLS+1)/4 \rceil$ [1]	...		...		...		...		...		...		C^{COLS}_{00}		C^{COLS}_{11}	

1.The location of the last ***C*** matrix depends on the value of *COLS*.

Table 26-79. *C* Matrix Input Vector *When Lx* = 0 (*C* Matrix Is of Size 1x1)

Memory line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#1	C^0_{00}		C^1_{00}		C^2_{00}		C^3_{00}		...		...		...		...	

The MAPLE_B2 expect to find the ***C*** samples vector at the address pointed by the *C_POINTER* field of the EQPE BD.

All the ***C*** samples in the job share one common scale factor which should be written to *C_SCL* field of the EQPE BD hence no C scale vector is required. The *C_SCL* field is ignored if *C_EN* of the EQPE BD is reset (*C_EN* = 0).

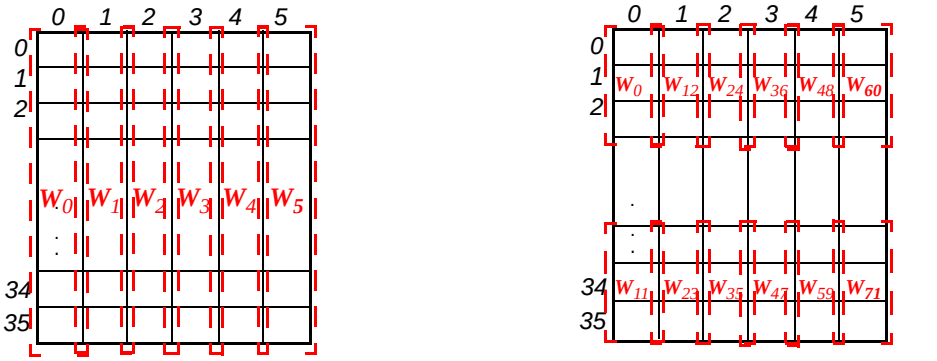
26.4.3.5.4.7 W Samples

The ***W*** samples are the interpolation weights. Each ***W*** sample is represented by a real 16-bit fractional fixed point representation of 2q14. The W samples are used only for LNEQ and MLEQ internal interpolations modes (*INTRP*=0,1).

26.4.3.5.4.7.1 W Samples Arrangement on the Sub-Carrier Grid

The parameter that controls the arrangement of the ***W*** samples on the sub-carrier grid is the *W_ROW* parameter of the EQPE BD. There is a distinct weight for every layer of every column for every reference symbol. The *W_ROW* parameter controls the number of rows that share the

same ***W*** samples. If *W_ROW*=0 then all the rows in the job share the same ***W*** samples. The following figure describe an example of the *W_ROW* usage:



(a) ROWS=35, *W_ROW*=0 *(b) ROWS*=35, *W_ROW*=3

Figure 26-97. W Matrices Arrangement on Sub-Carrier Grid

Note: if (*W_ROW* !=0) then the value of (*ROWS* + 1) must be a integer multiplication of *W_ROW*.

26.4.3.5.4.7.2 W Samples Input Data Structure

The ***W*** samples are of size of 2 bytes each. The total number of input ***W*** samples is:

if (*W_ROW* = 0)

II2: $2 \times 1 \times (Lx + 1) \times (COLS + 1) \times 2$ Bytes.

II4: $4 \times 1 \times (Lx + 1) \times (COLS + 1) \times 2$ Bytes.

else

II2: $2 \times \left\lceil \frac{ROWS + 1}{WROWS} \right\rceil \times (Lx + 1) \times (COLS + 1) \times 2$ Bytes

II4: $4 \times \left\lceil \frac{ROWS + 1}{WROWS} \right\rceil \times (Lx + 1) \times (COLS + 1) \times 2$ Bytes

The calculations above represent:

[references] x [weight per column] x [layers] x [columns] x [bytes per weight]

The total size of the ***W*** samples vector as calculated above is internally calculated by the MAPLE-B2 using the *W_SIZE* parameter of the EQPE BD which should be set as follows:

if ($W\_ROW = 0$)

$W\_SIZE = 1$

else

$W\_SIZE = \left\lceil \frac{ROWS + 1}{WROWS} \right\rceil$

The order to the ***W*** samples in the input vector is in correlation to the EQPE processing order of the sub-carriers as described in **Section 26.4.3.5.3**, *EQPE Sub-Carrier Grid*, that is, column by column.

If $W\_ROW = 0$, that is, all sub-carriers in each column share the same interpolation weight (see **Figure 26-97** (a)) then the expected order of the ***W*** samples in the input W vector would be:

For II2:

$W^0{}_0 0, W^0{}_0 1, W^0{}_1 0, W^0{}_1 1, \ldots, W^0{}_{Lx} 0, W^0{}_{Lx} 1, \backslash$
$W^1{}_0 0, W^1{}_0 1, W^1{}_1 0, W^1{}_1 1, \ldots, W^1{}_{Lx} 0, W^1{}_{Lx} 1, \backslash$
...
$W^{COLS}{}_0 0, W^{COLS}{}_0 1, W^{COLS}{}_1 0, W^{COLS}{}_1 1, \ldots, W^{COLS}{}_{Lx} 0, W^{COLS}{}_{Lx} 1.$

For II4:

$W^0{}_0 0, W^0{}_0 1, W^0{}_0 2, W^0{}_0 3, W^0{}_1 0, W^0{}_1 1, W^0{}_1 2, W^0{}_1 3, \ldots, W^0{}_{Lx} 0, W^0{}_{Lx} 1, W^0{}_{Lx} 2, W^0{}_{Lx} 3, \backslash$
$W^1{}_0 0, W^1{}_0 1, W^1{}_0 2, W^1{}_0 3, W^1{}_1 0, W^1{}_1 1, W^1{}_1 2, W^1{}_1 3, \ldots, W^1{}_{Lx} 0, W^1{}_{Lx} 1, W^1{}_{Lx} 2, W^1{}_{Lx} 3, \backslash$
...
$W^{COLS}{}_0 0, W^{COLS}{}_0 1, W^{COLS}{}_0 2, W^{COLS}{}_0 3, W^{COLS}{}_1 0, W^{COLS}{}_1 1, W^{COLS}{}_1 2, W^{COLS}{}_1 3, \ldots, W^{COLS}{}_{Lx} 0, \backslash$
$W^{COLS}{}_{Lx} 1, W^{COLS}{}_{Lx} 2, W^{COLS}{}_{lx} 3.$

Where:

— $W^x{}_y Z$ - *W* weight sample of column *x* ($x = 0 \ldots COLS$), with layer index *y* ($y = 0 \ldots Lx$). *Z* indicates it's the weight for reference input number *Z* (II2: $Z = 0,1$. II4: $Z = 0 \ldots 3$).

If $W\_ROW \neq 0$, that is, every number of rows equal to *W_ROW* the weights in the column change, the expected order of the ***W*** samples vector is as described in **Table 26-80**.

Table 26-80. *W* Samples Order in the *W* Vectors If $W\_ROW \neq 0$.

Interpolation Type	Index priority	W elements order
II2	for (x = 0, x <= *COLS*, x++) --for (i = 0, j <= J (= $\lceil (ROWS)/(WROWS) \rceil$), j++) ----for (y = 0, y <= *Lx*, y++) ------for (Z = 0, x <= *1*, x++) --------place $W^x{}_y Z$	$W^0{}_0 0, W^0{}_0 1, W^0{}_1 0, W^0{}_1 1, \ldots, W^0{}_{Lx} 0, W^0{}_{Lx} 1, W^0{}_1 0, \ldots,$ $W^0{}_{Lx} 0, W^0{}_{Lx} 1, \ldots, W^0{}_J 0, W^0{}_J 1, \ldots, W^0{}_{Lx} 1, W^1{}_0 0, \ldots,$ $W^1{}_{Lx} 1, \ldots, W^{COLS}{}_0 0, \ldots, W^{COLS}{}_{Lx} 1.$
II4	for (x = 0, x <= *COLS*, x++) --for (i = 0, j <= $\lceil (ROWS)/(WROWS) \rceil$, j++) ----for (y = 0, y <= *Lx*, y++) ------for (Z = 0, x <= *3*, x++) --------place $W^x{}_y Z$	$W^0{}_0 0, W^0{}_0 1, W^0{}_0 2, W^0{}_0 3, W^0{}_1 0, W^0{}_1 1, \ldots, W^0{}_{Lx} 0, W^0{}_{Lx} 1,$ $W^0{}_{Lx} 2, W^0{}_{Lx} 3, W^0{}_1 0, \ldots, W^0{}_{Lx} 0, W^0{}_{Lx} 1, W^0{}_{Lx} 2, W^0{}_{Lx} 3, \ldots,$ $W^0{}_J 0, W^0{}_J 1, W^0{}_J 2, W^0{}_J 3, \ldots, W^0{}_{Lx} 3, W^1{}_0 0, \ldots, W^1{}_{Lx} 3, \ldots,$ $W^{COLS}{}_0 0, \ldots, W^{COLS}{}_{Lx} 3.$

where for $W^x{}_y Z$:

- x is the sub-carrier grid column index (x = 0...*COLS*).
- i is the number of weight changes in each column (i = 0...$\lceil (ROWS)/(WROWS) \rceil$)
- y is the number of layers (y = 0...Lx)
- Z is the number of CE reference symbols (II2: Z = 0,1. II4: Z = 0...3)

The MAPLE-B2 expect to find the ***W*** samples vector at the address pointed by the *W_POINTER* field of the EQPE BD.

26.4.3.5.4.8 F Samples

The ***F*** samples are the feedback samples which are used in LNEQ mode for cancellation algorithms. The ***F*** samples are used only when cancellation is enabled, for details see **Section 26.4.3.5.6.3.1**, *Layer Cancellation*. Each feedback sample is represented by a 32-bit (16I,16Q) fractional mantissa in 1q15 (*V*). All ***F*** samples of the same column share the same scaling factor (F_SCL_{col}) such that the actual value of the sample is in: V x 2^{F_SCLcol}. The EQPE supports single layer cancellation only, therefore, the feedback samples of only one layer are expected as input to EQPE job.

26.4.3.5.4.8.1 F Samples Input Data Structure

The size of the F vector, which includes all ***F*** samples of a given layer to be cancelled, should include *(ROWS+1)* x *(COLS+1)* samples (each of size of 4 bytes). The order of the ***F*** samples input vector is column by column, that is, if F_x^y is a ***F*** sample of row index x (x = 0...*ROWS*) and of column index y (y = 0...*COLS*), then the expected order of the input vector is:

$$F^0_0, F^0_1, F^0_2,..., F^0_{ROWS}, F^1_0, F^1_1, F^1_2,..., F^1_{ROWS},..., F^{COLS}_0, F^{COLS}_1, F^{COLS}_2,..., F^{COLS}_{ROWS}.$$

The following figure illustrates such example:

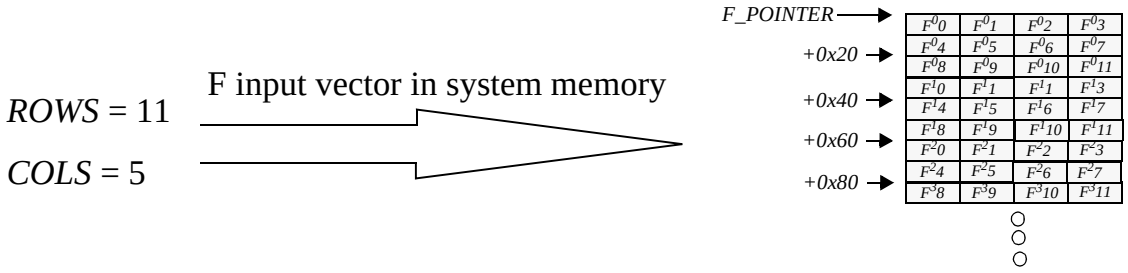


Figure 26-98. F Input Vector Example

The MAPLE-B2 expects to find the F input vector at the address pointed by the *F_POINTER* parameter of the EQPE BD.

26.4.3.5.4.8.2 F Scale Input Data Structure

The F scale input vector should include a single scale value, each of one byte, for each column as defined in the *COLS* field of the EQPE BD. The expected order of the scale values is:

$f_0, f_1, f_2, \ldots, f_{COLS}$.
where f_x is the scale value of column x (x = 0...*COLS*).

The MAPLE-B2 expects to find the ***F*** scale samples vector at the following address as calculated by the following EQPE BD parameters: Address = *SCL_PTR_BA* + (16 * *F_SCL_OFFSET*).

26.4.3.5.4.9 EQPE Input Data Structure - Summary

The following table summarize the EQPE input data vectors for each of its modes. The table summarize the sizes of each required vector in each mode.

Table 26-81. EQPE Input vectors sizes (in bytes)

Vector Type	LNEQ mode	MLEQ mode	TS mode	MIV mode
Y	(*ROWS*+1)*(*COLS*+1)*(*Rx*+1)*4	(*ROWS*+1)*(*COLS*+1)*(*Rx*+1)*4	(*ROWS*+1)*(*COLS*+1)*(*Lx*+1)*4	NA
Y Scale	(*COLS*+1)*(*RX*+1)	(*COLS*+1)*(*Rx*+1)	(*COLS*+1)*(*Rx*+1)	NA
H/R/M	If *INTRP*=0 2*(*ROWS*+1)*(*Rx*+1)*(*Lx*+1)*4 If *INTRP*=1 4*(*ROWS*+1)*(Rx+1)*(*Lx*+1)*4 If *INTRP*=2 (*COLS*+1)*(*ROWS*+1)*(*Rx*+1)*(*Lx*+1)*4 If *INTRP*=3 (*ROWS*+1)*(*Rx*+1)*(*Lx*+1)*4		(*COLS*+1)*(*ROWS*+1)*(*Lx*+1)*(*Lx*+1)*2	if *FH*=0 (*ROWS*+1)*(*Lx*+1)*(*Lx*+1)*2 if *FH*=1 (*ROWS*+1)*(*Lx*+1)*(*Lx*+1)*4
H/R/M Scale	If *INTRP*=0 2*(*Rx*+1)*(*Lx*+1) If *INTRP*=1 4*(*Rx*+1)*(*Lx*+1) If *INTRP*=2 (*COLS*+1)*(*Rx*+1)*(*Lx*+1) If *INTRP*=3 (*Rx*+1)*(*Lx*+1)		if *RM_SCL_TYPE*=0 1 Byte if *RM_SCL_TYPE*=1 (*COLS*+1)*(*ROWS*+1)	if *RM_SCL_TYPE*=0 1 Byte if *RM_SCL_TYPE*=1 (*ROWS*+1)
S	If *S_TYPE*=0 Roundup((*ROWS*+1)/*S_ROW*)* Roundup((*COLS*+1)/*S_COL*)*(*Lx*+1)*2 if *S_TYPE*=1 Roundup((*ROWS*+1)/*S_ROW*)* Roundup((*COLS*+1)/*S_COL*)*(*Lx*+1)*(*Lx*+1)*2	Roundup((*ROWS*+1)/*S_ROW*)* Roundup((*COLS*+1)/*S_COL*)*2		NA

Table 26-81. EQPE Input vectors sizes (in bytes)

Vector Type	LNEQ mode	MLEQ mode	TS mode	MIV mode

S Scale	if *S_SCL_TYPE*=0:
	1 Byte
	if *S_SCL_TYPE*=1:
	Roundup((*ROWS*+1)/*S_ROW*)*Roundup((*COLS*+1)/*S_COL*)
				NA

W	If *INTRP*=0 2*Roundup((*ROWS*+1)/*W_ROW*)*(*COLS*+1)*(*Lx*+1)*2 If *INTRP*=1 4*Roundup((*ROWS*+1)/*W_ROW*)*(*COLS*+1)*(*Lx*+1)*2		NA	NA
C	if *C_EN*=1: (*COLS*+1)*(*Lx*+1)*2	NA	NA	NA
C Scale	if *C_EN*=1 1 Byte	NA	NA	NA
F	if *F_EN*=1: (*COLS*+1)*(*ROWS*+1)*4	NA	NA	NA
F Scale	if *F_EN*=1 1 Byte	NA	NA	NA

26.4.3.5.5 EQPE Processing

The following sections describe the different aspects of the EQPE processing.

26.4.3.5.5.1 EQPE Algorithm Description

The EQPE supports the following operations:

- Linear Equalization (MMSE or Zero Forcing)
- ML Equalization and Detection (using QRD-M algorithm)
- Tree-Search
- General purpose Matrix Inversion

26.4.3.5.5.2 Linear Equalization

Linear Equalization is the implementation of all the equalization schemes that include Zero-Forcing or MMSE with or without cancellation (SIC schemes).

The model assumptions are:

$$\boldsymbol{y} = \boldsymbol{Hx} + \boldsymbol{n}$$

where:

$\boldsymbol{y}$ is a N_Rx1 vector of the received samples from each *Rx* antenna.

$\boldsymbol{x}$ is a N_Tx1 the transmitted signal.

$\boldsymbol{H}$ is the N_RxN_T CE (Channel Estimation) matrix.

$$\boldsymbol{x} \sim CN\{\underline{\mathbf{0}}, \boldsymbol{C}_x\}\ , \quad \boldsymbol{n} \sim CN\{\underline{\mathbf{0}}, \boldsymbol{C}_n\}$$

where:

$\boldsymbol{C}_x$ is the N_TxN_T Covariance matrix of the transmitted signal.

$\boldsymbol{C}_n$ is the N_RxN_R Covariance matrix of the noise.

CN is complex Normal distribution.

The MMSE Estimation is:

$$\hat{\boldsymbol{x}} = (\boldsymbol{H}^H \boldsymbol{C}_n^{-1} \boldsymbol{H} + \boldsymbol{C}_x^{-1})^{-1} \boldsymbol{H}^H \boldsymbol{C}_n^{-1} \boldsymbol{y}$$

The notation of this document is that: $\boldsymbol{C} = \boldsymbol{C}_x^{-1}$, $\boldsymbol{S} = \boldsymbol{C}_n^{-1}$

In this notation the implemented output is:

$$\hat{\boldsymbol{x}} = (\boldsymbol{H}^H \boldsymbol{S} \boldsymbol{H} + \boldsymbol{C})^{-1} \boldsymbol{H}^H \boldsymbol{S} \tilde{\boldsymbol{y}} \quad , \qquad \textbf{Equation 1}$$

$\boldsymbol{H}$,$\boldsymbol{S}$,$\boldsymbol{C}$ and $\boldsymbol{y}$ are the inputs to the EQPE (see **Section 26.4.3.5.4**).

If cancellation is configured: $\tilde{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{h}_i f_i$, otherwise $\tilde{\boldsymbol{y}} = \boldsymbol{y}$.

where:

$\boldsymbol{h}_i$ is the i[th] column of $\boldsymbol{H}$.

f_i is the estimation of layer i calculated from previous iterations (feedback layer).

Note: The EQPE supports only one internal layer cancellation. If more than one layer is required to be cancelled, then the host should perform the cancellation externally and give the EQPE the new $\boldsymbol{y}$ samples ($\boldsymbol{y}$ after cancellation) as the y samples.

Equation 1 is calculated in 3 main steps:

1. Matrix Multiplication:

$$\boldsymbol{D} = (\boldsymbol{H}^H \boldsymbol{S} \boldsymbol{H} + \boldsymbol{C})$$

$$\boldsymbol{z} = \boldsymbol{H}^H \boldsymbol{S} \tilde{\boldsymbol{y}}$$

2. QR Decomposition (using Given's-Rotation):

$$[\mathbf{Q},\mathbf{R}] = \mathrm{qr}(\mathbf{D})$$
$$\boldsymbol{q} = \boldsymbol{Q}^H \boldsymbol{z}$$

3. Back-Substitution:

solve $\boldsymbol{Rx}=\boldsymbol{q}$

Zero-Forcing algorithm uses the $\boldsymbol{H}$, $\boldsymbol{y}$, $\boldsymbol{C}=\boldsymbol{0}$, $\boldsymbol{S}=\boldsymbol{I}$ input such that

$$\hat{\boldsymbol{x}}_{ZF} = (\boldsymbol{H}^H\boldsymbol{H})^{-1}\boldsymbol{H}^H\tilde{\boldsymbol{y}} \qquad \textbf{Equation 2}$$

26.4.3.5.5.3 QRD-M Equalization (MLEQ)

QRD-M is a way to implement the Maximum Likelihood estimator or to approximate it.

For the model:

$$\boldsymbol{y} = \boldsymbol{Hx} + \boldsymbol{n}$$

$$\boldsymbol{n} \sim CN\{\boldsymbol{0}, \sigma^2\boldsymbol{I}\},\ \boldsymbol{x} \in QAM$$

The *ML* solution:

$$LLR(b) = \frac{1}{\sigma^2}[-\min_{\boldsymbol{x}:\ b=1}\{d(\boldsymbol{x})\} + \min_{\boldsymbol{x}:\ b=0}\{d(\boldsymbol{x})\}],\ d(\boldsymbol{x}) = \|\boldsymbol{y}-\boldsymbol{Hx}\|^2 \qquad \textbf{Equation 3}$$

where

$\boldsymbol{x}$ - the transmitted symbol vector $(N_T x1)$

$d(\boldsymbol{x})$ - the squared Euclidian distances between the received signal and constellation point $\boldsymbol{x}$.

$\boldsymbol{H}$ - the channel matrix $(N_R x N_T)$

$\boldsymbol{y}$ - the received signal vector $(N_R x1)$

σ^2 - the noise variance

The notation of this document is $\boldsymbol{S} = \boldsymbol{C}_n^{-1}$ and in MLEQ mode it is always a scaled identity matrix for example, $\boldsymbol{S} = \sigma^{-2}\boldsymbol{I}$

This solution has been shown to be optimal (without exploiting channel coding).

QRD-M:

First stage is to perform a QR Decomposition on the CE matrix $\boldsymbol{H}$, i.e $\boldsymbol{H}=\boldsymbol{QR}$.

The QRD has 2 properties:

- $\boldsymbol{Q}$ is a unitary matrix that is, $\boldsymbol{Q}^H\boldsymbol{Q}=\boldsymbol{I}$
- $\boldsymbol{R}$ is an upper-right triangular matrix.

This allows rewriting of the *ML* function:

$$d(\boldsymbol{x}) = \|\boldsymbol{y}-\boldsymbol{Hs}\|^2 = \|\boldsymbol{y}-\boldsymbol{QRx}\|^2 = \left\|Q^H\boldsymbol{y}-\boldsymbol{Rx}\right\|^2 = \|\tilde{\boldsymbol{y}}-\boldsymbol{Rx}\|^2$$

and the *ML* solution remains unchanged:

$$LLR(b) = \boldsymbol{S}[-\min_{\boldsymbol{x}:\,b=1}\{d(\boldsymbol{x})\} + \min_{\boldsymbol{x}:\,b=0}\{d(\boldsymbol{x})\}],\ d(\boldsymbol{x}) = \left\|\tilde{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{x}\right\|^2 \qquad \textbf{Equation 4}$$

The distances $d(\boldsymbol{x})$ in **Equation 4** can now be separated to N_T components, and the problem of finding the minimal d's can be illustrates as a tree-search.

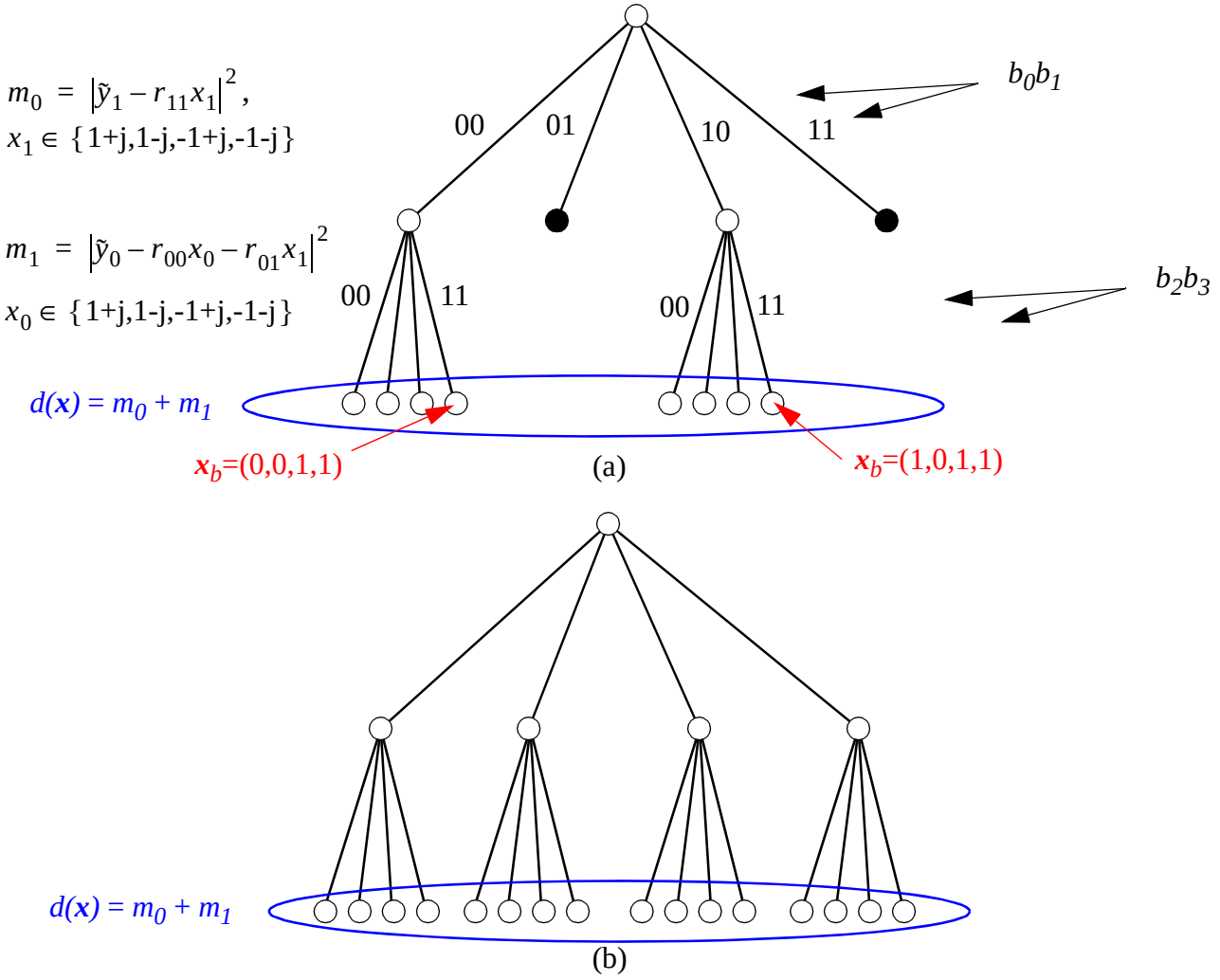


Figure 26-99. QRD-M Tree Search Example. QPSK, 2 Layers (a) M=2 (b) M=M_{max}=4

The *ML* solution can be obtained by calculating all the nodes of the tree. Each "leaf" of the tree (the nodes at the bottom level) represent the d's which are the sum of the first and second metrics. After calculating all of the *d*'s, the LLRs of the transmitted bits can be calculated according to **Equation 4**.

The *QRD-M* is a way to further reduce the complexity at the expense of the full performance, that is, it achieves sub-optimal performance with less operations. This is done by discarding parts of the tree that are less likely to be used by the LLR calculation **Equation 4**.

This is done by choosing the parameter $1<=M<=M_{max}$. At the first level of tree, all the nodes are calculated, and the *M* best nodes (the nodes with the minimal accumulated metric) are selected as survivors. At the next level, only the nodes coming from the surviving nodes are calculated. Again the M-best nodes are selected, and so on until the final level of the tree where the leaves represent the d's and the LLRs are calculated. At the final level of the tree, only (*Card* x *M*) *d*'s are available instead of $Card^{Nt}$. Because of this, we get an approximation of the LLRs and sub-optimal performance. (*Card* is the cardinality of the symbol constellation, for example, $Card_{QPSK}$=4,$Card_{16QAM}$=16...)

A greater problem might occur when one of the minimization problems solved in **Equation 4** to calculate the LLRs encounter an empty group (no *d*'s of that group survived), the method to deal with this problem are described in **Section 26.4.3.5.5.4**, *LLR Calculations*.

M provides a convenient trade-off control between performance and complexity. By choosing the maximal $M_{max}=Card^{(Nt-1)}$, we can obtain a full tree-search and the optimal performance. By reducing *M* we reduce complexity and narrow the tree-search.

26.4.3.5.5.4 LLR Calculations

The LLR of each bit is calculated according to **Equation 4**. When one of the minimization groups is empty, the EQPE takes maximal absolute value LLR from all the bits in the sub-carrier and assigns this value to the LLR with the proper sign.

26.4.3.5.5.5 Tree Search

The EQPE performs tree search only. The QR Decomposition is performed externally on the CE matrix ("H") and the EQPE is given R and Q^Hy (*Lx* x 1 vector) and performs tree search to produce LLRs of the transmitted bits as described in **Section 26.4.3.5.5.3**, *QRD-M Equalization (MLEQ)*.

26.4.3.5.5.6 Matrix Inversion

The EQPE performs matrix inversion for square matrices of size *Lx* x *Lx* using QR Decomposition:

$$[Q,R] = qr(M) \qquad \text{Equation 5}$$

The inversion is completed with Back-Substitution:

$$Rx_i = q_i \text{ , } i=1,...,Lx \qquad \text{Equation 6}$$

where:

$\boldsymbol{q}_i$ is the i^{th} column of $\boldsymbol{Q}^H$.

$\boldsymbol{x}_i$ is the i^{th} column of $\boldsymbol{M}^{-1}$ which are via Back-Substitution.

26.4.3.5.5.7 CE Matrix Interpolation

CE Matrix Interpolation is supported only for LNEQ and MLEQ modes (*ALG*=0,1). Interpolation is performed by using interpolation weights - "*W* samples" (see **Section 26.4.3.5.4.7**, *W Samples*) and the CE matrices samples - "*H* samples" (see **Section 26.4.3.5.4.2**, *H Samples*).

The EQPE supports 4 interpolation schemes:

- *Internal Interpolation 2* (II2)*:* (*INTRP*=0)
  In this mode the EQPE performs the interpolation internally using 2 references (A and B) for each sub-carrier. Only the reference H matrices are inserted and the EQPE performs weighted interpolation using them.
  Each row *k* has 2 CE reference matrices $\boldsymbol{H}^{A}_{k}$ and $\boldsymbol{H}^{B}_{k}$. By configuring $w^{A}_{i,j}$ and $w^{B}_{i,j}$ (*i*=0,1,...,*COLS* and *j*=0,...,*Lx*), Each column and layer can have a different weight for each reference matrix.
  The implemented interpolation for the sub-carrier in the k^{th} row in column *i* is (2 layers example):

$$H_{k,i} = [w^{A}_{i,0} \times h^{A}_{k,0} + w^{B}_{i,0} \times h^{B}_{k,0} \; ; \; w^{A}_{i,1} \times h^{A}_{k,1} + w^{B}_{i,1} \times h^{B}_{k,1}] \qquad \textbf{Equation 7}$$

  $\boldsymbol{h}^{A}_{k,0}$ is column 0 of reference matrix A of row k.

- *Internal Interpolation 4* (II4)[1]*:* (*INTRP*=1)
  In this mode the EQPE performs the interpolation internally using 4 reference matrices (A,B,C and D) for each sub-carrier. Only the reference H matrices are inserted and EQPE performs weight interpolation using them.
  Each row *k* has 4 CE reference matrices $\boldsymbol{H}^{A}_{k}$,$\boldsymbol{H}^{B}_{k}$,$\boldsymbol{H}^{C}_{k}$ and $\boldsymbol{H}^{D}_{k}$, and each Resource Element (RE) in the row can have a different weight for each reference vector or column in the reference matrix.
  The implemented interpolation for the sub-carrier in the k^{th} row in column *i* is:

$$h_{k,i} = [w^{A}_{i,0} \times h^{A}_{k} + w^{B}_{i,0} \times h^{B}_{k} + w^{C}_{i,0} \times h^{C}_{k} + w^{D}_{i,0} \times h^{D}_{k}] \qquad \textbf{Equation 8}$$

  $\boldsymbol{h}^{A}_{k}$ is reference vector A of row k.

- *External Interpolation:* (*INTRP*=2)
  In this mode, CE interpolation is performed by the host and the EQPE is given an explicit *H* matrix for each RE.
- *No Interpolation:* (*INTRP*=3)
  In this mode, no interpolation is performed and the EQPE is given one *H* matrix for each row. This matrix is used for each RE in the row.

1. This mode is not supported for 8x4, 8x3 jobs (*Rx*=7 and *Lx*=3,2).

Figure 26-100 shows an example of 3G-LTE and WiMAX allocations examples, and the translations of these jobs to the sub-carrier grid in internal interpolation 2 mode.

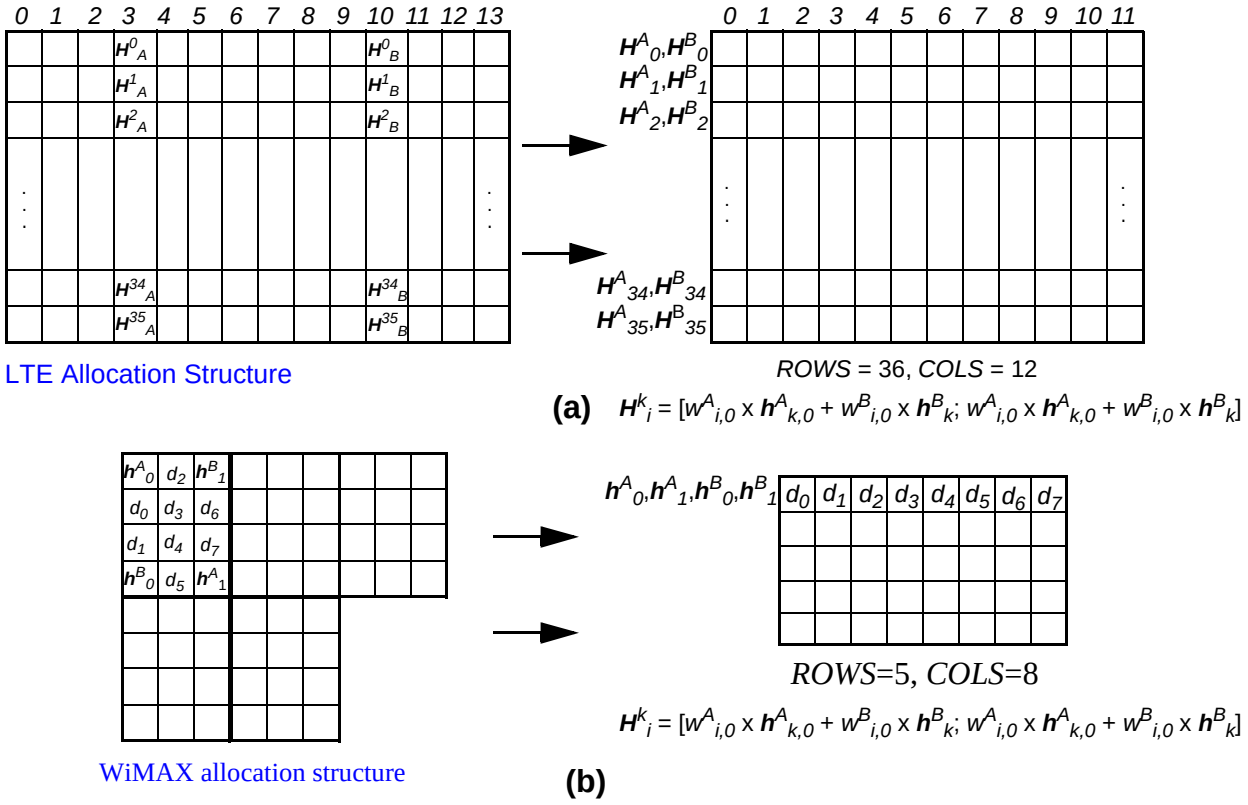


Figure 26-100. Internal Interpolation 2 (a) 3G-LTE Allocation (b) WiMAX Allocation

26.4.3.5.6 Scaling

The EQPE internal calculations are implemented with floating-point arithmetic. Each sample is represented by a mantissa of bits 20 (or more) and an 8-bit exponent factor.

26.4.3.5.6.1 Input Samples Scaling

The following subsections describe the scaling for various operating modes.

26.4.3.5.6.1.1 LNEQ, MLEQ. and TS Modes

Each of the input samples are represented in Block-Floating-Point representation (each data sample does not have a separate scaling factor rather there is one scaling factor shared for a group of samples depending on the input type). Each sample is composed by a mantissa (1q15 fraction) either real (16I) or complex (16I,16Q) and an 8-bit scale factor (*SCL*) such that the actual value of the sample is: $mantissa \times 2^{SCL}$.

26.4.3.5.6.1.2 MIV Mode

All input samples are in Block-Floating-Point representation. Each samples is composed by a mantissa (1q15 fraction) either real (16I) or complex (16I,16Q). There is one 8-bit scaling factor per matrix or one common scale factor for all matrices.

26.4.3.5.6.2 Output Samples Alignment (Scaling)

The EQPE supports several output alignment (scaling) configurations depending on its processing mode. The following sections describe the output alignment options for each mode.

26.4.3.5.6.2.1 Output Samples Alignment in LNEQ Mode

The output samples in LNEQ are initially (before alignment) in floating point representation with 32-bit (16I,16Q) mantissa and 8-bit exponent.

There are 3 supported output alignment modes in LNEQ which can be configured separately for each layer by configuring the *ALIGNx* field of the EQPE BD (x = 0...*Lx*):

- Max scale output alignment *(ALIGN<layer>*=0). In this mode each column and each layer is aligned separately to the maximal exponent in that column and layer. All the output samples for this column and layer shares the same exponent (the maximal) that is output by the MAPLE-B2 as described in **Section 26.4.3.5.8.1**, *EQPE Outputs for LNEQ*.
- User defined output alignment (*ALIGN<layer>*=1). In this mode, the host configures a scale value in the *ALIGN_VAL<layer>* field of the EQPE BD. The EQPE aligns all the output samples of the layer to that scale value. In case of overflow the EQPE saturates the sample, for example, assign the highest-positive/lowest-negative values according to sign of the sample.
- No Alignment (*ALIGN<layer>*=2). In this mode the EQPE does not align the output samples and they remain in floating point representation. The MAPLE-B2 outputs the scaling values for each symbol of each column of each layer as described in **Section 26.4.3.5.8.1**, *EQPE Outputs for LNEQ*.

Note: The User defined output alignment (*ALIGN<layer>*=1) may have a negative effect on the precision of the output samples due to discarding of LSB's (right shifts performed to reach *ALIGN_VAL<layer>*) and/or samples saturation (left shifts performed to reach *ALIGN_VAL<layer>*) that may occur due to the alignment.

26.4.3.5.6.2.2 Output Samples Alignment in MLEQ and TS Modes

The output samples (LLRs) in MLEQ and TS modes are initially (before alignment) in floating point representation 8-bit of 1q7 mantissa and 8 bit exponent for each LLR.

There are 2 supported alignment modes which can be configured separately for each layer by configuring *ALIGNx* field of the EQPE BD (x = 0...*Lx*):

- User defined output alignment (*ALIGN<layer>*=1). In this mode, the host configures a scale value in the *ALIGN_VAL<layer>* field of the EQPE BD. The EQPE aligns all the output samples of the layer to that scale value. In case of overflow the EQPE saturates the sample, for example, assign the highest-positive/lowest-negative values according to sign of the sample.
- No Alignment (*ALIGN<layer>*=2). In this mode the EQPE does not align the output samples and they remain in floating point representation. The MAPLE-B2 outputs the scaling values for each symbol of each column of each layer as described in **Section 26.4.3.5.8.2**, *EQPE Outputs for MLEQ and TS modes*.

26.4.3.5.6.2.3 Output Samples Alignment in MIV Mode

The output samples in MIV are initially (before alignment) in floating point representation with 32 bit (16I,16Q) of 1q15 mantissa and 8 bit exponent.

There are 4 supported alignment modes in MIV mode which can be configured in the *ALIGN0* field of the EQPE BD:

- Max scale output alignment (*ALIGN0*=0). In this mode each matrix is aligned separately to the maximal exponent in the matrix. All the output samples for the matrix shares the same exponent (the maximal). The maximal exponents (one for each matrix) is output by MAPLE-B2 as described in **Section 26.4.3.5.8.3**, *EQPE Outputs for MIV*.
- User Defined output alignment (*ALIGN0*=1). In this mode, the host configures a scale value in the *ALIGN_VAL0* field of the EQPE BD. The EQPE aligns all the output samples of all the matrices to that scale value, in case of overflow the EQPE saturates the sample, for example, assign the most highest positive/negative values according to sign of the sample.
- No Alignment (*ALIGN0*=2). In this mode the EQPE does not align the output samples and they remain in floating point representation. The MAPLE-B2 output the scale value of each element in each of the matrices as described in **Section 26.4.3.5.8.3**, *EQPE Outputs for MIV*.
- Global Max Alignment (*ALIGN0*=3). In this mode all output samples are aligned to an exponent value calculated by the first batch of EQPE MIV processing: due to limited EQPE output buffer size, the maximum alignment value of the first batch of processed matrices (which fills the output buffer), is used as an alignment value for the rest of the job processed matrices. Batch size varies according to matrix dimensions and type (full/hermitian). For example, for full 4×4 MIV job, batch size is 512 matrices. The single output scale value is output by MAPLE-B2 as described in **Section 26.4.3.5.8.3**, *EQPE Outputs for MIV*.

Note: The user-defined output alignment (*ALIGN0* = 1) may have a negative effect on precision of the output samples due to discarding of LSBs (right shifts performed to reach *ALIGN_VAL0*) and/or sample saturation (left shifts performed to reach *ALIGN_VAL0*) that may occur due to the alignment.

26.4.3.5.6.3 Layer Cancellation Rank Reduction and Layer Discarding

Layer Cancellation, Rank Reduction and Layer Discarding are optional features for LNEQ mode intended for Interference Cancellation (IC) schemes, where the equalization is performed in several iterations to improve the accuracy of the estimation. As from the second iteration, a-priory information is available in the form of improved estimations of some or all of the layers being estimated. These estimations is named feedback layers and is denoted as "f_i" for the i^{th} feedback layer.

26.4.3.5.6.3.1 Layer Cancellation

The EQPE supports layer cancellation by subtracting the feedback samples from the ***y*** samples. The EQPE supports single internal layer cancellation. If more than single layer cancellation is required, they may be performed by the host or one in the EQPE, and the rest by the host. For example, in 4x4 system where 3 cancellations are needed, the host may perform the first 2 cancellations, giving the EQPE a job with the updated ***y*** samples and layer cancellation of the third layer, alternatively the host may perform all 3 cancellations and give the EQPE a job with no internal cancellation.

The cancellation performed for each sub-carrier:

$$\tilde{\boldsymbol{y}}^k = \boldsymbol{y}^k - \boldsymbol{h}_i^k f_i^k \qquad \textbf{Equation 9}$$

where:

$\boldsymbol{h}^k_i$ is the i^{th} column of ***H*** in row k

f^k_i is row k of the estimation of layer i from a-priory knowledge.

This cancellation is performed before the equalization calculation.The cancellation performed is always a soft cancellation that is, the value subtracted from the ***y*** samples is always the "soft" values stored in the feedback buffer. Hard cancellation may be achieved by writing "hard" values to the feedback buffer.

Layer Cancellation is configured by setting *F_EN* bit and specifying the layer to be cancelled in *LTC* field of the EQPE BD. In addition to basic input samples, the feedback should also be written to the EQPE as described in **Section 26.4.3.5.4.8**, *F Samples*.

When performing cancellation (either internal or external) it is recommended to modify the MMSE calculation. This can be done in two ways:

1. Updating the signal covariance matrix to the new variance of the cancelled layers (it is assumed that the ***C*** matrices are calculated by the host). This is done by setting *C_EN* bit of the EQPE BD and preparing the ***C*** samples as described in **Section 26.4.3.5.4.6**, *C Samples*.
2. Rank reduction - see **Section 26.4.3.5.6.3.2**, *Rank Reduction*.

Note: There is no restriction on enabling both features in the same job, that is, the cancelled layer can also be reduced from the equalization calculations.

26.4.3.5.6.3.2 Rank Reduction

In some scenarios for example, Hard-IC or Soft-IC (when the feedback layers are estimated with high confidence), it is desirable to reduce the rank of the estimation problem (for example, reducing the equalization calculation from 4x2 to 4x1) after the cancellation by omitting one of the layers from the calculation. This is done by discarding the i^{th} column of ***H*** after the cancellation and reducing the problem to less layers equalization.

Rank reduction may be performed on all layers regardless of cancellation to support cases where the host performs the cancellation externally and requires rank reduction for those layers. The reduction is done on a column basis, so reduction for each column for each layer must be configured separately.

Enabling Rank Reduction is done by setting the *RRx*[15:0] bits of the EQPE BD (x = 0...3). Setting a bit in the *RRx* field reduces the matching column in the layer. For example, if *RR0*[3:0] bits are set then columns 0 to 3 of layer 0 are reduced from the calculations.

Note: Reducing all layers in a certain column is forbidden.

26.4.3.5.6.3.3 Layer Discarding

Layer Discarding is a feature intended for IC schemes that require the estimation of only some of the layers while still using them in the calculation of the other layers (in oppose to Rank Reduction). The MAPLE-B2 does not output these layers either for further iDFT processing (if the *POST_IDFT* bit of the EQPE BD is set) or to the system memory, thus saving redundant system data traffic or iDFT processing.

Layer Discarding is enabled by setting the *LD<layer>* bits of the EQPE BD. Discarding all layers in the job is forbidden.

26.4.3.5.7 CONVPE Support—FDU Processing

As part of CONVPE processing (see **Section 26.4.3.9**, *CONVPE Convolution and Correlation Processing Operation*), the EQPE includes a dedicated logic (FDU logic) which assists with vector multiplication processing. The FDU logic, which is not functional during EQPE processing (LNEQ, MLEQ, TS and MIV modes), is targeted for CONVPE processing assist only. Because both Equalization logic and FDU logic share the same EQPE internal resources (memories and logic), they cannot work in parallel, that is, the EQPE is either in FDU processing mode or Equalization processing mode. The FDU logic cannot be directly controlled by the host therefore has no programming model interface of its own. It can only be controlled by MAPLE-B2 firmware as part of the CONVPE processing. For functional details, see **Section 26.4.3.9.2.1**, *FDU Operation*.

26.4.3.5.8 EQPE Output Data Structure

The EQPE output data buffers varies with respect to the current processing mode (LNEQ, MLEQ, TS and MIV) and enabling/disabling additional features related to each processing mode. The following sections describe the output data buffers structures and type of the EQPE with respect to its processing modes.

26.4.3.5.8.1 EQPE Outputs for LNEQ

In LNEQ mode, the output samples are the estimation of the transmitted signals calculated according to **Equation 1** for each layer in the job. Each output sample is composed of a 32 bit mantissa (16I,16Q) in 1q15 representation (*V*) and an 8-bit signed exponent (*EXP*). The value of the sample is: $V \times 2^{EXP}$.

In the EQPE output buffer the mantissa is normalized, that is, for each sample at least one of the real or imaginary parts begins with 2’b01 (for positive value) or 2’b10 (for negative value). An exception is when both real and imaginary parts equal zero, in this case the exponent holds the most negative possible value 0x80 (-128). For LNEQ mode the EQPE supports 3 output alignment (scaling) types as described in **Section 26.4.3.5.6.2.1**, *Output Samples Alignment in LNEQ Mode*.

26.4.3.5.8.1.1 Data Samples (mantissa) Outputs for LNEQ

The valid output size in LNEQ mode depends upon the number of sub-carriers ((*ROWS+1)* x *(COLS + 1)*), the number of layers (*Lx*) and the cancellation options (layer discard *LDx* and rank reduction *RRx[y]*). In general, the output data structure size of each layer is (*ROWS*+1) x (*COLS*+1) x (4Byte mantissa), which is arranged column by column starting with column #0 to column #*COLS*. The output address pointer of each layer is pointed by the *OUT_POINTERx* field of the EQPE BD.

If the Layer Discard option (*LDx* field of the EQPE BD) is set for any of the layers, the MAPLE-B2 does not output any data related to that layer.

If the Rank Reduction option (*RRx[y]* field of the EQPE BD) is set for any of the columns in any of the layers, the MAPLE-B2 skip the reduced column outputs and continue outputting the following non reduced column while skipping the location of the reduced column in the system memory.

The following figure describe the MAPLE-B2 output mantissa structure for the different configurations:

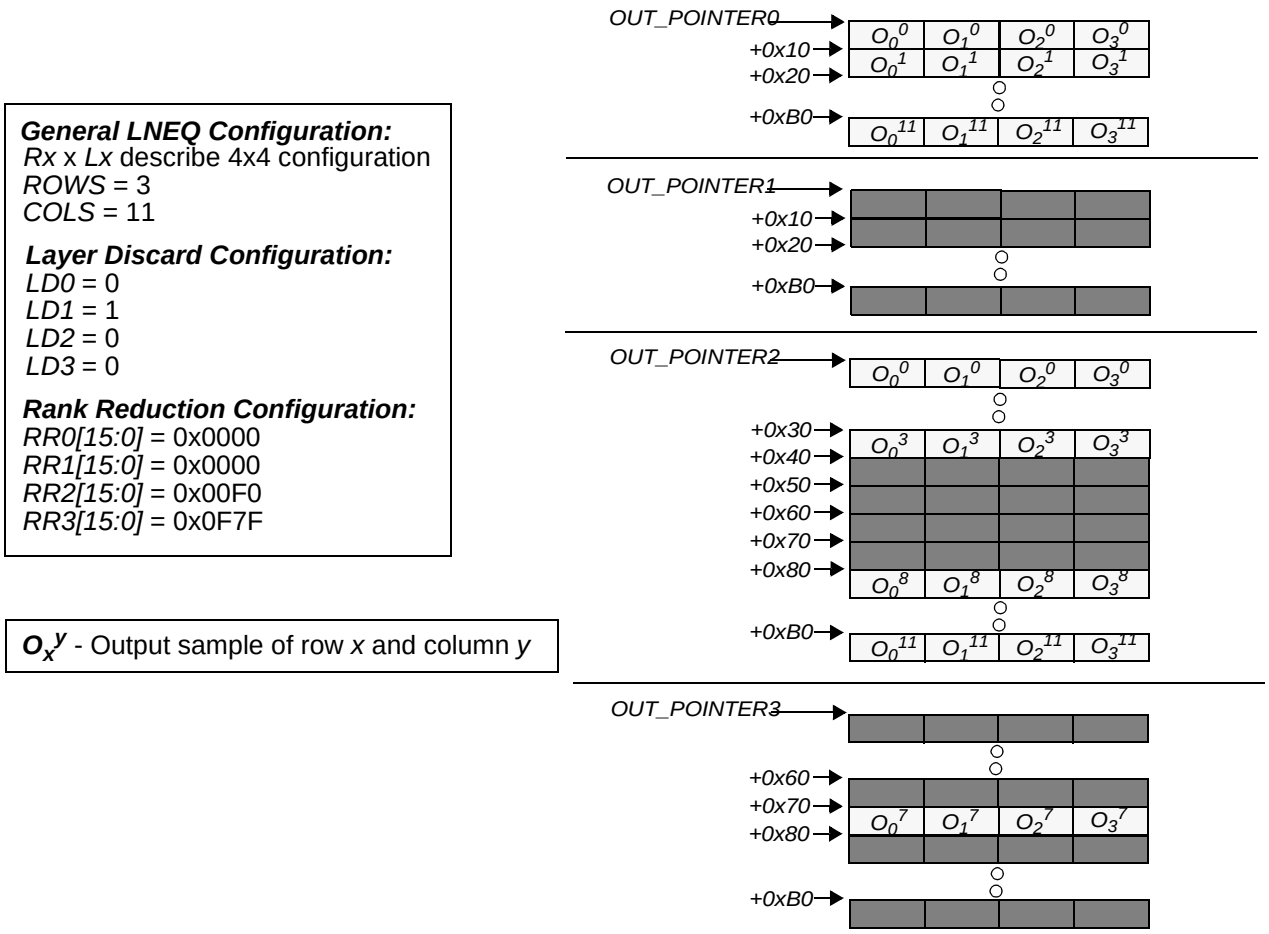


Figure 26-101. EQPE Output Data (mantissa) Structure for LNEQ Mode

The order of the data symbols in each column is starting from row #0 to row #ROWS where each sample contains the real part first following by the imaginary part as described in **Table 26-82**

Table 26-82. Output Samples Mantissas—LNEQ Mode

Memory line byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Sample	K				K + 1				K + 2				K + 3			
Value	*Real Part*		*Imaginary part*		*Real Part*		*Imaginary part*		*Real Part*		*Imaginary part*		*Real Part*		*Imaginary part*	

Each column of samples always starts on a new line. **Table 26-83** describe an example of output data in the system memory for the case of *ROWS* = 11

Table 26-83. Output samples arrangement example for *ROWS* = 11 (LNEQ mode)

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OUT_POINTERx	O_0^0				O_1^0				O_2^0				O_3^0			
+0x10	O_4^0				O_5^0				O_6^0				O_7^0			
+0x20	O_8^0				O_9^0				O_{10}^0				O_{11}^0			
+0x30	O_0^1				O_1^1				O_2^1				O_3^1			
...	...				...				...				...			
Value	*Real Part*		*Imaginary part*		*Real Part*		*Imaginary part*		*Real Part*		*Imaginary part*		*Real Part*		*Imaginary part*	

26.4.3.5.8.1.2 Scaled Samples Outputs for LNEQ

The MAPLE-B2 scales samples output data structure depends on the output alignment type as described in **Section 26.4.3.5.6.2.1**, *Output Samples Alignment in LNEQ Mode*.

26.4.3.5.8.1.3 Max Scale/User Defined Scale Output Alignment (LNEQ)

For Max Scale output alignment and User Defined output alignment (*ALIGNx* = 0,1), the output scale vector data structure is identical as in both cases the output scale vector for each layer includes a single scale sample for each column in the layer. In that case each layer's output scale vector structure is composed of single scale value (8 bits signed) for each column. The order of the scale values is starting from column #0 to column #*COLS*. If Rank Reduction feature is used (*RRx[y]*) then the reduced columns scale values is skipped in the same manner as the mantissa data (see **Section 26.4.3.5.8.1.1**), that is, the scale value location of the reduced column is skipped. If Layer Discard feature (*LDx*) is enabled for any of the layers, the MAPLE-B2 does not output the scale vector related to that layer.

The MAPLE-B2 outputs each of the layer's scale vector to the address calculated by EQPE BD parameters as follows:

Address<x> = *OUT_POINTER*<x>*16Bytes + *OUT_SCL_OFF*<x>*256Bytes

If (*OUT_SCL_OFF* = 0) the MAPLE-B2 output the scale vector at the next byte aligned address after the completion of writing the mantissa data as described above. **Table 26-84** describe an example the MAPLE-B2 output scale vectors structure for Max Scale and User Define output alignment with the Rank Reduction and Layer Discard parameters as described in **Figure 26-101**:

Table 26-84. Output Scale Vectors Example (Max Scale/ User Defined Scale in LNEQ Mode)

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*OUT_POINTER0**8B + *OUT_SCL_OFF0**256B	SCL_0	SCL_1	SCL_2	SCL_3	SCL_4	SCL_5	SCL_6	SCL_7	SCL_8	SCL_9	SCL_{10}	SCL_{11}				
*OUT_POINTER1**8B + *OUT_SCL_OFF1**256B																
*OUT_POINTER2**8B + *OUT_SCL_OFF2**256B	SCL_0	SCL_1	SCL_2	SCL_3					SCL_8	SCL_9	SCL_{10}	SCL_{11}				
*OUT_POINTER3**8B + *OUT_SCL_OFF3**256B								SCL_7								

26.4.3.5.8.1.4 No Output Alignment (LNEQ)

If No Alignment configuration (*ALIGNx* = 2) is used, the output scale vectors include scale value for each sample in each of the columns of each layer. The total number of scale values for each layer would be (*COLS*+1)x(*ROWS*+1). The order of the scale values in the scale vectors is identical to the order of the mantissa data as described above, that is, starting from the scale values of column #0 until the scale values of the last column (*#COLS*). The order of the scale values in each column is from index #0 to index *#ROWS*.

The scale values start address of each of the columns in the layer must be 4 bytes aligned. **Table 26-85** describe a scale vector example for *ROWS*=11:

Table 26-85. Output Scale Vector Example for *ROWS*=11 (No Alignment in LNEQ Mode)

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*OUT_POINTERx**8B + *OUT_SCL_OFFx**256B	SCL^0_0	SCL^0_1	SCL^0_2	SCL^0_3	SCL^0_4	SCL^0_5	SCL^0_6	SCL^0_7	SCL^0_8	SCL^0_9	SCL^0_{10}	SCL^0_{11}	SCL^1_0	SCL^1_1	SCL^1_2	SCL^1_3
*OUT_POINTERx**8B + *OUT_SCL_OFFx**256B + 0x10	SCL^1_4	SCL^1_5	SCL^1_6	SCL^1_7	SCL^1_8	SCL^1_9	SCL^1_{10}	SCL^1_{11}	SCL^2_0	SCL^2_1	SCL^2_2	SCL^2_3	SCL^2_4	SCL^2_5	SCL^2_6	SCL^2_7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

The start address of the scale vector for each layer and the vector structure in case of Rank Reduction or Layer Discard features are enabled is to as described in **Section 26.4.3.5.8.1.3**.

26.4.3.5.8.2 EQPE Outputs for MLEQ and TS modes

In MLEQ and TS modes, the output samples are the LLRs (Log-Likelihood Ratios) of the transmitted bits calculated according to **Equation 3** (See **Section 26.4.3.5.5.3**) for each layer in the job. Each output sample is composed of a 8 bit mantissa in 1q7 representation (*V*) and an 8-bit signed exponent (*EXP*). The value of the sample is: $V \times 2^{EXP}$.

In the EQPE output buffer, the mantissa is normalized, that is, mantissa begins with 2'b01 (for positive value) or 2'b10 (for negative value). An exception is when the LLR equals zero, in this case the mantissa is all zeros and the exponent holds the most negative possible value 0x80 (-128). For MLEQ and TS modes the EQPE supports 2 output alignment (scaling) types as described in **Section 26.4.3.5.6.2.2**, *Output Samples Alignment in MLEQ and TS Modes*.

The output size in this mode depends upon the number of sub-carriers (*(ROWS+1)* x (*COLS+1)*), the number of layers (*Lx*) and the modulation of each layer (*MOD<x>*). The output size for each layer is calculated as follows:

Mantissa Size = *(ROWS+1)* x (*COLS+1)* x *MODULATION*[1] x (1Byte mantissa)

The scale vectors (exponent) size of each layer is either one byte size (if User Defined Alignment is used) or equals to "Mantissa Size" as calculated above (if No Alignment is used).

The structure of the vectors (both mantissa and scales) of each layer is column by column successively starting from column #0 to column *#COLS*. The order of the samples in each column is from index #0 to index *#ROWS*.

The MAPLE-B2 output the mantissa vectors of each layer to the address pointed by the *OUT_POINTERx* field of the EQPE BD, and the scale vectors to the address calculated as follows:

Address<x> = *OUT_POINTER*<x>*16Bytes + *OUT_SCL_OFF*<x>*256Bytes

26.4.3.5.8.3 EQPE Outputs for MIV

In Matrix Inversion Mode (MIV), the output samples are the entries of the inverted matrix, calculated according to **Equation 5** and **Equation 6** (see **Section 26.4.3.5.5.6**). Each output sample is composed of a 32 bit mantissa (16I,16Q) in 1q15 representation (*V*) and an 8-bit signed exponent (EXP). The value of the sample is: $V \times 2^{EXP}$.

In the EQPE output buffer the mantissa is normalized, that is, at least one of the real or imaginary parts begins with 2'b01 (for positive value) or 2'b10 (for negative value). An exception is when both real and imaginary parts equal zero, in this case the exponent holds the most negative possible value 0x80 (-128). For MIV mode the EQPE supports 4 output alignment (scaling) types as described in **Section 26.4.3.5.6.2.3**, *Output Samples Alignment in MIV Mode*.

1. The MODULATION is 2 for QPSK, 4 for 16QAM and 6 for 64QAM. Each layer may have a different modulation and thus the size of the output may differ between layers.

26.4.3.5.8.3.1 Data Samples (mantissa) Outputs for MIV

The EQPE inverted matrices output data structure depends on the Matrix type. If Full matrix inversion is required (*FH* = 1), then both the input and output matrix types are Full matrix. If Hermitian matrix inversion is required (*FH* = 0), then both the input and output matrix types are Hermitian. The output data structure of the mantissas (for both Full and Hermitian types of matrix) is identical to the input data structure of the matrices in MIV mode and is described in details in **Section 26.4.3.5.4.4.1**, *M Matrix Input Data Structure*.

26.4.3.5.8.3.2 Scale Samples (exponent) Outputs for MIV

The scale samples vector output data structure depends on the configuration of the output samples alignment used for the job. The following sections describe the scale vector output data structure for each of the supported alignments.

26.4.3.5.8.3.3 Max Scale Output Alignment for MIV

In this case the EQPE output single scale value (8 bits signed) for each matrix. The order of the values in the vector is from matrix #0 to matrix #*ROWS*. The address of the scale vector is calculated by MAPLE-B2 as follows:

Address = *OUT_POINTER*0*16Bytes + *OUT_SCL_OFF*0*256Bytes

If (*OUT_SCL_OFF0* = 0) the MAPLE-B2 output the scale vector at the next byte aligned address after the completion of writing the mantissa data of the last matrix as described above.

26.4.3.5.8.3.4 User Defined/Global Max Scale Output Alignment (MIV)

In this case the EQPE output single scale value for all the job. The address location of the 8 bits scale value is as described in the Max Scale Output Alignment case (see **Section 26.4.3.5.8.3.3**).

26.4.3.5.8.3.5 No Output Alignment (MIV)

In this case the EQPE output a scale value for each element of each of the inverted matrices. The scale vector address is calculated by MAPLE-B2 in the same manner as in **Section 26.4.3.5.8.3.3**, *Max Scale Output Alignment for MIV*. The structure of the output vector depend on both the matrices dimensions (*Lx*) and the matrices type, that is, Hermitian or Full (*FH*). Assuming $e^{i}_{x,y}$ is an element of matrix i (i = 0...*ROWS*), located in row *x* and column *y* (*x*,*y* = 0...Lx), the following tables describe the scale vectors structure for the different cases.

Table 26-86. Output Scale Vector Structure of Full MIV 1x1 (*FH*=1, *Lx*=0)

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*OUT_POINTER0**16B + *OUT_SCL_OFF0**256B	$e^{0}_{0,0}$	$e^{1}_{0,0}$	$e^{2}_{0,0}$	$e^{3}_{0,0}$	$e^{4}_{0,0}$	...	...	...								

Table 26-87. Output Scale Vector Structure of Full MIV 2x2 (*FH*=1, *Lx*=1)

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*OUT_POINTER0**16B + *OUT_SCL_OFF0**256B	$e^0_{0,0}$	$e^0_{0,1}$	$e^0_{1,0}$	$e^0_{1,1}$	$e^1_{0,0}$	$e^1_{0,1}$	$e^1_{1,0}$	$e^1_{1,1}$	$e^2_{0,0}$	$e^2_{0,1}$	$e^2_{1,0}$	$e^2_{1,1}$	...	....		

Table 26-88. Output Scale Vector Structure of Full MIV 3x3 (*FH*=1, *Lx*=2)

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*OUT_POINTER0**16B + *OUT_SCL_OFF0**256B	$e^0_{0,0}$	$e^0_{0,1}$	$e^0_{0,2}$	X	$e^0_{1,0}$	$e^0_{1,1}$	$e^0_{1,2}$	X	$e^0_{2,0}$	$e^0_{2,1}$	$e^0_{2,2}$	X	X	X	X	X
+0x10	$e^1_{0,0}$	$e^1_{0,1}$	$e^1_{0,2}$	X	$e^1_{1,0}$	$e^1_{1,1}$	$e^1_{1,2}$	X	$e^1_{2,0}$	$e^1_{2,1}$	$e^1_{2,2}$	X	X	X	X	X
+0x20	$e^2_{0,0}$	$e^2_{0,1}$	$e^2_{0,2}$	X	$e^2_{1,0}$	$e^2_{1,1}$	$e^2_{1,2}$	X	$e^2_{2,0}$	$e^2_{2,1}$	$e^2_{2,2}$	X	X	X	X	X
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 26-89. Output Scale Vector Structure of Full MIV 4x4 (*FH*=1, *Lx*=3)

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*OUT_POINTER0**16B + *OUT_SCL_OFF0**256B	$e^0_{0,0}$	$e^0_{0,1}$	$e^0_{0,2}$	$e^0_{0,3}$	$e^0_{1,0}$	$e^0_{1,1}$	$e^0_{1,2}$	$e^0_{1,3}$	$e^0_{2,0}$	$e^0_{2,1}$	$e^0_{2,2}$	$e^0_{2,3}$	$e^0_{3,0}$	$e^0_{3,1}$	$e^0_{3,2}$	$e^0_{3,3}$
+0x10	$e^1_{0,0}$	$e^1_{0,1}$	$e^1_{0,2}$	$e^1_{0,3}$	$e^1_{1,0}$	$e^1_{1,1}$	$e^1_{1,2}$	$e^1_{1,3}$	$e^1_{2,0}$	$e^1_{2,1}$	$e^1_{2,2}$	$e^1_{2,3}$	$e^1_{3,0}$	$e^1_{3,1}$	$e^1_{3,2}$	$e^1_{3,3}$
+0x20	$e^2_{0,0}$	$e^2_{0,1}$	$e^2_{0,2}$	$e^2_{0,3}$	$e^2_{1,0}$	$e^2_{1,1}$	$e^2_{1,2}$	$e^2_{1,3}$	$e^2_{2,0}$	$e^2_{2,1}$	$e^2_{2,2}$	$e^2_{2,3}$	$e^2_{3,0}$	$e^2_{3,1}$	$e^2_{3,2}$	$e^2_{3,3}$
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 26-90. Output Scale Vector Structure of Hermitian MIV 1x1 (*FH*=0, *Lx*=0)

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*OUT_POINTER0**16B + *OUT_SCL_OFF0**256B	$e^0_{0,0}$	$e^1_{0,0}$	$e^2_{0,0}$	$e^3_{0,0}$	$e^4_{0,0}$	...	...	...								

Table 26-91. Output Scale Vector Structure of Hermitian MIV 2x2 (*FH*=0, *Lx*=1)

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*OUT_POINTER0**16B + *OUT_SCL_OFF0**256B	$e^0_{0,0}$	$e^0_{1,1}$	$e^0_{0,1}$	X	$e^1_{0,0}$	$e^1_{1,1}$	$e^1_{0,1}$	X	$e^2_{0,0}$	$e^2_{1,1}$	$e^2_{0,1}$	X	$e^3_{0,0}$	$e^3_{1,1}$	$e^3_{0,1}$	X
+0x10	$e^4_{0,0}$	$e^4_{1,1}$	$e^4_{0,1}$	X	$e^5_{0,0}$	$e^5_{1,1}$	$e^5_{0,1}$	X	$e^6_{0,0}$	$e^6_{1,1}$	$e^6_{0,1}$	X	$e^7_{0,0}$	$e^7_{1,1}$	$e^7_{0,1}$	X
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 26-92. Output Scale Vector Structure of Hermitian MIV 3x3 (*FH*=0, *Lx*=2)

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*OUT_POINTER0**16B + *OUT_SCL_OFF0**256B	$e^0_{0,0}$	$e^0_{1,1}$	$e^0_{2,2}$	X	$e^0_{0,1}$	$e^0_{0,2}$	X	$e^0_{1,2}$	X	X	X	X	X	X	X	X
+0x10	$e^1_{0,0}$	$e^1_{1,1}$	$e^1_{2,2}$	X	$e^1_{0,1}$	$e^1_{0,2}$	X	$e^1_{1,2}$	X	X	X	X	X	X	X	X
+0x20	$e^2_{0,0}$	$e^2_{1,1}$	$e^2_{2,2}$	X	$e^2_{0,1}$	$e^2_{0,2}$	X	$e^2_{1,2}$	X	X	X	X	X	X	X	X
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 26-93. Output Scale Vector Structure of Hermitian MIV 4x4 (*FH*=0, *Lx*=3)

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*OUT_POINTER0**16B + *OUT_SCL_OFF0**256B	$e^0_{0,0}$	$e^0_{1,1}$	$e^0_{2,2}$	$e^0_{3,3}$	$e^0_{0,1}$	$e^0_{0,2}$	$e^0_{0,3}$	$e^0_{1,2}$	$e^0_{1,3}$	$e^0_{2,3}$	X	X	X	X	X	X
+0x10	$e^1_{0,0}$	$e^1_{1,1}$	$e^1_{2,2}$	$e^1_{3,3}$	$e^1_{0,1}$	$e^1_{0,2}$	$e^1_{0,3}$	$e^1_{1,2}$	$e^1_{1,3}$	$e^1_{2,3}$	X	X	X	X	X	X
+0x20	$e^2_{0,0}$	$e^2_{1,1}$	$e^2_{2,2}$	$e^2_{3,3}$	$e^2_{0,1}$	$e^2_{0,2}$	$e^2_{0,3}$	$e^2_{1,2}$	$e^2_{1,3}$	$e^2_{2,3}$	X	X	X	X	X	X
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

26.4.3.5.8.4 EQPE Output Data Structure - Summary

The following table summarize the EQPE output data vectors for each of its modes. The table summarize the sizes of each output vector in each mode.

Table 26-94. EQPE Output vectors sizes (in bytes)

Vector Type	LNEQ mode	MLEQ mode	TS mode	MIV mode
Output Samples	(*COLS*+1)*(*ROWS*+1)*(*Lx*+1)*4	per layer <*x*>=0,...,*Lx*: (*ROWS*+1)*(*COLS*+1)**MOD*<*x*> where *MOD*<*x*> = 2/4/6 for QPSK/16QAM/64QAM		if *FH*=0 (*ROWS*+1)*(*Lx*+1)*(*Lx*+1)*2 if *FH*=1 (*ROWS*+1)*(*Lx*+1)*(*Lx*+1)*4

Output Scale	per layer <x>=0,...,*Lx*:
	if *ALIGN*<*x*>=0:
	(*COLS*+1)
	if *ALIGN*<*x*>=1
	1 Byte
	if *ALIGN*<*x*>=2:
	(*COLS*+1)*(*ROWS*+1)
		per layer <*x*>=0,...,*Lx*:
		if *ALIGN*<*x*>=1
		1 Byte
		if *ALIGN*<*x*>=2:
		(*ROWS*+1)*(*COLS*+1)**MOD*<*x*>
		where *MOD*<*x*> = 2/4/6 for QPSK/16QAM/64QAM
				if *ALIGN0*=0:
				(*ROWS*+1)
				if *ALIGN0*=1:
				1 Byte
				if *ALIGN0*=2:
				(*ROWS*+1)*(*Lx*+1)*(*Lx*+1)
				if *ALIGN0*=3:
				1 Byte

26.4.3.5.9 Pre/Post DFT/iDFT Processing

For EQPE LNEQ mode, the MAPLE-B2 is capable of internally concatenating a DFT/iDFT processing to the EQPE processing thus significantly reduce the overall system latency and the required control interface towards the MAPLE-B2. The DFT/iDFT processing assist is executed using the eFTPE engines in the MAPLE-B2. For details on the eFTPE engines see **Section 26.4.3.3**, *eFTPE FFT/iFFT/DFT/iDFT Operation*.

There are two such concatenation options supported by the MAPLE-B2:

- iDFT processing of the EQPE outputs. In this flow every output column of the EQPE sub-carrier grid is being internally transferred into an eFTPE engine for iDFT processing. On iDFT processing completion, the MAPLE-B2 outputs the results into the system memory as defined by the EQPE BD parameters. Enabling this processing flow is done by setting the *POST_IDFT* bit of the EQPE BD.

- DFT processing of the EQPE feedback layer inputs. In this flow the MAPLE-B2 executes DFT processing on the input feedback layer to be cancelled by the EQPE. This processing is done prior to the EQPE processing. Each column of the input feedback layer is first being processed by an eFTPE engine (DFT processing) and only then internally transferred into the EQPE feedback buffer for EQPE processing. Enabling this processing flow is optional only if Feedback Cancellation is enabled (*F_EN*) and is done by setting the *F_DFT* bit of the EQPE BD.

Enabling both processing flows, that is, executing pre DFT processing on the input layer to cancel and post iDFT processing on the output layer is allowable.

The following sections describes in details the internal processing flow of these modes with respect to the EQPE BD parameters configurations.

26.4.3.5.9.1 Pre DFT Processing

When the pre DFT processing flow is enabled (by setting the *F_DFT* and *F_EN* bits of the EQPE BD), the MAPLE-B2 uses the following EQPE BD parameters to set an eFTPE engine to execute the pre DFT processing of the feedback layer:

- *COLS* - As each column of the input layer requires DFT processing, the *COLS* field is used by the MAPLE-B2 to determined the required number of DFT jobs to be processed.
- *ROWS* - The number of rows in each of the column is used to determine the DFT size to be processed for each of the columns in the layer.
- *F_DFT* - Indicates the MAPLE-B2 that the job type is DFT (and not iDFT).
- *SCL_PTR_BA* and *F_SCL_OFFSET* - Points to the address in the system memory where the scale value for each of the Feedback input columns is located. The scale value of each input column is used as the input exponent value of the eFTPE engine as described in **Section 26.4.3.3.9.4.5**, *Input Exponent*.

Additional settings for the eFTPE when executing pre EQPE processing are:

- Adaptive Scaling (see **Section 26.4.3.3.9.4.3**, *Adaptive Scaling*) - The eFTPE is always configured to work in Adaptive Scaling mode to maximize execution precision.
- Adaptive Input Scaling (see **Section 26.4.3.3.9.4.7**, *Adaptive Input Scaling*) - The eFTPE is always configured to work in Adaptive Input Scaling mode to maximize execution precision.

After EQPE BD parsing the MAPLE-B2 configure one of the eFTPE engines as described above and fetches the Feedback data from the address pointed by the *F_POINTER* field of the EQPE BD and into the chosen eFTPE engines' input buffer. For each of the processed columns in the eFTPE, when the DFT processing is completed the MAPLE-B2 transfer the output data out of the eFTPE output buffer and into the Feedback input buffer of the EQPE. The eFTPE Adaptive

Overall Scaling Status indication (see **Section 26.4.3.3.9.4.3**, *Adaptive Scaling*) is used as the Feedback input scale value of the EQPE.

26.4.3.5.9.2 Post iDFT Processing

When the post iDFT processing flow is enabled (by setting the *POST_IDFT* bit of the EQPE BD), the MAPLE-B2 uses the following EQPE BD parameters to set an eFTPE engine to execute the post iDFT processing of the EQPE output layers:

- *Lx and COLS* - Indicated the number of columns and number of layers to be output from the EQPE and requires iDFT post processing. As each column of each layer requires different iDFT processing the total number of iDFT processing is *Lx* x *COLS*. If Layer Discard (*LDx*) is enabled in the EQPE BD for any of the layers, then no iDFT processing occurs for this layer. If Rank Reduction (*RRx[y]*) is enabled for any of the columns of any of the output layers, then no iDFT processing occurs for the reduced column.
- *ROWS* - The number of rows in each of the column of the layers is used to determine the DFT size to be processed for each of the columns of the layers.
- *POST_IDFT* - Indicates the MAPLE-B2 that the job type is iDFT (and not DFT).
- *ALIGNx* - Indicates for each layer if the eFTPE is to work with configured Overall Scaling enabled or not (see **Section 26.4.3.3.9.4.4**, *Overall Scaling Amount Programming*). If the *ALIGNx* value is 0 (Max Scale Output Alignment) the Overall Scaling configuration in the eFTPE is disabled and the *OUT_SCL_OFFx* field of the EQPE BD points to the accumulated results of the EQPE + eFTPE scale values for each of the output columns of each layer. If the *ALIGNx* value is 1 (User Defined Output Alignment), the EQPE is activated with Max Scale Output Alignment mode (*ALIGNx*=0), but the eFTPE Overall Scaling configuration is enabled and the overall scaling value used for the eFTPE is the *ALIGN_VALx* field of the EQPE BD. Such flow assures maximum precision while the final EQPE+eFTPE processing results are as required in the EQPE BD.
- *PST_SCLR* - If set, the eFTPE Scalar Post Multiplication is enabled (see **Section 26.4.3.3.6**, *Scalar Post Multiplication in eFTPE*) for each iDFT result. The MAPLE-B2 supports a different scalar value for each iDFT job, that is, *Lx* x *COLS* different values. The scalar values used for the post multiplications are expected in the system memory as pointed by the *PST_SCLR_PTR* fields of the EQPE BD.
- *PST_SCLR_PTR* - The pointer to the system memory of the scalar values (*COLS* x *Lx* values) to be multiplied with the iDFT results if the *PST_SCLR* field of the EQPE BD is set. The *Lx* x *COLS* scalar values are expected successively hence the size of the data structure expected by MAPLE-B2 is:

  $4_{bytes} \times L_x \times COLS$

- *PST_VEC* - If set, the eFTPE vector post multiplication is enabled (see **Section 26.4.3.3.5.2**, *Post-Multiplication processing support in the eFTPE*). The MAPLE-B2 supports a different vector for each of the EQPE layers (*Lx*). It means that all iDFT jobs

related to the same layer uses the same vector. The Post Multiplication vectors are fetched by the MAPLE-B2 from the address detailed in the *PST_VEC_PTR* field of the EQPE BD.

- *PST_VEC_PTR* - Pointer in the system memory of the post multiplication vectors (*Lx* vectors) to be multiplied with the iDFT results if the *PST_VEC* field of the EQPE BD is set. The *Lx* post multiplication vectors are expected successively in the system memory hence the size of the data structure expected by MAPLE-B2 is:

  $4_{bytes} \times L_x \times ROWS$

Note: When Post iDFT Processing is enabled, the “No alignment” option (*ALIGNx*=2) must not be used as the eFTPE does not support floating point inputs.

Note: One addition setting for the eFTPE when executing post EQPE processing is Adaptive Scaling (see **Section 26.4.3.3.9.4.3**, *Adaptive Scaling*) - The eFTPE is always configured to work in Adaptive Scaling mode to maximize execution precision.

After EQPE BD parsing, the MAPLE-B2 configures the EQPE according to the BD parameters and launches the job into the EQPE. When the internal EQPE output_buffer_ready/job_done indication is received by MAPLE-B2, it configures an eFTPE engines as described above and transfer the EQPE output results into the eFTPE input buffer for iDFT processing. On eFTPE internal job done indication the MAPLE-B2 use the *OUT_POINTERx* parameter of the EQPE BD in order determined the output address of the results. The Scale values of each column of each layer after the iDFT processing are accumulated and are being output as described in **Section 26.4.3.5.8.1.3**, *Max Scale/User Defined Scale Output Alignment (LNEQ)*.

If the *INT_EN* bit of the EQPE BD is set, the MAPLE-B2 generates the relevant interrupt only after the completion of all the iDFT processing of all the layers.

26.4.3.5.10 EQPE Status Indications

The EQPE reports a matrix singularity indication. The report includes how many singular (or close to singular) matrices have been processed in the job.

The singularity check is done by configuring a threshold value in the EQPE Threshold Register (EQ_THRESH). If the ***R*** matrix (the upper-right matrix computed by the QRD) has one or more diagonal elements (the diagonal elements are positive real numbers) with exponents which are smaller than the threshold value, the matrix is considered to be singular (close to singular) and a dedicated counter in the EQPE is increment by 1. When the EQPE finishes the job, this counter, which holds the number of (close to) singular matrices processed in the job, is copied to the EQPE BD *SING* status field.

26.4.3.5.11 EQPE ECC Support

As part of the MAPLE-B2 ECC support (see **Section 26.4.3.1.1, *Memory Error Correction/Detection Support***), the EQPE include a status register which specify the exact memory module in which the ECC error has occurred.

If the general ECC error indication of the MAPLE-B2 is asserted, and by reading the PSIF PIC Event Register 2 (PSPICER2), the EQPE ECC indication bit is asserted, it means that the source of the ECC error is in the EQPE. Reading the EQPE ECC Event Register (EQ_ECCEVENT) gives indication as to which internal EQPE memory was the source of the ECC error. The EQ_ECCEVENT register includes a general ECC error bit ([31]) and a per memory indication bits ([21:0]). Resetting the EQPE general ECC error bit and relevant memory indication bits is done by writing ‘1’ to the relevant bits.

The following summarize the required actions to identify the source of the ECC interrupt:

1. Read the PSIF PIC Event Register 2 (PSPICER2) to detect which PE is the source of the ECC interrupt
2. If the EQ_ECC bit is asserted, then the source of the ECC interrupt is in the EQPE.
3. Read the EQ_ECCEVENT register to identify the exact memory module which was the source of the ECC interrupt.

To reset the ECC interrupt, the following action must occur:

1. Clear the general ECC error bit and the memory dedicated ECC error bit indications from the EQPE ECC Event Register (EQ_ECCEVENT) by writing ‘1’ to the relevant bits.
2. Clear the *EQ_ECC* error bit from the PSIF PIC Event Register 2 (PSPICER2) by writing ‘1’ to the bit.

26.4.3.6 CRPE Uplink/Downlink UMTS Chip Rate Processing Operation

The CRPE includes the following three processing units:

- CRPE Uplink Batch (CRPE-ULB). See **Section 26.4.3.6.1**.
- CRPE Uplink Fast (CRPE-ULF). See **Section 26.4.3.6.2**
- CRPE Downlink (CRPE-DL). See **Section 26.4.3.6.3**

The following subsections describe the functional operation of each of these units, and **Section 26.4.3.6.4** describe the CRPE internal reset flow which allows internal reset for the CRPE modules only in case of loosing synchronization with the Antenna I/F.

26.4.3.6.1 Chip Rate Uplink Batch Processing Element

The MAPLE-B2 supports UMTS (TS 25.211 and TS 25.213) Batch Chip Rate processing of data and control channels for Uplink as described in **Section 26.2**, *MAPLE-B2 Features*. These operations are executed using the Chip Rate- Uplink Batch Processing Element (CRPE-ULB).

26.4.3.6.1.1 CRPE-ULB Initialization

The CRPE-ULB initialization is composed of five stages:

1. CRPE-ULB Mode configuration parameter done by the API during the MAPLE-B2 initialization.
2. CRPE-ULB Parameters initialization done by the host after MAPLE-B2 initialization by API is completed and before the processing beginning of CRPE-ULB.
3. CRPE-ULB Core Descriptors initialization for each defined Group done by the host after the parameters initialization is completed.
4. CRPE-ULB Registers initialization done by the host after MAPLE-B2 initialization by API is completed and before the processing beginning of CRPE-ULB.
5. `Maple_crpe_ulb_init` routine activation which triggers the CRPE-ULB to start scanning the valid groups to process.

Note: It is recommended to follow the order of the above initialization steps to assure correct operation.

26.4.3.6.1.1.1 CRPE-ULB API Initialization

During the MAPLE-B2 initialization (see *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)), the CRPE-ULB mode configuration parameter in the API must be initialized: See **Section 26.5.2.5**, *CRPE-ULB Mode Configuration Parameter (CRUBMCP)*, on page 26-328 for programming details.

26.4.3.6.1.1.2 CRPE-ULB Parameters Initialization

After the MAPLE-B2 initialization routine and before the beginning of the processing by the CRPE-ULB, the following CRPE-ULB parameters must be initialized:

- MAPLE CRPE-ULB Physical Channel <x> Base Address Parameter (MCUBPCHxBAP): Should be initialized for each active channel. Indicates the base address in the system memory where the MAPLE-B2 is to output the channels data. See **Section 26.5.3.5.2.1**, *MAPLE CRPE-ULB Physical Channel <x> Base Address Parameter (MCUBPCHxBAP)*, on page 26-351 for details.
- MAPLE CRPE-ULB Physical Channel <x> Size Parameter (MCUBPCHxSZP): Should be initialized for each active channel. Indicates the size of the output buffer in the system

memory. See **Section 26.5.3.5.2.2**, *MAPLE CRPE-ULB Physical Channel <x> Size Parameter (MCUBPCHxSZP)*, on page 26-352 for details.

- MAPLE CRPE-ULB Physical Channel <x> Write Pointer Parameter (MCUBPCHxWPP): Should be initialized with the value of *MCUBPCHxBAP[PCHxBA]* for each active channel. Maintained by MAPLE-B2 and points to the address where next data is to be written in the output buffer. See **Section 26.5.3.5.2.3**, *MAPLE CRPE-ULB Physical Channel <x> Write Pointer Parameter (MCUBPCHxWPP)*, on page 26-353 for details.
- MAPLE CRPE-ULB Physical Channel <x> Output Buffer Interrupt Configuration Parameter (MCUBPCHxOBICP): Should be initialized for each active channel. Indicates the interrupt methodology for each channel. See **Section 26.5.3.5.2.4**, *MAPLE CRPE-ULB Physical Channel <x> Output Buffer Interrupt Configuration Parameter (MCUBPCHxOBICP)*, on page 26-354 for details.
- MAPLE CRPE-ULB Group Configuration parameters: Should be initialized for each active group. Includes all relevant parameters for each of the defined groups. See **Section 26.5.3.5.2.5**, *MAPLE CRPE-ULB Group <x> Configuration 1 Parameter (MCUBGxC1P)*, on page 26-355, **Section 26.5.3.5.2.6**, *MAPLE CRPE-ULB Group <x> Configuration 2 Parameter (MCUBGxC2P)*, on page 26-356, **Section 26.5.3.5.2.7**, *MAPLE CRPE-ULB Group <x> Configuration 3 Parameter (MCUBGxC3P)*, on page 26-357, and **Section 26.5.3.5.2.8**, *MAPLE CRPE-ULB Group <x> Configuration 4 Parameter (MCUBGxC4P)*, on page 26-358 for details.
- MAPLE CRPE-ULB Antenna <x> Descriptor Parameter (MCUBANTxDP): Should be initialized for each active Antenna. Holds the base address in the system memory of each active antenna. See **Section 26.5.3.5.2.9**, *MAPLE CRPE-ULB Antenna <x> Descriptor Parameter (MCUBANTxDP)*, on page 26-359 for details.
- MAPLE CRPE-ULB Configuration Parameter (MCUBCP): Holds general configuration related to the CRPE-ULB operation mode. See **Section 26.5.3.5.2.10**, *MAPLE CRPE-ULB Configuration Parameter (MCUBCP)*, on page 26-360 for details.
- MAPLE CRPE-ULB Output Buffer Interrupt Configuration Parameter (MCUBOBICP): Describes the general output buffer interrupt configuration. See **Section 26.5.3.5.2.11**, *MAPLE CRPE-ULB Output Buffer Interrupt Configuration Parameter (MCUBOBICP)*, on page 26-361 for details.

26.4.3.6.1.1.3 CRPE-ULB Core Descriptors Initialization

After the Groups definition is completed during the parameters initialization, each Group's Core Descriptors should be initialized. The Core Descriptors data structure is detailed in **Section 26.5.4.6.1**, *CRPE-ULB Core Descriptors*. The number of Core Descriptors to be initialized for each defined Group is determined by the Group's number of Cores (*MCUBGxC1P[GxNOC]*).

26.4.3.6.1.1.4 CRPE-ULB Registers Initialization

After the MAPLE-B2 initialization routine and before the beginning of the processing by the CRPE-ULB, the following CRPE-ULB registers must be initialized:

- CRPE-ULB Interpolation Weights Configuration Registers: Holds the interpolation weights for CRPE-ULB Interpolation operation. See **Section 26.5.5.5.1**, *CRPE-ULB Interpolation Weights 1 Sample <x> Configuration Register (CRUBIW1SxCR)*, on page 26-525 and **Section 26.5.5.5.2**, *CRPE-ULB Interpolation Weights 2 Sample <x> Configuration Register (CRUBIW2SxCR)*, on page 26-526 for details.
- CRPE-ULB Group First Antenna <x> Configuration Register (CRUBGFAxCR): Holds the configuration values of the first antenna of each group. **Section 26.5.5.5.3**, *CRPE-ULB Group First Antenna <x> Configuration Register (CRUBGFAxCR)*, on page 26-527 for details.
- CRPE-ULB Group Number Of Antenna <x> Configuration Register (CRUBGNOAxCR): Holds the number of antennas for each group. See **Section 26.5.5.5.4**, *CRPE-ULB Group Number Of Antenna <x> Configuration Register (CRUBGNOAxCR)*, on page 26-528 for details.

26.4.3.6.1.1.5 Maple_crpe_ulb_init Routine Activation

Once all internal configuration is completed (as indicated in **Section 26.4.3.2.6.1.1** through **Section 26.4.3.2.6.1.4**), and the CRPE-ULB can start with data processing, the `Maple_crpe_ulb_init` routine must be initialized as described in **Section 26.4.5**, *MAPLE-B2 Internal Task Control*. This routine enable the CRPE-ULB internal interrupt assertion (towards the internal RISC engines), thus allowing them to start scanning for valid Groups for processing.

26.4.3.6.1.2 CRPE-ULB Modes of Operation

The CRPE-ULB supports the following modes of operation set during the initialization sequence as described in **Section 26.4.3.6.1.1**, *CRPE-ULB Initialization*:

26.4.3.6.1.2.1 Interpolation Mode

The CRPE-ULB allows 4 interpolation modes, depending on *INT_MODE* of the CRUBMCP initialization parameter:

Table 26-95. CRPE-ULB Interpolation Mode

CRUBMCP [INT_MODE]	MCUBCP [IIL]	Description	Number of Antennas supported if DBL_ANT_EN=0	Number of Antennas supported if DBL_ANT_EN=1
0	0	System memory contains OVSi (i=0,8) and no internal interpolation is performed	48	96
0	1	System memory contains OVSi (i=0,4,8,12) and no internal interpolation is performed	24	48

Table 26-95. CRPE-ULB Interpolation Mode (Continued)

CRUBMCP [INT_MODE]	MCUBCP [IIL]	Description	Number of Antennas supported if DBL_ANT_EN=0	Number of Antennas supported if DBL_ANT_EN=1
0	2	System memory contains OVSi (i=0,2,4,6,8,10,12,14) and no internal interpolation is performed	12	24
0	3	System memory contains OVSi (i=0...15) and no internal interpolation is performed	6	NA
1	X	System memory contains OVS0 and OVS8 and internal interpolation brings this to OVSi (i=0,8) or (i=0,4,8,12) or (i=0,2,4,6,8,10,12,14) or (i=0...15) depending on the fingers FOFFSET_L value.	48	96

26.4.3.6.1.2.2 Delay Spread Mode

The CRPE-ULB allows 2 modes of operation with regard to maximum delay spread between fingers as defined in the CRUBMCP initialization parameter:

- *DLY_SPRD* = 0; Delay spread up to 256 chips
- *DLY_SPRD* = 1; Delay spread up to 512 chips

The 2 modes differ in their performance, as described under **Section 26.4.3.6.1.5.2, *Finger Commands***.

Note: In case of larger than 512 chips delay between fingers, user should define multiple ULB processing channels for this specific PCH. THis is a typical case of RRH (Remote Radio Head) configurations.

26.4.3.6.1.2.3 Finger Combining Modes

The CRPE-ULB allows 3 finger combining modes determined by *FC_MODE* of the CRUBMCP initialization parameter:

- Finger combining bypass where each finger's resulting symbol is represented as a 32 bit (16I,16Q) value.
- Finger combining where resulting symbol is represented as 16-bit fixed point value with user defined scale.
- Finger combining where the resulting symbol is represented as a 6 bit exponent (extended to 16bits) with 16 bit mantissa with dynamic scaling.

For details on Finger combining see **Section 26.4.3.6.1.12**, *Finger Combining*.

26.4.3.6.1.2.4 One Output Buffer Mode

The CRPE-ULB supports two modes of operation with regard to its output buffer configuration as described in the CRUBMCP initialization parameter:

- *OOB* = 0; A job composed of a group of antennas should not exceed 4096 symbols.
- *OOB* = 1; A job composed of a group of antennas should not exceed 8192 symbols.

For details on the output data modes see **Section 26.4.3.6.1.13**, *Output Buffer Capacity*.

26.4.3.6.1.3 CRPE-ULB General Processing Flow

The CRPE Uplink Batch processing is fed by up to 96 antennas along with finger and physical channel commands from the system memory using MAPLE-B2 DMA, processes them, and outputs them to separate physical channel symbol rings.

Figure 26-102 illustrates the data structures used for the uplink batch processing:

Batch jobs are divided to groups. The commands belonging to the same group as well as their associated antennas are processed as one batch job on one single sub slot. That means that the VCOP is waiting till there’s enough data in all antennas belonging to the group to process their next sub slot and then reads the finger/physical channel commands from the system memory, process those commands followed by processing the antenna data.

Splitting the work into groups can be useful for example in the case where the same antenna is being processed more than once (cancellation schemes), and on different time points. In that case two antenna numbers shall be used for processing the same antenna data.

As shall be described in **Section 26.4.3.6.1.13**, *Output Buffer Capacity* the output buffer of the CRPE-ULB accelerator is limited. Splitting the work into several groups allows the MAPLE-B2 to empty the internal output buffer into the system before continuing to the job of the next group.

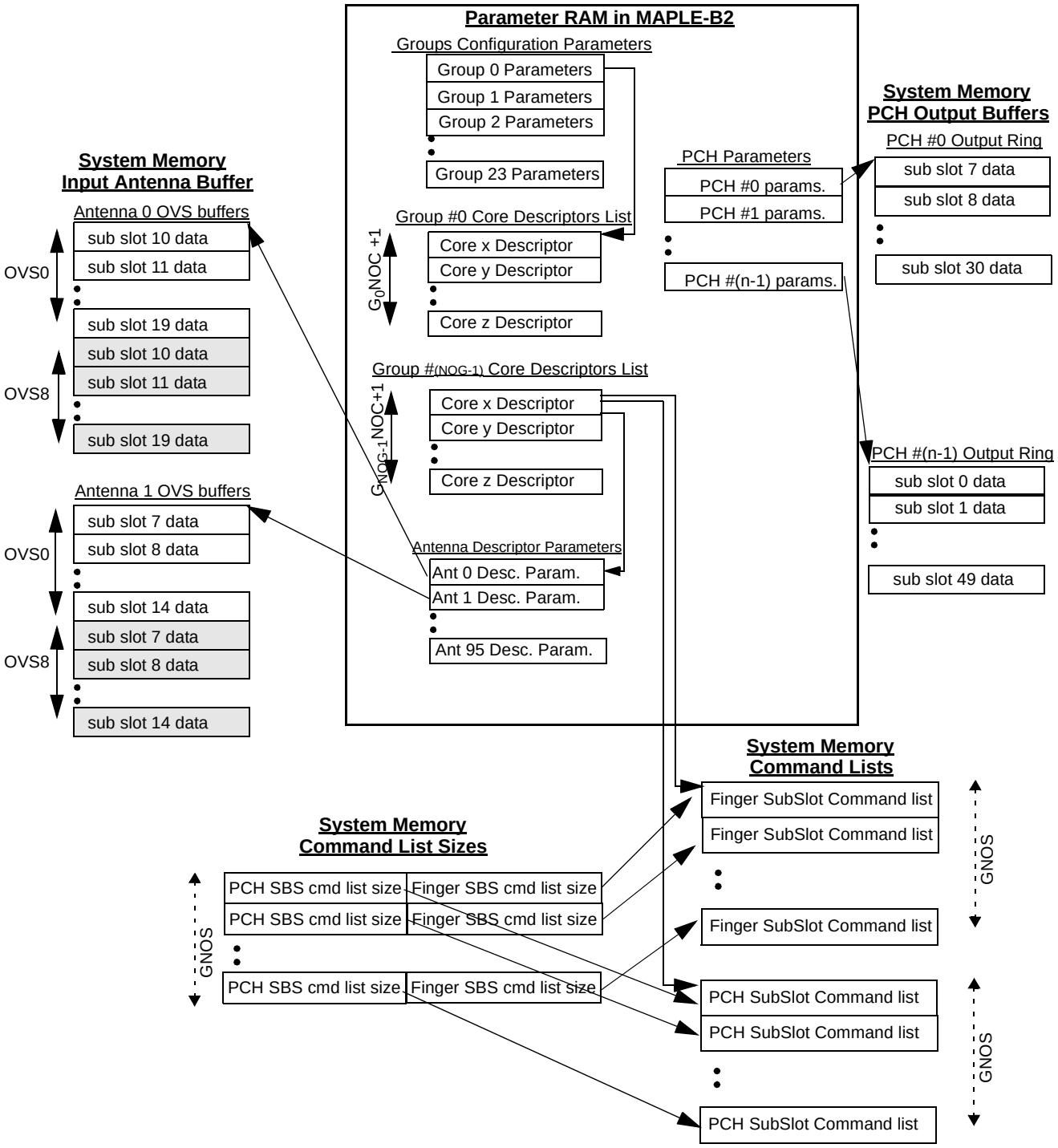


Figure 26-102. Uplink Batch Data Structure Map

26.4.3.6.1.4 CRPE-ULB Data Organization

The CRPE Uplink Batch processing uses the following data organization which supports the following:

- **Groups**: up to 24 groups. Each group is associated with up to 8 cores. The cores in the group are organized in a list inside the MAPLE-B2 parameter memory starting at address *MCUBGxCP1[CDLBA]*. All cores belonging to a group are processed by the CRPE-ULB as a single batch (per Sub-Slot).
- **Core:** each core manages the availability of the input stream (PCH commands, Fingers commands and Antenna data) towards the CRPE-ULB using its own antenna input data structure and commands lists. For details see **Section 26.4.3.6.1.5**, *CRPE-ULB Command Processing Flow*, on page 26-204
- **Antennas**: Antennas are allocated to Groups using the following configurations:
  - Antennas base address are defined in the *MCUBANTxDP[ABA]* parameter fields.
  - Antennas size is defined in the *GANTSZ* field of the *MCUBGxC3P* parameter.
  - Antennas allocated to a Group index start from the Group's first antenna index: *CRUBGFAxCR[GxFANT]*, and to antenna index: *CRUBGFAxCR[GxFANT] + CRUBGNOAxCR[GxNOA] - 1*

Note: An antenna index can be allocated to a single Group only, that is, no overlapping of antennas across Groups.

Configuration example:

- Group #0 contains antennas 0–1 and a single core handling all PCHs of the antennas.
- Group #1 contains antennas 2–9 and two cores. Each Core is responsible for different PCHs allocated to this group.

26.4.3.6.1.5 CRPE-ULB Command Processing Flow

In order for the MAPLE-B2 to start with Sub-Slot processing of a certain Group (Group x), all the Sub-Slot Write Pointers of the Cores related to this Group (*CSWP* fields of the Group's Core Descriptors) must have a different value from the Group Sub-Slot Read Pointer (*MCUBGxC2P[GSRP]*). This indication implies that the PCH/Finger Commands (of all the Cores of the Group) for the next Sub-Slot to be processed are already updated by the Cores in the system memory. Once the above is met, the following is executed by the MAPLE-B2:

1. Read from the Core Command Size Table of each Core (pointed by the *CCLSTBA* field of each Core Descriptor) the size of the PCH command list for the current Sub-Slot. The

Sub-Slot offset within the Core Command Size Table is indicated by the Group x Sub-Slot Command Read Pointer: *MCUBGxC2P[GSCRP]*.

2. Once PCH Command size is available: Fetch the PCH commands (of each Core) from the Core Command List. The Sub-Slot offset within the PCH Command List is indicated by the Group x Sub-Slot Command Read Pointer: *MCUBGxC2P[GSCRP]*.
3. Read from the Core Command Size Table of each Core (pointed by the *CCLSTBA* field of each Core Descriptor) the size of the Finger command list for the current Sub-Slot. The Sub-Slot offset within the Core Command Size Table is indicated by the Group x Sub-Slot Command Read Pointer: *MCUBGxC2P[GSCRP]*.
4. Once Finger Command size is available: Fetch the Finger commands (of each Core) from the Core Command List. The Sub-Slot offset within the Finger Command List is indicated by the Group x Sub-Slot Command Read Pointer: *MCUBGxC2P[GSCRP]*.
5. Once PCH/Finger Commands of all Cores are fetched into the CRPE -ULB, the MAPLE-B2 resets the sizes of the PCH/Finger Commands (of each Core) in the Core Command Size Table indicating that this Sub-Slot Commands are already fetched and can be over-run.
6. The MAPLE-B2 increment by one the *MCUBGxC2P[GSCRP]* field indicating that this Sub-Slot index is under processing.
7. Pending internal CRPE-ULB input buffer availability, the MAPLE-B2 fetches Antenna data relevant for this Sub-Slot processing from the Antenna System buffers as described in **Section 26.4.3.6.1.7**, *Antenna Input Sub Slot Data Structure (All modes)*.

Note: Incrementing *MCUBGxC2P[GSRP] and MCUBGxC2P[GSCRP]* does not guarantee data coherency - i.e the associated data transfers have not necessarily completed. Only incrementing *MCUBGxC4P[GOBSWP]* implies that all input data for the job has been used hence can be overwritten

When the CRPE-ULB Output buffer is ready with a Group's Sub-Slot processed data, the MAPLE-B2 performs the following:

1. Check if the current Sub-Slot Completion drives Finished Channels Buffer update and potential interrupt assertion (for details see **Section 26.4.3.6.1.15**, *Output Interrupt and Finished Channels Buffer Maintenance.*)
2. For each PCH related to the Group, the following is executed:
   - PCH output data is transferred to system memory as described in **Section 26.4.3.6.1.14**, *Output Buffer Data Structure*.
   - If Finished Channels Buffer update check point is reached, checks the PCH needs based on Finished Channel definition and acts accordingly (see **Section 26.4.3.6.1.15.1**, *Finished Channel Definition*).
3. Increment *MCUBGxC4P[GOBSWP]* by one, indicating the last Sub-Slot results are available in the system memory.

26.4.3.6.1.5.1 Core Descriptors and Command List Size Table

The Core Descriptor that describes a core within a Group (see **Section 26.5.4.6.1**, *CRPE-ULB Core Descriptors*, on page 26-475 for details) includes several fields describing the commands data base:

- *CCLSTBA* points to a cyclic table which holds the actual size of the PCH command list in bytes and the actual size of the Finger command list in bytes for each Sub-Slot entry (*GNOS* entries) of a core in a specific Group. The size of each command list must be aligned to 16 bytes (the 3 LSB are ignored by MAPLE-B2).The following figure describe the expected Command List Size Data Structure:

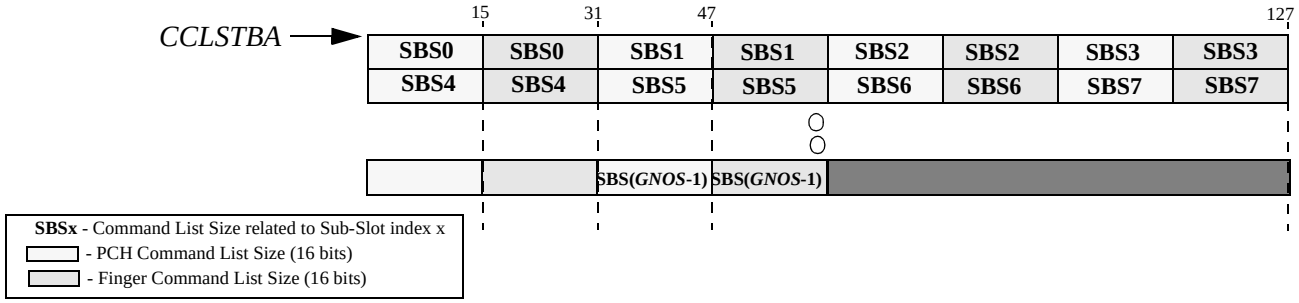


Figure 26-103. Command List Size Data Structure

- *CPBA*/*CFBA* is a base address to the PCH/Finger Command List of the Core associated with Sub-Slot index #0. The PCH/Finger Command Lists of the following Sub-Slot indexes are concatenated to this Command List. The MAPLE-B2 uses this base address to calculate the location of the PCH/Finger Command Lists.
- *CMAXP*/*CMAXF* (in 16 bytes multiplication). The maximum potential size of the PCH/Finger Command List of a Core per Sub-Slot. The VCOP uses this size to calculate the start address of the PCH/Finger Command List associated with each Sub-Slot index. **Figure 26-104** describe an example of PCH Command List Data Structure and Address calculation:

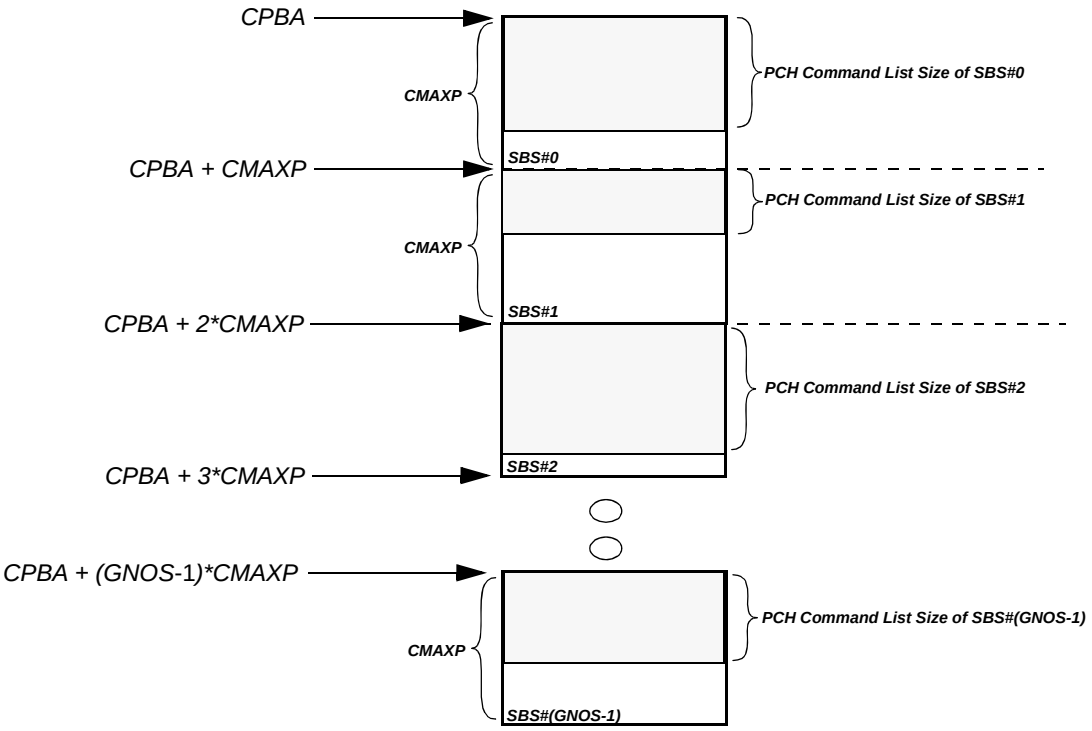


Figure 26-104. PCH Command List Data Structure

26.4.3.6.1.5.2 Finger Commands

The Finger Command data structure defines fingers attributes and allows performing operations on Fingers associated with a PCH. The Finger Command data structure is described in **Section 26.5.4.6.2**, *CRPE-ULB Finger Command Structure*.

The following describe the parameter fields in the Finger Command:

- *FMO* is the Finger Memory Offset of the CRPE-ULB operation result. Valid when Finger Combining mode is disabled (*CRUBMCP[FC_MODE]*=0). For details see **Section 26.4.3.6.1.14.1**, *Finger Combining Bypass (FC_MODE = 0)*.
- *AID* is the Antenna ID number (0 to 95).
- *PCH_ID*. The value of this field should be identical to the PCH_ID field of the PCH Command related to the PCH it is associated with.
- *GAIN_I*, *GAIN_Q* represent the gain complex value to be multiplied with the generated symbol after chip combining.
- *FOFFSET_H* is the offset (in chips) of the finger from the beginning of the Sub-Slot.
- *FOFFSET_L* is the offset of the start of the finger within a chip.
- *CMD* describe which type of Finger command is required. **Table 26-96** describe the command type options:

Table 26-96. Finger Commands Types Options

[CMD] Value	Command Meaning	Required Valid Fields In Command
0	NULL	None.
1	Modify Finger - Gain	CMD, FID, GAIN_I, GAIN_Q
2	Add new Finger	All field
3	Remove finger	CMD,FID, AID, FOFFSET_L
5	modify finger - all	All fields

- *FID* is the unique Finger ID for each existing finger.

The MAPLE-B2 supports two optional Finger command sizes : 16 bytes command size and 8 bytes command size.

16 Bytes Finger Command size

The following figure describe the structure of a 16 bytes Finger command:

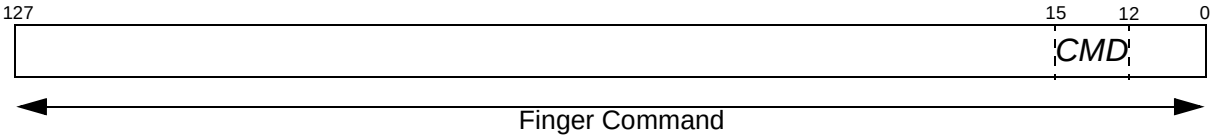


Figure 26-105. 16 bytes Finger command

For the 16 bytes command size all *CMD* values described in **Table 26-96** are valid.

8 Bytes Finger Command size

In case the required Finger command type is “Modify Finger - Gain” (*CMD* = 1 as per **Table 26-96**), it is possible to place two such Finger commands in a 16 bytes data structure as follows:

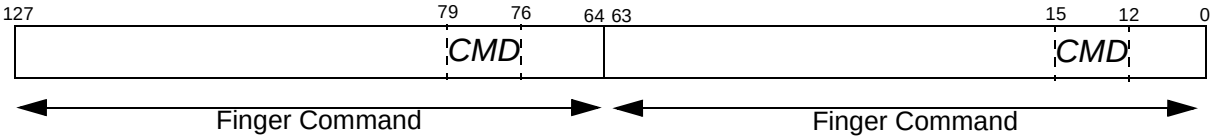


Figure 26-106. 8 bytes Finger command

Note: A 16 bytes command size start address must be aligned to 16 bytes address. Also, if single 8 bytes command is required in a 16 bytes aligned address, the other 8 bytes command field must get a NULL value (*CMD* = 0).

26.4.3.6.1.5.3 PCH Commands

The PCH Command data structure defines PCH attributes and allows performing operations on the PCH. The PCH Command data structure is described in **Section 26.5.4.6.3**, *CRPE-ULB PCH Command Structure*.

The following describe the parameter fields in the PCH Command:

- *NOF* defines the maximum Number Of Fingers for this PCH. It it used only in the case of Finger Combining is disabled (*CRUBMCP[FC_MODE]*=0).
- *ODT* is the Output Data Type for this PCH. It can be either the real portion only, the imaginary portion only, the complex number (both portions) or out data is disabled (compress PCH). For details see **Section 26.4.3.6.1.13**, *Output Buffer Capacity*
- *CS* indicates the Code type for this PCH (long/short Code). for details see **Section 26.4.3.6.1.9**, *Descrambling*
- *GRP_ID* indicates to which Group this PCH is related.There are up to 24 Groups supported.
- *EXP* represent the scale value used in the output buffer during Finger combining. For details see **Section 26.4.3.6.1.12**, *Finger Combining*.
- *FCOR_I*, *FCOR_Q* represent the real and imaginary portions of the frequency correction factor to be multiplied (each Sub-Slot) with the previous frequency correction factor. for details see **Section 26.4.3.6.1.12**, *Finger Combining*.
- *OVSF* indicates the PCH Orthogonal Variable Spreading Factor for this PCH.
- *SCRI* is the initial code value for the Descrambling processing. For details see **Section 26.4.3.6.1.9**, *Descrambling*.
- *PCH_ID* is the PCH ID number this PCH command is targeted for.
- *CMD* describe which type of PCH command is required. **Table 26-97** describe the command type options:

Table 26-97. PCH Commands Types Options

[CMD] Value	Command Meaning	Required Valid Fields In Command
6	add / modify PCH	All Fields
7	Remove PCH	PCH_ID
8	Reset FCOR_I_VAR, FCOR_Q_VAL values	PCH_ID

- *SF* indicates the PCH Spreading Factor.
- *SCBO* indicates the offset (in Sub-Slot units) between the PCH scrambling code and its first Sub-Slot.

26.4.3.6.1.6 Interpolation Modes

The following subsections describe the operation of the various interpolation modes (*CRUBMCP[INT_MODE]*) as defined by the programmed input interpolation level (*MCUBCP[IIL]*)

26.4.3.6.1.6.1 Internal Interpolation Antenna Input Data Structure (*INT_MODE* = 1)

For the case of *CRUBMCP[INT_MODE]* = 1, the interpolation processing is done inside the CRPE-ULB and the expected input data buffers from each antenna are:

1. Antenna Over Sampling #0 input buffer (OVS0).
2. Antenna Over Sampling #8 input buffer (OVS8).

Figure 26-107 illustrates an example of a single antenna input data structure for the case where the number of input sub slots in the ring is 5 (*MCUBGxC3P[GANTSZ]* = 5). The antenna data structure is located in the system in address pointed to by *MCUBANTxDP[ABA]*. The VCOP reads the next 256 chips data from the location pointed to by *MCUBGxC3P[GARP]* for all antennas belonging to the group.

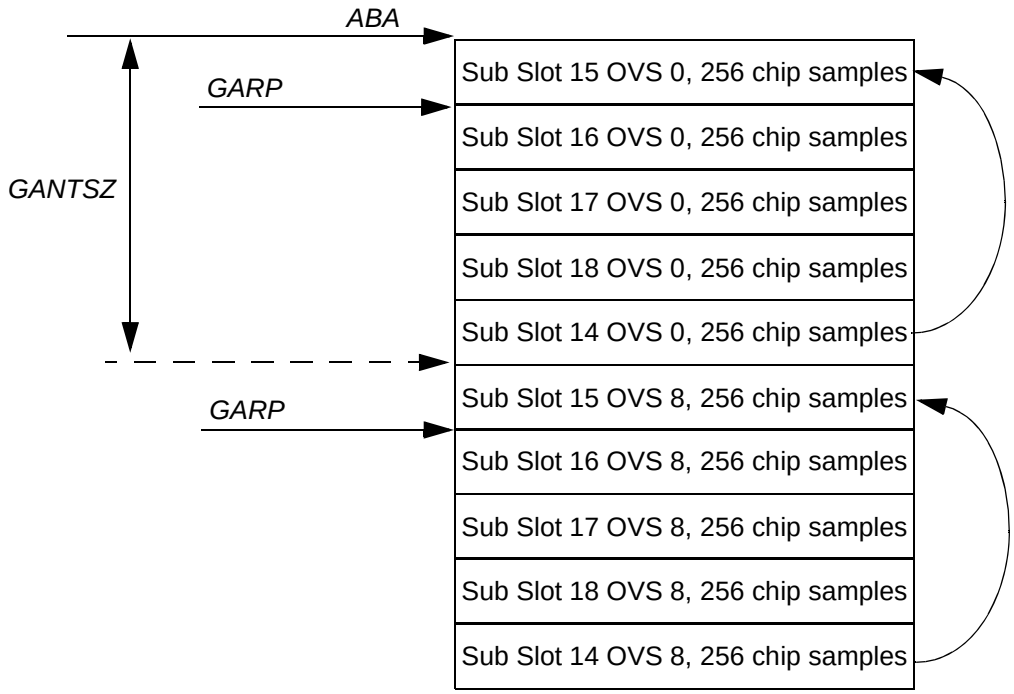


Figure 26-107. Single Antenna input Data Structure Where *GANTSZ* = 5 and *INT_MODE* = 1

26.4.3.6.1.6.2 External Interpolation 2x Antenna Input Data Structure (*INT_MODE*=0, *IIL*=0)

For the case of *CRUBMCP[INT_MODE]* = 0, the antenna input data structure is assumed to be interpolated already. With Input Interpolation Level 2 (*MCUBCP[IIL]*=0) the input antenna data contains two over-samples per chip.

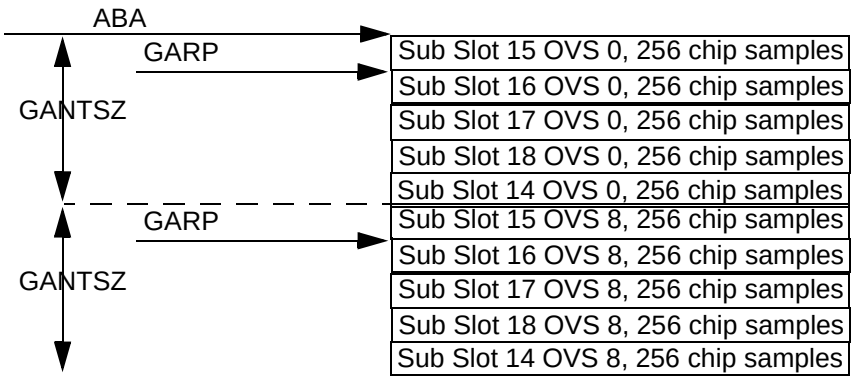


Figure 26-108. Single Antenna Input Data Structure Where *GANTSZ*=5, *INT_MODE* = 0, *IIL*=0

26.4.3.6.1.6.3 External Interpolation 4x Antenna Input Data Structure (*INT_MODE*=0, *IIL*=1)

For the case of *CRUBMCP[INT_MODE]* = 0, the antenna input data structure is assumed to be interpolated already. With Input Interpolation Level 4 (*MCUBCP[IIL]*=1) the input antenna data contains four over-samples per chip.

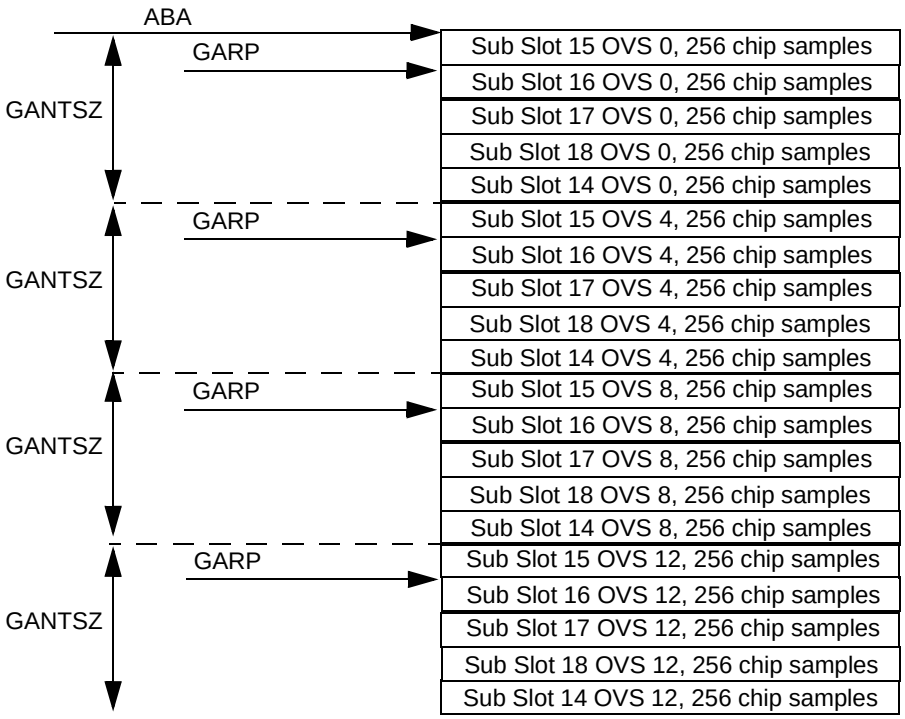


Figure 26-109. Single Antenna Input Data Structure Where *GANTSZ*=5, *INT_MODE* = 0, *IIL*=1

26.4.3.6.1.6.4 External Interpolation 8x Antenna Input Data Structure (*INT_MODE*=0, *IIL*=2)

For the case of *CRUBMCP[INT_MODE]* = 0, the antenna input data structure is assumed to be interpolated already. With Input Interpolation Level 8 (*MCUBCP[IIL]*=2) the input antenna data contains eight over-samples per chip.

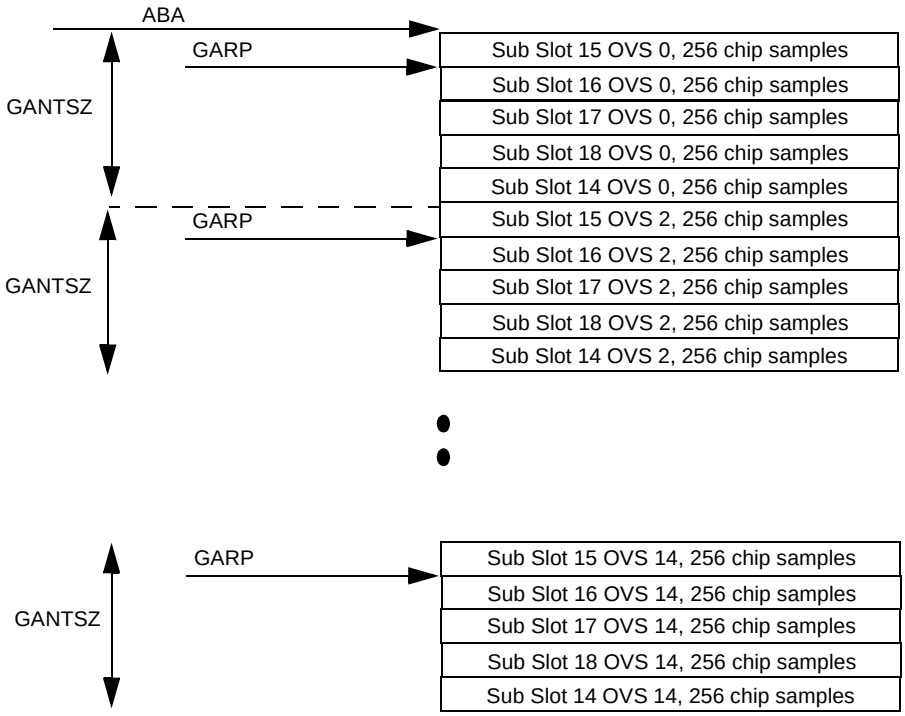


Figure 26-110. Single Antenna Input Data Structure Where *GANTSZ*=5, *INT_MODE* = 0, *IIL*=2

26.4.3.6.1.6.5 External Interpolation 16x Antenna Input Data Structure (*INT_MODE*=0, *IIL*=3)

For the case of *CRUBMCP[INT_MODE]* = 0, the antenna input data structure is assumed to be interpolated already. With Input Interpolation Level 8 (*MCUBCP[IIL]*=3) the input antenna data contains sixteen over-samples per chip.

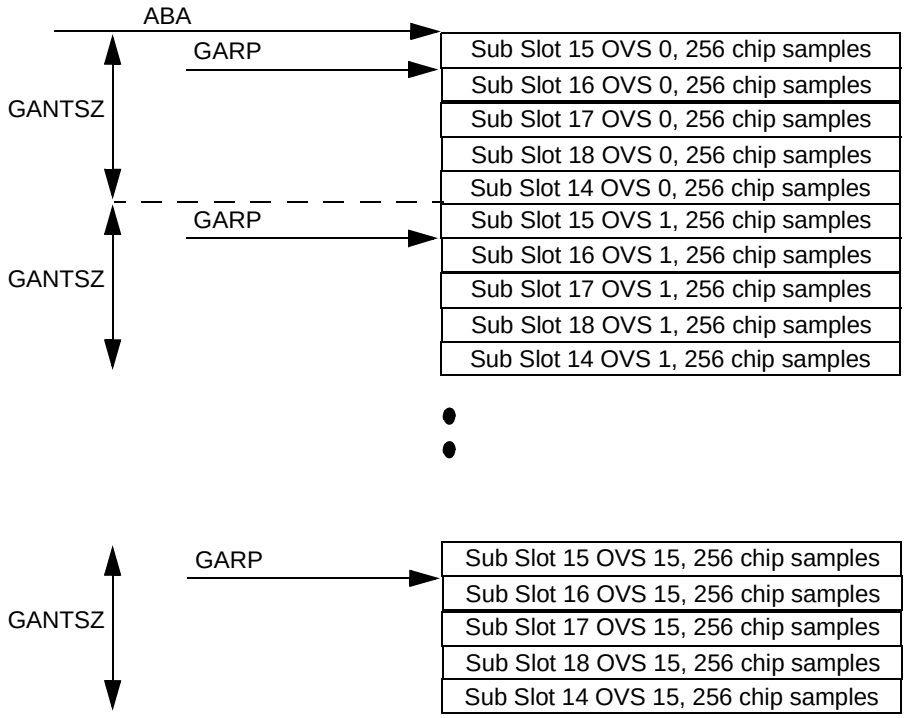


Figure 26-111. Single Antenna Input Data Structure Where *GANTSZ*=5, *INT_MODE* = 0, *IIL*=3

26.4.3.6.1.7 Antenna Input Sub Slot Data Structure (All modes)

Figure 26-112 illustrates one sub slot input data structure

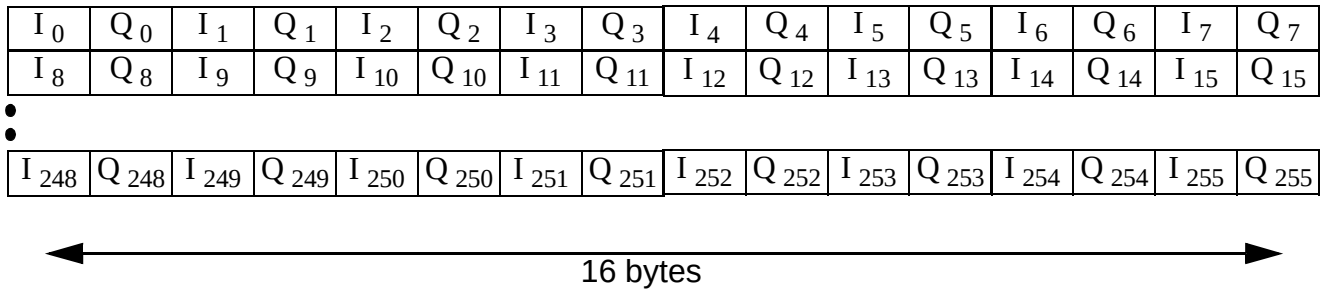


Figure 26-112. Sub Slot Data Structure

Figure 26-113 illustrates another example: Group #1 which is composed of 3 antennas. Antenna 2,3 and 4 which base address is located in *MCUBANT2DP[ABA]*, *MCUBANT3DP[ABA]* and *MCUBANT4DP[ABA]* correspondingly. All 3 antennas share the same group antenna size: *MCUBG1C3P[GANTSZ]* = 10 sub slots and the same read pointer *MCUBG1C3P[GARP]* = 2. Antenna #2 is managed by core #0 and antennas #3 and #4 by core #1. The figure illustrates the pointers values with respect to current memory structure of a single oversampled buffer per antenna. Using the *ABA* and *GANTSZ* value, the MAPLE-B2 internally calculates the relevant pointers for the other oversampled buffers.

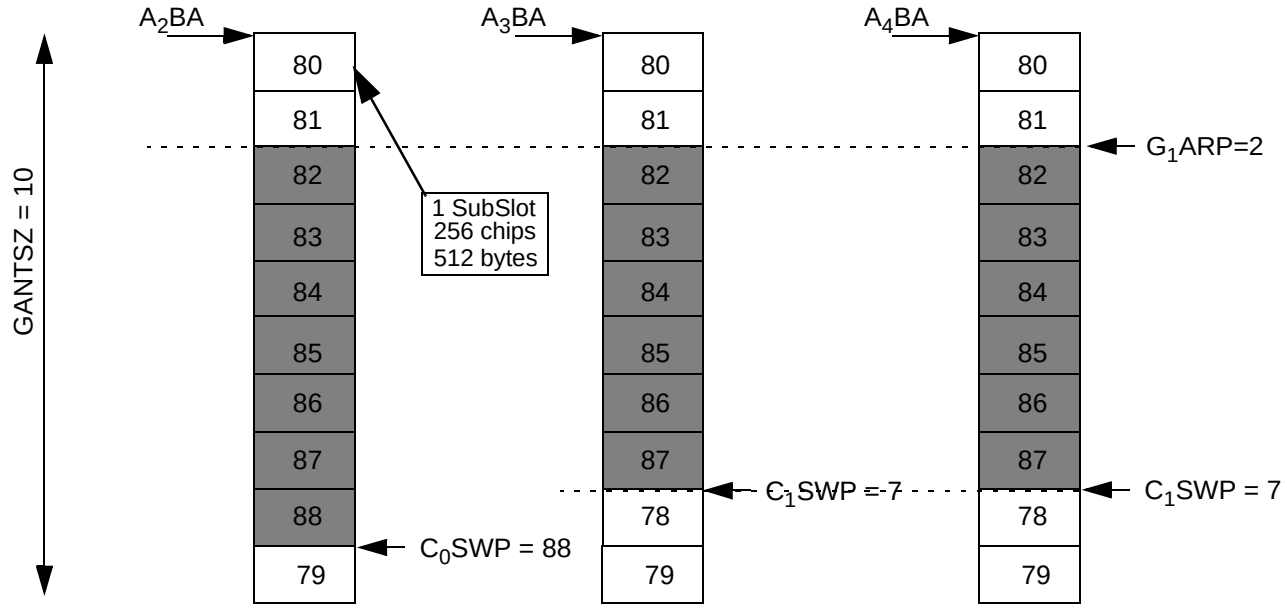


Figure 26-113. Input Buffer Structure of Single Oversampled Buffer for the Group's Antennas

VCOP processes sub slot k for all antennas in a group only if all the group's cores Write Pointers (C_xSWP) >= (k + 4)[1] and only after reading all commands belonging to sub slot k (refer to **Section 26.4.3.6.1.5**).

26.4.3.6.1.8 Internal Interpolation Processing implementation

The internal interpolation operation uses one set of 8 weights to create a single interpolated samples. The 8 nearest over-samples (4 past and 4 future samples) are used to generate the interpolated sample using the 8 relevant weights (*CRUBIW$_x$S$_y$CR[IW$_z$S$_y$]*, x = 0,1, y = 0...7, z = 0...7).

1. In case *CRUBMCP[DLY_SPRD]* = 1, then sub slot k can be processed only if (C_xSWP) >= (k+5)

26.4.3.6.1.8.1 Interpolation x16: *INT_MODE* = 1

Figure 26-114 illustrates how over-Samples are created for Interpolation x 16 for S_{n+1}:

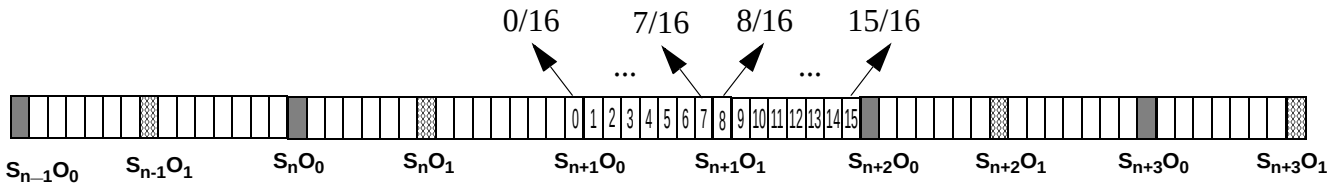


Figure 26-114. 8 Phase, 8 Tap, x16 Interpolation

For (x = 0 ; x = 7; x++):

$$S_{n+1;x/16} = IW0S_x * S_{n-1}O_1 + IW1S_x * S_nO_0 + IW2S_x * S_nO_1 + IW3S_x * S_{n+1}O_0 + IW4S_x * S_{n+1}O_1 + IW5S_x * S_{n+2}O_0 + IW6S_x * S_{n+2}O_1 + IW7S_x * S_{n+3}O_0$$

For (x = 8 ; x = 15; x++):

$$S_{n+1;x/16} = IW0S_x * S_nO_0 + IW1S_x * S_nO_1 + IW2S_x * S_{n+1}O_0 + IW3S_x * S_{n+1}O_1 + IW4S_x * S_{n+2}O_0 + IW5S_x * S_{n+2}O_1 + IW6S_x * S_{n+3}O_0 + IW7S_x * S_{n+3}O_1$$

26.4.3.6.1.8.2 Interpolation x8: *INT_MODE* = 1

Figure 26-115 illustrates how Over-Samples are created for Interpolation x 8 for S_{n+1}:

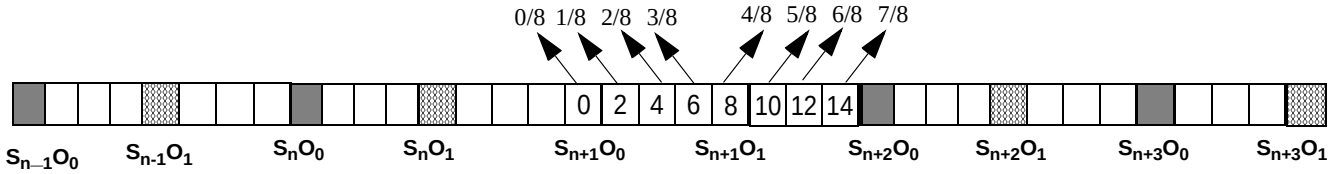


Figure 26-115. 8 Phase,8 Tap, x8 Interpolation

For (x = 0,2,4,6):

$$S_{n+1;x/8} = IW0S_x * S_{n-1}O_1 + IW1S_x * S_nO_0 + IW2S_x * S_nO_1 + IW3S_x * S_{n+1}O_0 + IW4S_x * S_{n+1}O_1 + IW5S_x * S_{n+2}O_0 + IW6S_x * S_{n+2}O_1 + IW7S_x * S_{n+3}O_0$$

For (x = 8,10,12,14):

$$S_{n+1;x/8} = IW0S_x * S_nO_0 + IW1S_x * S_nO_1 + IW2S_x * S_{n+1}O_0 + IW3S_x * S_{n+1}O_1 + IW4S_x * S_{n+2}O_0 + IW5S_x * S_{n+2}O_1 + IW6S_x * S_{n+3}O_0 + IW7S_x * S_{n+3}O_1$$

26.4.3.6.1.8.3 Interpolation x4: *INT_MODE* = 1

Figure 26-116 illustrates how Over-Samples are created for Interpolation x 4 for S_{n+1}:

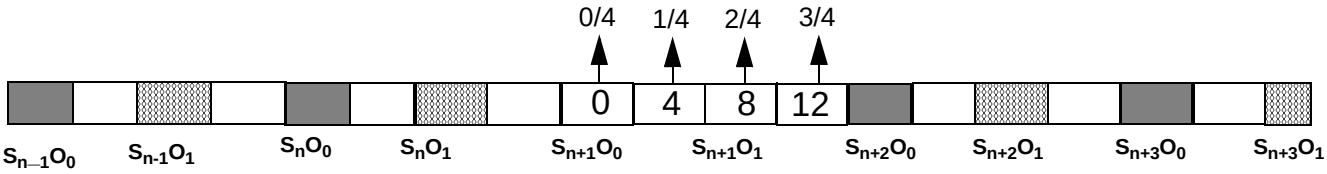


Figure 26-116. 8 Phase, 8 Tap, x4 Interpolation

For (x = 0,4):

$$S_{n+1;x/4} = IW0S_x * S_{n-1}O_1 + IW1S_x * S_nO_0 + IW2S_x * S_nO_1 + IW3S_x * S_{n+1}O_0 + IW4S_x * S_{n+1}O_1 + IW5S_x * S_{n+2}O_0 + IW6S_x * S_{n+2}O_1 + IW7S_x * S_{n+3}O_0$$

For (x = 8,12):

$$S_{n+1;x/4} = IW0S_x * S_nO_0 + IW1S_x * S_nO_1 + IW2S_x * S_{n+1}O_0 + IW3S_x * S_{n+1}O_1 + IW4S_x * S_{n+2}O_0 + IW5S_x * S_{n+2}O_1 + IW6S_x * S_{n+3}O_0 + IW7S_x * S_{n+3}O_1$$

26.4.3.6.1.8.4 Interpolation x2: *INT_MODE* = 1

Figure 26-117 illustrates how Over-Samples are created for Interpolation x 2 for S_{n+1}:

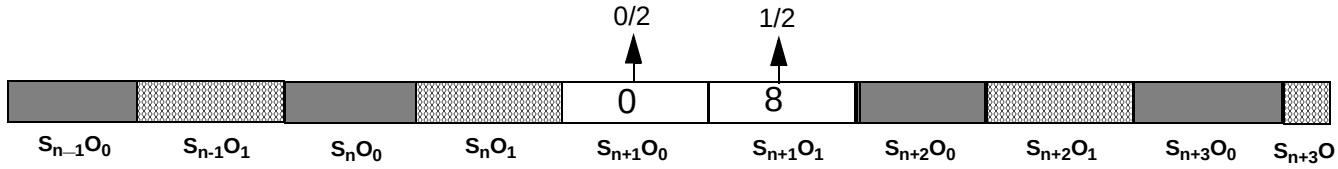


Figure 26-117. 8 Phase, 8 Tap, x2 Interpolation

$$S_{n+1;0/2} = IW0S_0 * S_{n-1}O_1 + IW1S_0 * S_nO_0 + IW2S_0 * S_nO_1 + IW3S_0 * S_{n+1}O_0 + IW4S_0 * S_{n+1}O_1 + IW5S_0 * S_{n+2}O_0 + IW6S_0 * S_{n+2}O_1 + IW7S_0 * S_{n+3}O_0$$

$$S_{n+1;1/2} = IW0S_0 * S_nO_0 + IW1S_0 * S_nO_1 + IW2S_0 * S_{n+1}O_0 + IW3S_0 * S_{n+1}O_1 + IW4S_0 * S_{n+2}O_0 + IW5S_0 * S_{n+2}O_1 + IW6S_0 * S_{n+3}O_0 + IW7S_0 * S_{n+3}O_1$$

26.4.3.6.1.8.5 Internal Interpolation Control

When working in Internal Interpolation mode (*CRUBMCP[INT_MODE]*=1), the internal interpolation processing is executed per finger according to the *FOFFSET_L* field of the Finger command (see **Section 26.4.3.6.1.5.2**, *Finger Commands*). This parameter describe the Finger offset between two chips and hence dictate the required internal interpolation executed for this chip. The range of values for the *FOFFSET_L* parameter is 0 to 15 and the internal interpolation executed for the finger is described in **Table 26-98**:

Table 26-98. Internal Interpolation Type with Respect to *FOFFSET_L*

FOFFSET_L	Internal Interpolation execution
1,3,5,7,9,11,13,15	x16
2,6,10,14	x8
4,8,12	x4
0,8	x2

Note: When working in Double Antenna mode (*CRUBMCP[DOUBLE_ANT_EN]*), x16 internal interpolation is not supported hence allowed values for the *FOFFSET_L* are: 0, 2, 4, 6, 8, 10, 12 and 14.

26.4.3.6.1.8.6 Interpolation Saturation During Internal Interpolation

Each interpolated chip is the results of adding 8 multiplications results, each multiplication between an input chip sample (8 bits) and its corresponding weight (8 bits). Because each multiplication result can potentially be as large a 16 bits, their sum may be as large as 19 bits. This intermediate result is then divided (shifted right) by the value of *CRUBMCP[ISHA]*, and if the remaining result is greater than (+127) or smaller than (-128) then it is considered as a saturated result, else the result is truncated to 8 bits representation. If the result is saturated the *CRUBISCSR[IS_COUNT]* is increment by one and the 8 bits result get the value of (+127) or (-128) depend on saturation direction. **Figure 26-118** describe the internal interpolation calculations implementation:

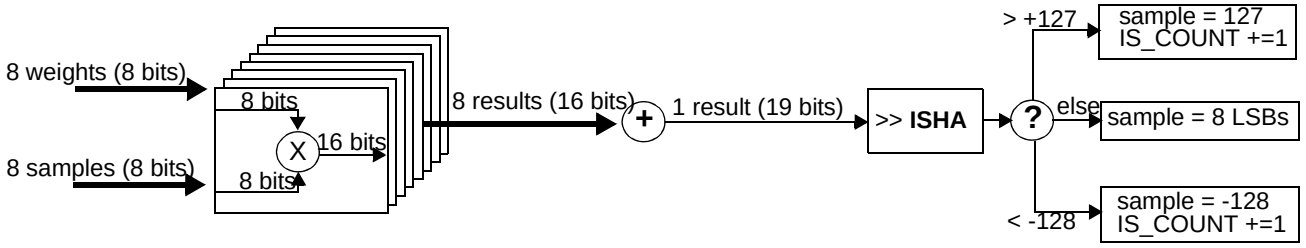


Figure 26-118. Internal Interpolation Calculation Flow

26.4.3.6.1.9 Descrambling

The CRPE-ULB performs the descrambling operation in accordance with the 3GPP standard (section 4.3.2). The complex-value sequence of interpolated chips is multiplied by a complex-value scrambling code that is internally generated. The scrambling code may either be a long code or a short code, depending on the value of the *CS* field of the PCH command associated with the currently processed finger.

Figure 26-119 illustrates the descrambling process. A scrambled chip is multiplied by the conjugate of the i^{th} element of the scrambling code $C_n[i]$, and a descrambled chip is generated.

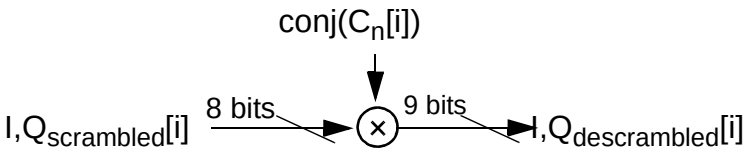


Figure 26-119. Chip Descrambling Illustration

The scrambling operation can potentially increase the I,Q data dynamic range from 8 bits to 10 bits. However, as we treat the temporary result: -(-256) as +255, rather than +256, 9 bits suffice.

26.4.3.6.1.9.1 Scrambling Data Offset

Prior to the descrambling operation, the *CSBO* value of the PCH command - which is PCH specific, and the *MCUBGxC2P[GxSRP]* value - which is common to all currently processed PCHs, needs to be properly initialized.

Figure 26-120 illustrates the relationship between *CSBO* and *GxSRP*:

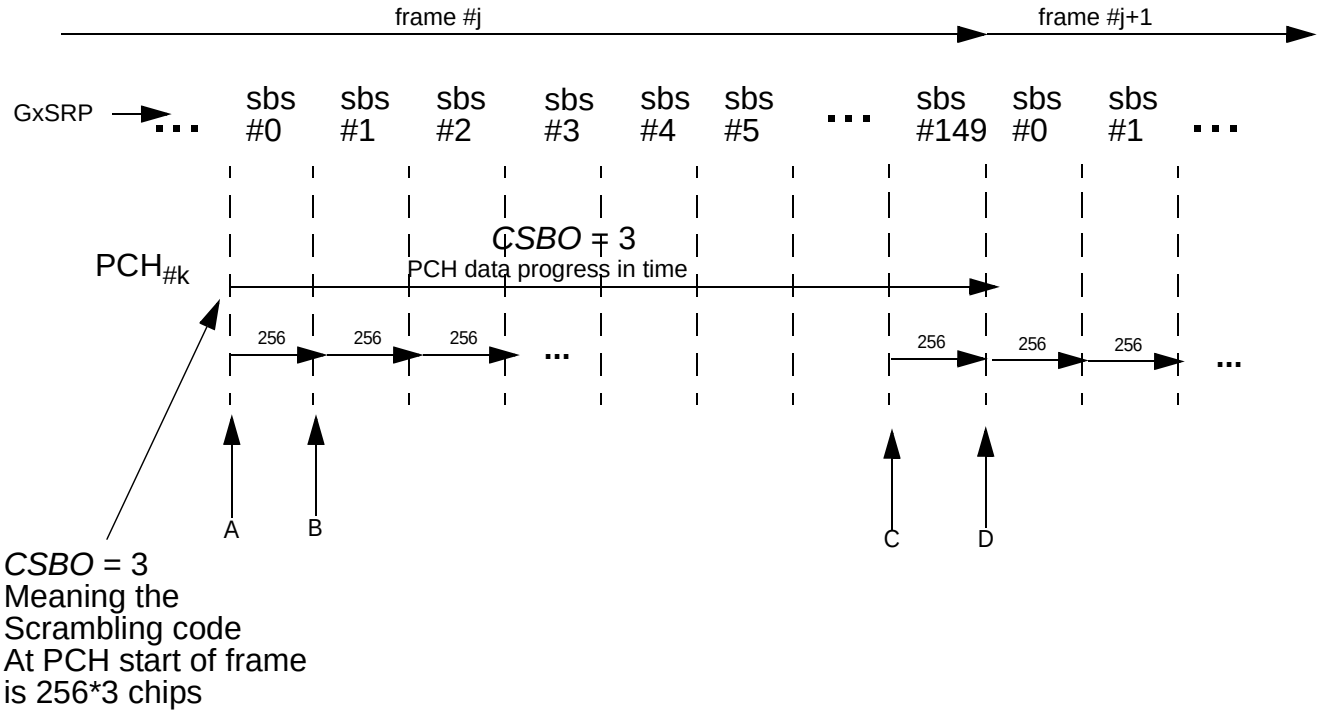


Figure 26-120. Scrambling Code Init Per PCH

The *CSBO* value, indicates, in units of sub-slots (256 chips), the time period between the PCH scrambling code initialization, when it started generating one complex value each chip-period, and the time in which the PCH transmission of its first chip. In the example, *CSBO* = 3 for PCH #k, since 3 sub-slots pass between the scrambling code initialization, and the beginning of transmission.

The *GxSRP* value, indicates the currently processed sub-slot by CRPE-ULB. It is common to all PCHs.

As a result, for each 256 chips job, the scrambling initial value for each PCH, is calculated as follows:

1. The scrambling code generator is initialized using the *SCRI* value of the PCH command (PCH specific).
2. The scrambling code generator is 'advanced' in time for an amount of chips equals to: (*GxSRP* + *CSBO*)%150[1] * 256;

In the example above (**Figure 26-120**), the scrambling code initial chip starting points are:

— Point A: (0+3)%150 *256 = 768 chips
— Point B: (1+3)%150 * 256 = 1,024 chips
— Point C: (149+3)%150 * 256 = 2 * 256 = 512 chips
— Point D: (0+3)%150 * 256 = 768 chips

26.4.3.6.1.9.2 Descrambling Using Long Codes

If the value of *CS* for a PCH is '0' (long code indication) - the descrambling code that is generated internally is a long code. This section provides the definition of descrambling code $C_{long,n}[i]$ for n=0,..., $2^{\wedge 24} - 1$ and i=0,..., 38399. The descrambling code is a complex sequence consisting of any of the following complex numbers: 1+j, 1-j, -1+j, -1-j. Hence every complex element of the scrambling pattern can be generally represented as follows:

$C_{long,n}[i] = I_n[i] + (j*Q_n[i])$

where $I_n[i]$ = 1 or –1, and $Q_n[i]$= 1 or –1.

Let X(i) denote the i^{th} symbol of sequence X.

Initial condition for sequence X is:

$X(0)=SCRI[0],\ X(1)=SCRI[1],\ldots\ X(23)=SCRI[23]$

where *SCRI* is initialized in the PCH command.

Recursive definition of subsequent X symbols is:

$X_n(i+24) = X_n(i+3)\ \text{xor}\ X_n(i).$

Let Y(i) denote the i^{th} symbol of sequence Y.

Initial condition for sequence Y is:

$Y(0)=Y(1)=Y(2)=\ldots=Y(22)=Y(23)=1.$

Recursive definition of subsequent Y symbols is:

$Y(i+24) = Y(i+3)\ \text{xor}\ Y(i+2)\ \text{xor}\ Y(i+1)\ \text{xor}\ Y(i).$

Let $z_n(i)$ denote the i^{th} symbol of sequence z_n.

The n_{th} sequence z_n is defined as:

$z_n(i) = X_n(i)\ \text{xor}\ Y(i).$

1. 150 is the number of sub-slots in a frame - the scrambling code is initialized every frame.

Let $Z_n(i)$ denote the i^{th} symbol of sequence Z_n.

The n_{th} sequence Z_n is defined as:

$$Z_n(i) = \begin{cases} +1 \text{ if } x_n(i) = 0 \\ -1 \text{ if } x_n(i) = 1 \end{cases}$$

Finally, the n_{th} complex descrambling code sequence $C_{long,n}$ is defined as:

$$C_{long,n}[i] = I_n[i] + j*Q_n[i]$$

$$I_n[i] = C_{long,1,n}[i] = Z_n(i)$$

$$Q_n[i] = C_{long,1,n}[i] * C_{long,2,n}[2*floor(i/2)] * (-1)^i$$
$$= Z_n(i) * Z_n(2*floor(i/2)+16777232) * (-1)^i$$

26.4.3.6.1.9.3 Descrambling Using Short Codes

If the value of *CS* for a PCH is '1' (short code indication) - the descrambling code that is generated internally is a short code. This section provides the definition of descrambling code $C_{short,n}[i]$ for n=0,..., $2^{\wedge}24 - 1$ and i=0,..., 255. The code is repeated cyclically by setting $C_{short,n}[255] = C_{short,n}[0]$. The descrambling code is a complex sequence consisting of any of the following complex numbers: 1+j, 1-j, -1+j, -1-j. Hence every complex element of the descrambling pattern can be generally represented as follows:

$$C_{short,n}[i] = I_n[i] - j*Q_n[i]$$

where $I_n[i]$ = 1 or -1, and $Q_n[i]$= 1 or -1.

Let a(i) denote the i^{th} symbol of sequence a.

Initial condition for sequence a is:

$$a(0)=2*SCRI[0]+1,\ a(1)=2*SCRI[1],\ldots\ a(7)=2*SCRI[7]$$

where *SCRI* is initialized in the PCH command.

Recursive definition of subsequent a symbols is:

$$a(i) = \{3*a(i-3) + a(i-5) + 3*a(i-6) + 2*a(i-7) + 3*a(i-8)\} \% 4$$

Let b(i) denote the i^{th} symbol of sequence b.

Initial condition for sequence a is:

$$b(0)=2*SCRI[8],\ \ldots\ b(7)=2*SCRI[15]$$

Recursive definition of subsequent b symbols is:

$$b(i) = \{ b(i-1) + b(i-3) + b(i-7) + b(i-8) \} \% 2$$

Let d(i) denote the i^{th} symbol of sequence d.

Initial condition for sequence a is:

$$d(0)=2*[SCRI_{16}]\ ,\ \ldots\ d(7)=2*[SCRI_{23}]$$

Recursive definition of subsequent b symbols is:

$$d(i) = \{ d(i-1) + d(i-3) + d(i-4) + d(i-8) \} \% 2$$

Let $z_n(i)$ denote the i^{th} symbol of sequence z_n.

The n_{th} sequence z_n is defined as:

$$z_n(i) = \{ a(i) + 2b(i) + 2d(i) \} \% 4$$

For all sequences, i = 0...254 , where $z_n(255) = z_n(0)$, which extends the sequence to 256 chips.

Let $Z_n(i)$ denote the i^{th} symbol of sequence Z_n.

The n_{th} sequences $C_{short,1,n}(i)$ and $C_{short,2,n}(i)$ are defined as listed in **Table 26-99**.

Table 26-99. Short Codes Sequences Definitions

$Z_n(i)$	$C_{short,1,n}(i)$	$C_{short,2,n}(i)$
0	+1	+1
1	-1	+1
2	-1	-1
3	+1	-1

Finally, the n_{th} complex descrambling code sequence $C_{short,n}$ is defined as:

$$C_{short,n}[i] = I_n[i] + (j \times Q_n[i]).$$

$$I_n[i] = C_{short,1,n}[i]$$

$$Q_n[i] = C_{short,1,n}[i] * C_{short,2,n}[2*floor(i/2)] * (-1)^i$$

26.4.3.6.1.10 Despreading

The CRPE-ULB performs despreading by correlating the finger chips with their matching elements from the channelization code. The channelization code $C_{ch,SF,OVSF}$ pattern, as it is defined in the 3GPP standard, is generated internally in accordance with section 4.3.1.1 of the standard. The parameters *SF* and *OVSF* from the PCH Command parameters, govern which channelization code is generated.

26.4.3.6.1.11 Combining

The descrambled and despreaded chips of a symbol, are accumulated to form a soft-symbol. The sum of up to 256 chips (9 bits each) generates an intermediate 17 bits result.

26.4.3.6.1.12 Finger Combining

The CRPE-ULB support Finger Combining for symbols of all fingers associated with a PCH. Finger Combining output mode, internal precision, and whether it is used are all determined by the *CRUBMCP[FC_MODE]*, as summarized in **Table 26-100**:

Table 26-100. CRPE-ULB Fingers Combining Modes

FC_MODE value	Description	Supported Output data types: *PCH[ODT]*	Data Output Format with respect to *PCH[ODT]*	Saturation
0	- FC bypass. - Output all fingers as-is per PCH	[2] - Complex I,Q	Each symbol: I (16 bits) + Q (16 bits) , fixed point, unscaled	If *DDS*[1] = '0' it may occur
1	FC enabled with Fixed scale per PCH	[0] - Real only [1] - Imaginary only	[0] - Each symbol: 16 bit fixed point (real portion), with fixed scale [EXP] per PCH. [1] - Each symbol: 16bit fixed point (imaginary portion), with fixed scale [EXP] per PCH.	May occur (depending on [EXP]
2	Reserved			
3	FC enabled. Dynamic scale Floating point format	[0] - Real only [1] - Imaginary only	[0] - Each symbol : 16bit floating point + 6bit exponent , sign extended to16bit (real portion) [1] - each symbol : 16bit floating point + 6bit exponent , sign extended to 16bit (imaginary portion)	Does not occur.

1. DDS is defined in the CRUBMCP parameter.

The Finger Combining operation functionality is illustrated in **Figure 26-121**:

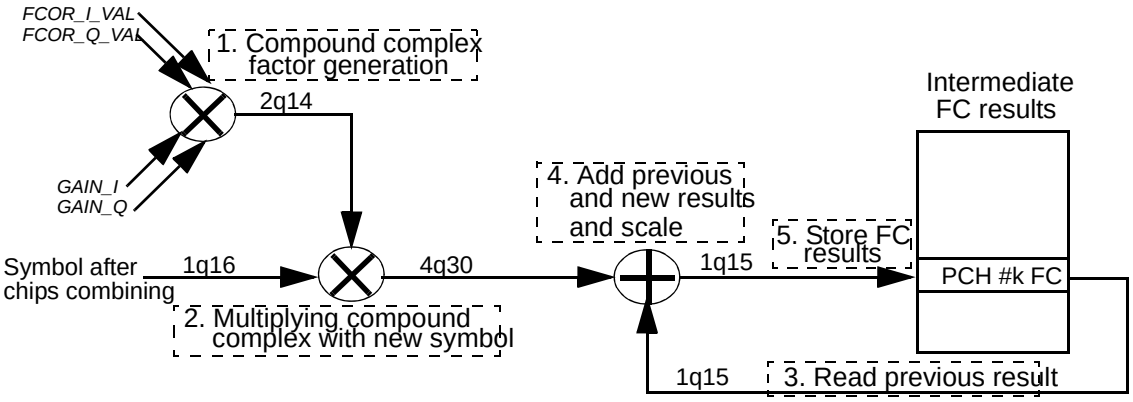


Figure 26-121. Finger Combining Operation

The following list describes the details of the stages shown in **Figure 26-121**:

1. Compound Complex Factor (CCF) is generated by a complex multiplication of the internal complex frequency correction I and Q factors (*FCOR_I_VAL* and *FCOR_Q_VAL* parameters generated by multiplying previous *FCOR_I_VAL* and *FCOR_Q_VAL* with the *FCOR_I* and *FCOR_Q* once per Sub-Slot) with the fingers specific gain factors (*GAIN_I* and *GAIN_Q* parameters of the Finger Command. For details, see **Section 26.4.3.6.1.12.2**, *Frequency Correction Factor*, **Section 26.5.4.6.2**, *CRPE-ULB Finger Command Structure*, and **Section 26.5.4.6.3**, *CRPE-ULB PCH Command Structure*.
2. The CCF is multiplied (complex multiplication) by the symbol generated from the Chips combine unit (see **Section 26.4.3.6.1.12**, *Finger Combining*).
3. The previous intermediate Finger Combining result is read from the output buffer.
4. The previous result and the new result are combined and scaled.

5. The combined result is written to the output buffer.

26.4.3.6.1.12.1 FC Bypass Mode

When *FC_MODE* = 0, the CRPE-ULB does not perform Finger Combining to the symbols associated with any of the PCH. Instead, the PCH related fingers are output by MAPLE-B2 as described in **Section 26.4.3.6.1.14.1**, *Finger Combining Bypass (FC_MODE = 0)*

26.4.3.6.1.12.2 Frequency Correction Factor

The CRPE-ULB supports frequency correction for the purpose of Finger Combining calculation, as described in **Section 26.4.3.6.1.12**, *Finger Combining*. The *FCOR_I* and *FCOR_Q* parameter of the PCH Command represent the expression: '*FCOR_I* + j**FCOR_Q*' or e^{j*b}:

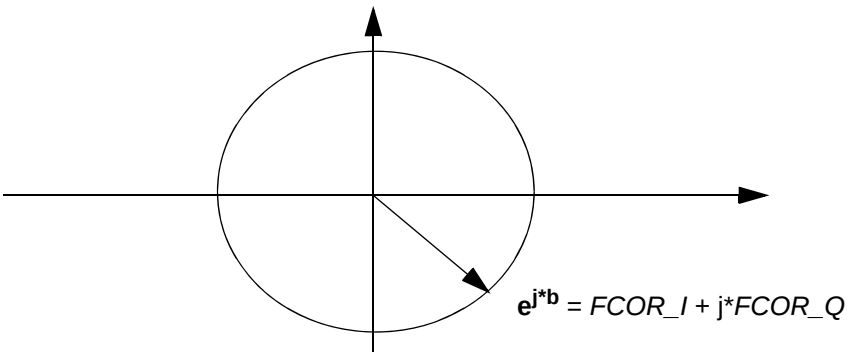


Figure 26-122. Frequency Correction—Representation of Phase

These parameters express the angle by which the phase of the internal *FCOR_I_VAL* and *FCOR_Q_VAL* is rotated (multiplied) once per sub-slot . After adding / modifying a PCH, the value of the *FCOR_I_VAL* and *FCOR_Q_VAL* is set to 0x4000 + 0x0*j , which is equivalent to "1+0*j" in 2q14 format. The *FCOR_I_VAL* and *FCOR_Q_VAL* values may be reset by applying *CMD* opcode of '8' to the PCH command.

In **Figure 26-123**, *FCOR_I* and *FCOR_Q* are set so they represent $e^{j*(pi/8)}$, so each sub-slot, the *FCOR_I_VAL* and *FCOR_Q_VAL* for this PCH is rotated by pi/8. The initial phase for *FCOR_I_VAL* and *FCOR_Q_VAL* (base) is '0x4000+0*j'

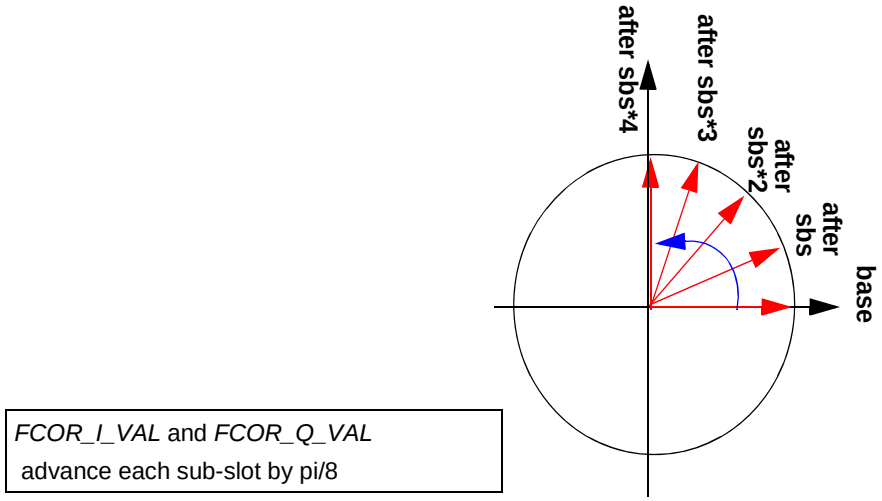


Figure 26-123. Frequency Correction Example

Note: The value '*FCOR_I* + j**FCOR_Q*', which is represented in fixed point format of 2q14, must be bounded by the unit circle, that is, ($FCOR\_I^{\wedge 2} + FCOR\_Q^{\wedge 2}$) <= 1 for correct operation.

26.4.3.6.1.13 Output Buffer Capacity

The CRPE-ULB output buffer supports for up to 4,096 I or Q (real) symbols, in *OOB* = 0 mode, or 8,192 in *OOB* = 1 mode, per group[1]. In general, when possible, it is preferable to work with *OOB* = 0 as it means internal double output buffer implementation hence better performance. **Table 26-101** summarizes the 'cost', in units of symbols, per user, depending on PCH *SF*, *FC_MODE* and *ODT*:

Table 26-101. Number of Real Symbols Per PCH

PCH [SF] value	[FC_MODE] = 0 PCH [ODT] = 2 (Cost in unit of symbols per sub-slot)	[FC_MODE] = 1/3 PCH [ODT] = 0/1 (Cost in unit of symbols per sub-slot)
256	2 * [NOF+1]	1
128	4 * [NOF+1]	2
64	8 * [NOF+1]	4
32	16 * [NOF+1]	8
16	32 * [NOF+1]	16
8	64 * [NOF+1]	32
4	128 * [NOF+1]	64
2	256 * [NOF+1]	128
Note: When FC_MODE = 0, PCH[ODT] must equal 2.		

1. See **Section 26.4.3.6.1.4**, *CRPE-ULB Data Organization* for the definition of a group

However, due to internal implementation, the maximal achieved utilization may be less than 100%. This is due to the required alignment per PCH, according to its *SF*, as described in the table below: (for the *FC_MODE* = 0 case, the effective size, which is *NOF* * (number of symbols) determines the alignment).

Table 26-102. Alignment of PCH According to SF

PCH [SF] Value	Size [units of symbols]	Required Alignment [units of symbols]
256	1	May start at alignments: 0,1,2,3,4,5,6,7
128	2	May start at alignments: 0,2,4,6
64	4	May start at alignments: 0,2,4
32 / 16 / 8 / 4 / 2	8 / 16 / 32 / 64 / 128	May start at alignment 0
Note: A memory line holds 8 symbols, so for all of these cases, they take multiples of one line.		

Using the values in **Table 26-101** and **Table 26-102**, it is possible to calculate the maximum number of PCH supported per required scenario, using the following pseudo code:

```
Alignment = 0;
For (i=0; i < LAST_PCH_NUM; i++)
{
     pch_min_size = func(SF,NOF); // calculate number of symbols according to Table 26-101
     pch_effective_size = pch_min_size + func(Alligment,pch_min_size); according to Table
     26-102
     Alignment = (Alignment + pch_effective_size) % 8 ; // up to 8 symbols per line; update
     alignment

}
```

The effective PCH size, therefore, depends on whether the PCH previous to it (previous PCH_ID) ended in an allowed alignment. If not - an unused space is between the end of the previous PCH and the beginning of the current one.

The following bulleted items provide several use cases as examples:

- Use Case #1. Two sectors AMR without FC (*FC_MODE*=0)
  - 268 PCH with *SF* = 256 and *ODT* = 0
  - Assuming 8 fingers per PCH
  - 268 x 1 symbols per PCH × 8 fingers= 2,144 symbols

  Conclusion: Use case #1 can be supported with *OOB* = 0 (<4,096 symbols)
- Use Case #2. Three sectors, high bit rate AMR, with FC (*FC_MODE*=1/3)
  - 384 PCH with *SF* = 128 and *ODT* = 0
  - 384 × 2 symbols per PCH = 768 symbols

  Conclusion: Use case #2 can be supported with *OOB* = 0 (<4,096 symbols)

- Use Case #3. Two sectors, High Speed, with FC (*FC_MODE*=1/3)
  - 8 PCH × SF8 with VoiP, per sector.
  - 16 PCH × SF4 with High speed data PCH, per sector.
  - 4 PCH × SF256 with Control, per sector
  - Total per sector: 8 × 32 symbols + 16 × 64 symbols + 4 × 1 symbol = 1,284 symbols.
  - Total per device = 1,284 × 2 = 2,568 symbols

  Conclusion: Use case #3 can be supported with *OOB* = 0 (<4,096 symbols)
- Use Case #4. Two sectors, High Speed, without FC (*FC_MODE* =0)
  - 4 PCH × SF2 × 3 fingers
  - 4 PCH × SF4 × 4 fingers
  - Total: 4 × 256 symbols x 3 fingers + 4 × 128 symbols × 4 fingers = 5,120 symbols

  Conclusion: Use case #4 can be supported with *OOB* = 1 only (<8,192 symbols).

26.4.3.6.1.14 Output Buffer Data Structure

Each physical channel has its own output buffer in system memory. (see *MCUBPCHx* parameters in **Section 26.5.3.5.2.1** to **Section 26.5.3.5.2.4**). The output buffer is a cyclical buffer of Sub-Slot data entries. The structure of Sub-Slot data depends on the value of *CRUBMCP[FC_MODE]*. The following subsections describe the Sub-Slot data entry structure depending on *FC_MODE*.

26.4.3.6.1.14.1 Finger Combining Bypass (*FC_MODE* = 0)

In this mode, the MAPLE-B2 does not perform FC to the symbols generated by each finger. As a result the output data buffer of each PCH contains all the fingers related to this PCH. The output buffer structure in this case is arranged as follows: each Finger Command (see **Section 26.5.4.6.2**) includes a Finger Memory Offset (*FMO*) indicating the Finger offset index in the output memory. All of the symbols of finger i are followed by the symbols of finger i+1. The order of the symbols/fingers can also be described as follows:

- The Finger with *FMO*=0 containing symbols 0 to n–1
- The Finger with *FMO*=1 containing symbols 0 to n–1
- ...
- The Finger with *FMO=NOF* containing symbols 0 to n–1.

  where:

  FMO - Finger Memory Offset index (see Finger Command in **Section 26.5.4.6.2**.)

  NOF - Number of Fingers for each PCH see PCH Command in **Section 26.5.4.6.3**.)

  n - Number of symbols in Sub-Slot.

The number of symbols ‘n’ depends on the Spreading Factor (*SF)* of the PCH (see PCH Command in **Section 26.5.4.6.3**), as described in **Table 26-103**:

Table 26-103. Number of Symbols Per Finger

PCH [SF] Value	Value of 'n'
256	1
128	2
64	4
32	8
16	16
8	32
4	64
2	128

Figure 26-124 shows an example of the output data structure of a Sub-Slot with the following properties:

- *SF* = 6 (Spreading factor is: 128)
- *NOF* = 5

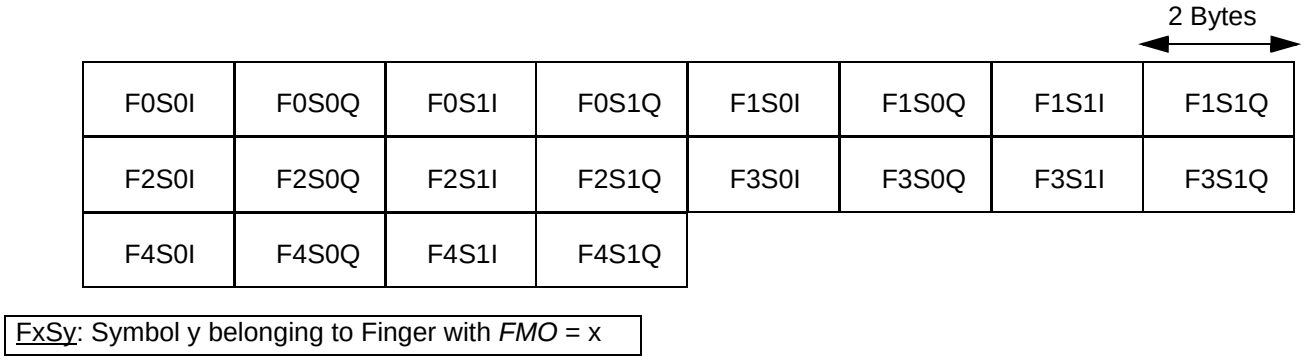


Figure 26-124. Sub-Slot Output Data Structure for *FC_MODE* = 0 Example

Note that *FMO* determines the exact offset in which the finger is located, regardless if there are fingers with *FMO* index smaller than the current *FMO*. For example: If in the above example (**Figure 26-124**) currently only 2 fingers are associated with this PCH, with *FMO* of '3' and '4' - the Sub-Slot data entry contains 'place-holders' in offsets 0,1 and 2, and the fingers data is located in offsets 3 and 4, respectively:

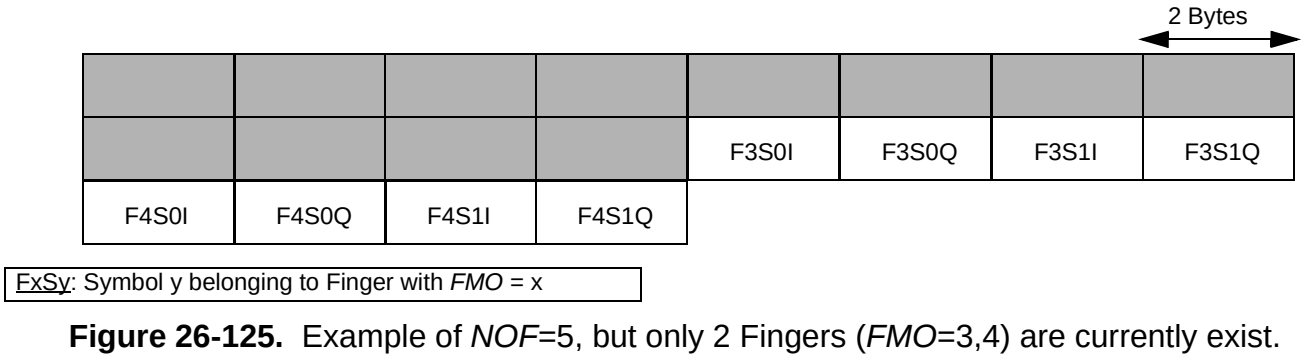


Figure 26-125. Example of *NOF*=5, but only 2 Fingers (*FMO*=3,4) are currently exist.

26.4.3.6.1.14.2 Finger Combining Enabled 16 bits Fixed Point Outputs (*FC_MODE* = 1)

In this mode, MAPLE-B2 performs FC to the symbols generated by each finger. The result of the FC is a 16-bit fixed point symbol, not normalized, with a scale of $2^{\wedge EXP}$ (*EXP* is described in the PCH command - **Section 26.5.4.6.3**). The symbols may become saturated if the user defined scale is smaller from the internally calculated scale. The Sub-Slot output data entry structure is:

Symbol 0 ; Symbol 1 ; ... ; Symbol ‘n-1’

where ‘n’ is defined as per **Table 26-103**.

Each symbol may contain an ‘I’ portion only or ‘Q’ portion only, depending on the *ODT* field of the PCH Command. **Figure 26-126** provides an example for the output data structure of a Sub-Slot, with the following properties:

- *SF* = 2 (Spreading Factor is 8)
- *ODT* = 0

2 Bytes

S0	S1	S2	S3	S4	S5	S6	S7
S8	S9	S10	S11	S12	S13	S14	S15
S16	S17	S18	S19	S20	S21	S22	S23
S24	S25	S26	S27	S28	S29	S30	S31

Figure 26-126. *Sub-Slot Output Data Structure for FC_MODE* = 1 Example

26.4.3.6.1.14.3 Finger Combining Enabled 32 bits Floating Point Outputs (*FC_MODE* = 3)

In this mode, MAPLE-B2 performs FC to the symbols generated by each finger. The result of the FC is a 32-bit floating point symbol composed of:

- 16 bits sign extended exponent generated as follows: *exponent*={10{*EXP[5]*},*EXP[5:0]*} where *EXP* is described in the PCH command - **Section 26.5.4.6.3**. The number represented by the exponent is $2^{exponent}$.
- 16 bits normalized mantissa.

The Sub-Slot output data entry structure is:

Symbol 0 mantissa; Symbol 0 exponent; Symbol 1 mantissa; Symbol 1 exponent;... ;
Symbol ‘n-1’ mantissa; Symbol ‘n-1’ exponent

where ‘n’ is defined as per **Table 26-103**.

Each symbol may contain an ‘I’ portion only or ‘Q’ portion only, depending on the *ODT* field of the PCH Command. **Figure 26-126** provides an example for the output data structure of a Sub-Slot, with the following properties:

- *SF* = 3 (Spreading Factor is 16)
- *ODT* = 0

2 Bytes

S0-M	S0-E	S1-M	S1E	S2-M	S2-E	S3-M	S3-E
S4-M	S4-E	S5-M	S5-E	S6-M	S6-E	S7-M	S7-E
S8-M	S8-E	S9-M	S9-E	S10-M	S10-E	S11-M	S11-E
S12-M	S12-E	S13-M	S13-E	S14-M	S14-E	S15-M	S15-E

Figure 26-127. *Sub-Slot Output Data Structure for FC_MODE* = 3 Example

26.4.3.6.1.14.4 System Output Buffers Structure

Each PCH has a cyclic buffer of multiple Sub-Slots data entries with parametrized base address, and size (see *PCHxBA* and *PCHxSZ* fields in **Section 26.5.3.5.2.1** and **Section 26.5.3.5.2.2**). MAPLE-B2 writes symbols to the output buffer while updating a write pointer to the buffer (see *PCHxWP* field in **Section 26.5.3.5.2.3**). When all PCH processing related to a certain Group in a certain Sub-Slot is completed, and the MAPLE-B2 got indication that all output data for this Sub-Slot is placed in external system buffers, it updates the Group Sub-Slot Write-Pointer (*MCUBGxC4P[GOBSWP]*). This pointer points to the index of the completed Sub-Slot within 2 frames time window (0...299).

26.4.3.6.1.14.5 Compressed Mode (*ODT* = 0x3)

Each PCH, regardless of the FC operating mode, may receive the *ODT* value of 0x3, which stands for compressed mode. While the PCH is compressed mode, it is considered as being processed for the purpose of Group Sub-Slot write pointer update, and output interrupt, but the PCH write pointer (*PCHxWP* field in **Section 26.5.3.5.2.3**) is not updated. A PCH may enter/exit compressed mode every Sub-Slot, using a PCH command.

26.4.3.6.1.15 Output Interrupt and Finished Channels Buffer Maintenance.

MAPLE-B2 uses a single interrupt as indication of finished channels. A finished channel is a channel for which MAPLE-B2 has processed a specified amount of Sub-Slots. It does not mean that the processing of that channel stops.

There is a single interrupt dedicated for all Groups/channels. Finished channel IDs are stored in a cyclic buffer, which once the interrupt is asserted, can be read by the host to identify the finished channels. For details see **Section 26.4.3.6.1.15.2**, *Finished Channel Buffer*, on page 26-230.

The host can define how often to perform a search for finished channels using the *OB_INT_CHK_STEP* field of the MCUBOBICP parameter. This field indicates the number of Sub-Slots between each such search and interrupt assertion. For details see **Section 26.4.3.6.1.15.3**, *Finished Channel Search.*

26.4.3.6.1.15.1 Finished Channel Definition

Finished Channels definition is done by configuring a pair of parameters for each PCH:

1. *MARK* - determines the next Sub-Slot index (0...1199) were the channel is considered finished.
2. *STEP* - determines the value by which *MARK* is increment once current *MARK* value has reached, that is, if *MARK* index is reached *MARK = MARK + STEP*.

The *MARK* and *STEP* fields can be configured in the *MCUBPCHxOBICP* parameter.

26.4.3.6.1.15.2 Finished Channel Buffer

Once a check point for a Group is reached (that is, *#OB_INT_CHK_STEP* Sub-Slots have passed for this Group from last check), the MAPLE-B2 searches for all the finished channels in the Group (see **Section 26.4.3.6.1.15.1**, *Finished Channel Definition*). Each finished channel ID is added to the finished channel IDs buffer (*MCUBFCBxP*) by writing its PCH ID (*PCH_ID* field in the PCH Command - **Section 26.5.3.5.2.3**), and the write pointer to the finished channel IDs buffer (*MCUBOBICP[FCWP]*) is updated accordingly. The MAPLE-B2 asserts the output interrupt only if finished channel IDs were added to the buffer during this search for this Group. When the host receives the output interrupt, it looks in the *MCUBFCBxP* for all the channels that were added since the last interrupt was serviced. The MAPLE-B2 uses the bd_done[31] interrupt line (see **Section 26.4.3.1.2**, *MAPLE-B2 Interrupts*) as the ULB_DONE output interrupt[1].

Note: The ULB_DONE interrupt must be initialized and enabled as described in **Section 26.4.3.1.2**, *MAPLE-B2 Interrupts*.

The finished channels buffer (*MCUBFCBxP*) is a cyclical buffer with 512 entries. The MAPLE-B2 adds finished buffers to this channel and updates the associated write pointer (*FCWP*) without checking for possible buffer overrun. It is up to the host to make sure no overrun occur in this buffer.

26.4.3.6.1.15.3 Finished Channel Search

When MAPLE-B2 processing reaches a check point for a Group (*#OB_INT_CHK_STEP* Sub-Slots have passed for this Group from last check), it then searches for all the PCH of the Group for which the current Group processed Sub-Slot index is greater (or equal) to the *MARK* value. Each such channel is added to the finished channels buffer and its *MARK* value is increment by *STEP*. An internal *MASK* bit (defined in the *MCUBPCHxOBICP* parameter*)* is used by MAPLE-B2 to overcome modulo calculations; it should be initialized to 1 when the channel opens.

Note: The above flow suggest that a finished channel is reported only during check points and on the same Sub-Slot index indicated by the *MARK* field. For example, if for certain PCH_X configuration MARK=430, Current Sub-Slot index for this Group is 400, *OB_INT_CHK_STEP* field is set to 50 and this Sub-Slot index for this Group is a check point. The next check point for this Group is only when Sub-Slot index gets to 450, and at that point only the PCH_X is identified as finished channel and the ULB_DONE interrupt is asserted.

In order for the finished channels buffer and interrupt assertion mechanism to work correctly the following limitations must apply:

1. The *STEP* fields of all PCH must be greater (or equal) than the *OB_INT_CHK_STEP* parameter definition.
2. The *MASK* bit must be set to ‘1’ during PCH initialization.
3. The *OB_INT_CHK_STEP* value must be configured according to the limitations as described in **Section 26.4.3.6.1.15.4**, *Finished Channel Search Limitations.*

1. If the CRPE-ULB functionality is enabled, the BD_DONE interrupt [31] should not be allocated to any of the other’s PE rings.

26.4.3.6.1.15.4 Finished Channel Search Limitations.

Each finished channels search check point invokes the internal RISC engine to perform a search as described above for all valid PCH in the group. Such a search, if performed in frequent check-points for each group (small value for the *OB_INT_CHK_STEP* parameter) and on high number of valid PCH per group may degrade the whole MAPLE-B2 performance as it utilizes the RISC on the expanse of other PEs. To avoid such degradations, the following equation describe the recommended *OB_INT_CHK_STEP* and *STEP* parameters configurations with respect to the total number of active PCH defined:

$$\frac{\sum_{\forall(x \in \{\text{active channels}\})} \left(10 + \frac{20}{\text{MCUBPCHxOBICP[STEP]}}\right)}{\text{MCUOBICP[OB\_INT\_CHK\_STEP]}} < 3001$$

Examples:

- If there are 100 active channels, each with *STEP* value of 1 (check every Sub-Slot) and the *OB_INT_CHK_STEP* is also set to 1, the result of the left side of the equation is 3000.
- If all channels are active (512 channels), each with *STEP* value of 2 and the *OB_INT_CHK_STEP* is set to 4, the result of the left side of the equation is 2560.

26.4.3.6.1.16 CRPE-ULB Performance

The CRPE-ULB performance is dictated by 3 main components :

- *Interpolation*. Affected by the number of Input antennas and the Interpolation over-sample.
- *Descrambling and Despreading*. Affected by the total number of fingers and their Spreading Factor
- *Maintenance*. Affected by the number of Finger commands, and a minimal amount of maintenance time where maintenance is the period between two successive Sub-Slots processing periods.

These components share the same ‘budget’ of 33,300 cycles per Sub-Slot, or 333,000 cycles per slot according to the following calculations (and assuming 500MHz clock frequency):

$$256_{\text{chips in sbs}} * 1/(3.84e6)_{\text{single chip time (sec.)}} * 500e6_{\text{cycles/sec.}} =\sim 33{,}300 \text{ cycles.}$$

Table 26-104 details how many cycles each ULB operation requires:

Table 26-104. CRPE-ULB Operations Performance (in cycles)

Operation	Operation Configuration	Cycles per Operation
Interpolation	*DLY_SPRD* = 0	36 cycles per ANT per OVS
	DLY_SPRD = 1	48 cycles per ANT per OVS
Descrambling and Despreading (DD)	Spreading Factor 2 (*SF* = 0)	32 cycles per Finger
	Spreading Factor 4 (*SF* = 1)	16 cycles per Finger
	Spreading Factor 8 (*SF* = 2)	8 cycles per Finger
	Spreading Factor 16 to 256 (*SF* = 2...7)	4 cycles per Finger
Maintenance	Minimal maintenance period per Group per Sub-Slot	1550 cycles
	'Modify Gain' Finger command	1 cycle per Finger
	'Add New' Finger command	4 cycles per Finger
	'Remove' Finger command	4 cycles per Finger
	'Modify' Finger command	6 cycles per Finger

The following equation calculates the amount of cycles required for the CRPE-ULB to perform its tasks per Sub-Slot:

ULB SBS Cycles =
*MAX(Interolation , DD)*1.25 +*
*MAX(minimal Maintenance,Fingers Commands execution)*1.1*

The following illustrates the CRPE-ULB performance calculations for an example use-case. The example assumes two Groups configuration where each Group includes:

- 24 Antennas
- Internal Interpolation with Delay Spread of up to 256 chips (*DLY_SPRD* = 0)
- Number of OVS is 8.
- Number of PCHs is 64
- Spreading Factor of PCHs is 4
- Number of Fingers per PCH is 8
- In 2 Sub-Slots (of the Slot) all fingers are modified ('Finger Modify' command).
- In 8 Sub-slots (of the Slot) no fingers command are executed.

As per the above configuration assumptions the CRPE-ULB performance (average calculation per Sub-Slot per Group):

Interpolation cycles = $24_{Ant.}$ * 8_{OVS} * 36_{cycles} = 6912 cycles.

DD cycles = 64_{PCH} * $8_{Fingers}$ * 16_{cycles} = 8192 cycles.

Finger Commands execution and Maintenance period (average per Sub-Slot) = $2/10_{SBS/Slot}$ * 64_{PCH} * $8_{Fingers}$ * 6_{cycles} + $8/10_{SBS/Slot}$ * 1550_{cycles} = 1855 cycles.

and the total Sub-Slot cycles of the CRPE-ULB per Group are:

ULB SBS Cycles = 8192*1.25 + 1855*1.1 = 12280 cycles.

Assuming the CRPE-ULB work on two such Groups, its utilization is estimated as:

CRPE-ULB utilization = (12280*2)/33300 = **~73%**

26.4.3.6.2 Chip Rate Uplink Fast Operation

The MAPLE-B2 supports UMTS (TS 25.211 and TS25.213) up-link Chip Rate processing for low latency Physical channels such as DPCCH, E-DPCCH, RACH and HS-DPCCH as described in **Section 26.2**, *MAPLE-B2 Features*. These operations are executed using the Chip Rate- Up-Link Fast Processing Elements (CRPE-ULF).

26.4.3.6.2.1 CRPE-ULF Initialization

The CRPE-ULF initialization is composed of:

- CRPE-ULF Parameters initialization done by the host after MAPLE-B2 initialization by API is completed and before the processing beginning of CRPE-ULF.
- CRPE-ULF Configuration Registers initialization done by the host after MAPLE-B2 initialization by API is completed and before the processing beginning of CRPE-ULF.

26.4.3.6.2.1.1 CRPE-ULF Parameters Initialization

After the MAPLE-B2 initialization routine and before the beginning of the processing by the CRPE-ULF, the following CRPE-ULF parameters must be initialized:

- MAPLE CRPE-ULF Command Input Buffer Address <x> Parameter (MCUFCIBAxP): This set of parameters holds the base address and the size of the command input buffers of the CRPE-ULF. See **Section 26.5.3.5.3.1**, *MAPLE CRPE-ULF Command Input Buffer Address <x> Parameter (MCUFCIBAxP)*, on page 26-364 for detail.
- MAPLE CRPE-ULF Command Input Buffer Write Pointer <x> Parameter (MCUFCIBWPxP): Holds the command input buffer write pointer. Should be set to zero during initialization. See **Section 26.5.3.5.3.2**, *MAPLE CRPE-ULF Command Input Buffer Write Pointer <x> Parameter (MCUFCIBWPxP)*, on page 26-365 for details.
- MAPLE CRPE-ULF Command Input Buffer Read Pointer <x> Parameter (MCUFCIBRPxP): Holds the command input buffer read pointer. Should be set to zero during initialization. See **Section 26.5.3.5.3.3**, *MAPLE CRPE-ULF Command Input Buffer Read Pointer <x> Parameter (MCUFCIBRPxP)*, on page 26-366 for details.

- MAPLE CRPE-ULF Interpolation Bypass General Configuration Parameter (MCUFIBGCP): This parameter holds the general configuration parameters for interpolation bypass mode. See **Section 26.5.3.5.3.4**, *MAPLE CRPE-ULF Interpolation Bypass General Configuration Parameter (MCUFIBGCP)*, on page 26-367 for details.
- MAPLE CRPE-ULF Interpolation Bypass Group Attributes <x> Parameter (MCUFIBGAxP): This set of parameters holds the group attributes for interpolation bypass mode. **Section 26.5.3.5.3.6**, *MAPLE CRPE-ULF Interpolation Bypass Group Attributes <x> Parameter (MCUFIBGAxP)*, on page 26-369 for details.
- MAPLE CRPE-ULF Interpolation Bypass Antenna Address <x> Parameter (MCUFIBAAxP): This set of parameters holds the base address of the antenna input buffers in system memory. See **Section 26.5.3.5.3.7**, *MAPLE CRPE-ULF Interpolation Bypass Antenna Address <x> Parameter (MCUFIBAAxP)*, on page 26-370 for details.

26.4.3.6.2.1.2 CRPE-ULF Configuration Registers Initialization

After the MAPLE-B2 initialization routine (see *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)) and before the beginning of the processing by the CRPE-ULF, the following CRPE-ULF configuration registers must be initialized:

- ULF General Configuration Register (ULFGCR): Includes general configuration parameters for the CRPE-ULF. See **Section 26.5.5.6.1**, *ULF General Configuration Register (ULFGCR)*, on page 26-532 for details.
- ULF Interpolation Configuration Register <x> (ULFICRx): This set of registers defines the interpolation weights of the CRPE-ULF. See **Section 26.5.5.6.3**, *ULF Interpolation Configuration Register <x> (ULFICRx)*, on page 26-534 for details.
- ULF Output Buffer <x> Base Configuration Register (ULFOBxBCR): This set of registers holds the base address of the output buffers of the CRPE-ULF. See **Section 26.5.5.6.4**, *ULF Output Buffer <x> Base Configuration Register (ULFOBxBCR)*, on page 26-536 for details.
- ULF Output Buffer <x> Attributes Configuration Register (ULFOBxACR): This set of registers hold the attributes parameters of the output buffers of the CRPE-ULF. See **Section 26.5.5.6.5**, *ULF Output Buffer <x> Attributes Configuration Register (ULFOBxACR)*, on page 26-537 for details.

26.4.3.6.2.2 CRPE-ULF Modes of Operation

The CRPE-ULF supports the following modes of operation set during the initialization sequence as described in **Section 26.4.3.6.2.1**, *CRPE-ULF Initialization*:

26.4.3.6.2.2.1 CRPE-ULF Interpolation Modes

The interpolation modes are set by writing to *ULFGCR[INB]* and *ULFGCR[OVS]*. The interpolation modes are summarized in **Table 26-105**:

Table 26-105. Interpolation Modes

INB	*OVS*	Description
0	0	Data is written to IB by Antenna I/F with OVS_2 and interpolated internally to OVS_2
0	1	Data is written to IB by Antenna I/F with OVS_2 and interpolated internally to OVS_4
0	2	Data is written to IB by Antenna I/F with OVS_2 and interpolated internally to OVS_8
0	3	Data is written to IB by Antenna I/F with OVS_2 and interpolated internally to OVS_{16}
1	0	non interpolated R_X data with OVS_2 is fetched from system memory by MAPLE-B2
1	1	Interpolated R_X data with OVS_4 is fetched from system memory by MAPLE-B2
1	2	Interpolated R_X data with OVS_8 is fetched from system memory by MAPLE-B2
1	3	Interpolated R_X data with OVS_{16} is fetched from system memory by MAPLE-B2

26.4.3.6.2.2.2 CRPE-ULF Delay Spread Modes

The delay spread mode is set by writing to *ULFGCR[MDS]*. The delay spread modes are described in **Table 26-106**

Table 26-106. Delay Spread Modes

MDS	Maximum Delay Spread	Max Number of PCHs
0	256 chips	512
1	512 chips	341

26.4.3.6.2.2.3 CRPE-ULF Pilot Correlation Modes

The PILOT correlation mode is set by writing to *ULFGCR[PCE]*. The PILOT correlation modes are described in **Table 26-107**

Table 26-107. Pilot Correlation Modes

DS	Description
0	Correlation with PILOT bits is disabled
1	Correlation with PILOT bits is enabled

26.4.3.6.2.2.4 CRPE-ULF Early/Late Modes

The Early-Late (EL) mode is set by writing to *ULFGCR[ELDx]*. The EL modes are described in **Table 26-108**

Table 26-108. EL Modes

ELD7	ELD6	ELD5	ELD4	ELD3	ELD2	ELD1	ELD0	Description
x	x	x	x	x	x	x	0	EL processing is enabled for PILOT Soft Symbol
x	x	x	x	x	x	0	x	EL processing is enabled for TFCI Soft Symbol
x	x	x	x	x	0	x	x	EL processing is enabled for FBI Soft Symbol

Table 26-108. EL Modes

ELD7	ELD6	ELD5	ELD4	ELD3	ELD2	ELD1	ELD0	Description
x	x	x	x	0	x	x	x	EL processing is enabled for TPC Soft Symbol
x	x	x	0	x	x	x	x	EL processing is enabled for E-DPCCH Soft Symbol
x	x	0	x	x	x	x	x	EL processing is enabled for HS-DPCCH Soft Symbol
x	0	x	x	x	x	x	x	EL processing is enabled for general purpose #0 Soft Symbol
0	x	x	x	x	x	x	x	EL processing is enabled for general purpose #1 Soft Symbol

26.4.3.6.2.3 CRPE-ULF Functional Description

The CRPE Uplink Fast processing receives up to 24 antennas data along with finger and physical channel commands from the system memory, process the data, and outputs it to symbol rings located within the system memory. The input data can be written to the MAPLE-B2 MBus Slave1 port (see **Figure 26-3**) by the CPRI or to be read from system memory for modes in which the interpolation is bypassed. The commands describe the physical channels and the fingers. The output data is written to system memory and calculated for the following three cases:

- Early
- On-time
- Late

Figure 26-128 describes data structures used for the CRPE-ULF processing

26.4.3.6.2.4 CRPE-ULF Data Organization

The MAPLE-B2 is programmed with information about up-to 3200 fingers belong to up to 512 Physical Channels. The CRPE-ULF correlate interpolated version of input data with the Spreading and Scrambling code of each finger. The accumulation results of all the fingers belong to the same PCH are packed and written to up-to 18 queues in system memory. The default mode to input data is by writing to the slave port of MAPLE-B2. Additional input data mode is the Interpolation Bypass mode in which data from system memory is being fetched by MAPLE-B2.

The IB data structure and memory map is configured by the following:

- *ULFGCR[IBM]* which selects the access mode to IB.
- *ULFGCR[INB]* which enables or disables interpolation logic.
- *ULFGCR[NOA]* which set the number of active antennas.
- *ULFGCR[OVS]* which set the over-sampling rate.

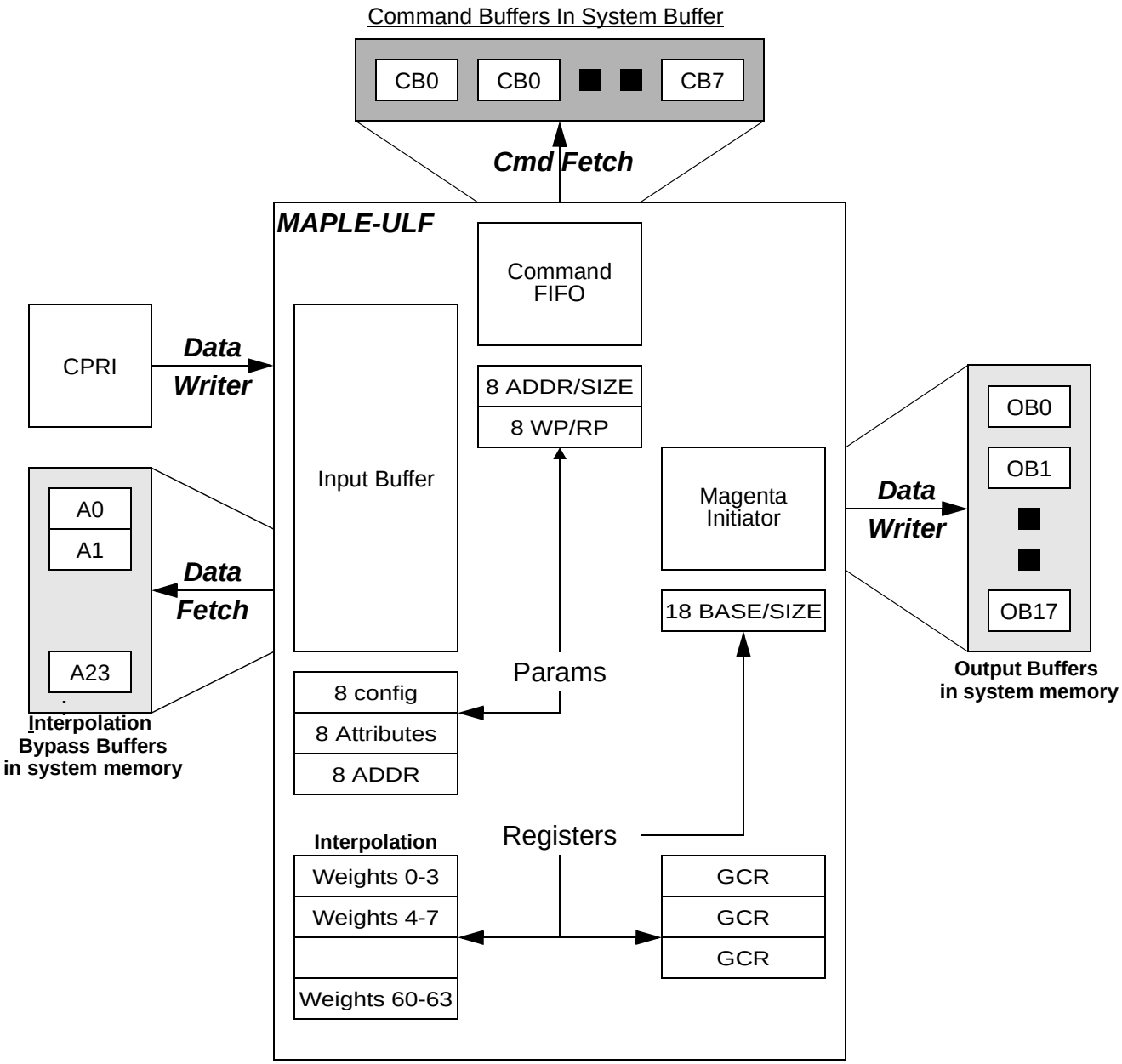


Figure 26-128. ULF Data Structure

26.4.3.6.2.5 Time Synchronization

ULF synchronizes to the system time by counting the 32 chip processing period logic. The time synchronization logic consists of a Global Chip Counter. The reset value of the counter is 76797, it's increment by 32 for each completed processing period and it's wrapped around on 2 frames boundary.

SBS	299								0			
PP	0	1	2	3	4	5	6	7	0	1	2	3
GCC	76573	76605	76637	76669	76669	76701	76733	76765	76797	29	61	93

SBS - Sub-Slot
PP - Processing Period
GCC - Global Chip Counter

Figure 26-129. Global Chip Counter implementation

26.4.3.6.2.6 Input Data Structures

The following subsections described the input data structures based on the interpolation modes selected.

26.4.3.6.2.6.1 Input Data Structure—Interpolation Enabled

In interpolation enabled mode, the Input Buffer (IB) is divided to up-to 24 pairs of cyclic buffer (up to a total 48 cyclic buffers). Each such pair of cyclic buffers contains data of single antenna such that the first cyclic buffer contains first samples of the chips and the second buffer contains the second sample of the chips. The size of each such cyclic buffer is 288 bytes and it can be accessed with write transactions granularity of 16 bytes. A pair of buffers is able to store up-to 144 chips such that the first sample of each chip is written to the first buffer and the second sample of each chip is written to the second buffer.

A chip is written to successive 2 bytes such that the most significant Byte stores the Real part and the least significant Byte stores the Imaginary part. **Figure 26-130** describes the data structure of a chip:

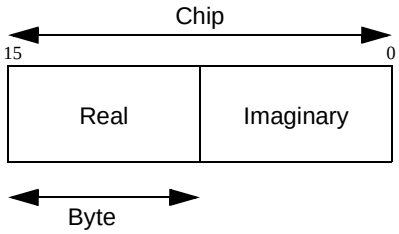


Figure 26-130. Data Structure of a Chip

There are 2 active buffers per active antenna. The number of active buffers is 2x(*ULFGCR[NOA]*+1). In interpolation enable mode, there are 2 modes to write the data to the

IB: Direct Mode and Window Mode. Choosing the input buffer mode is done by configuring the *ULFGCR[IBM]* bit.

Direct Mode

The data is written with real addresses of the IB in a cyclic order. which means that after writing chip to address 143, the next chip is written to address chip 0 and so on. **Figure 26-131** Describes the IB data structure with Interpolation in Direct mode:

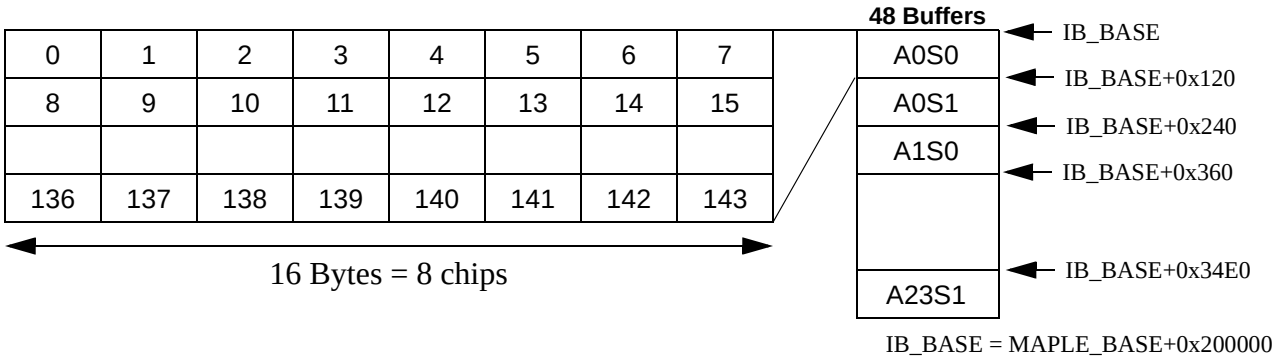


Figure 26-131. IB Data Structure Interpolation Enable - Direct Mode

Window Mode

The data is written to 4 virtual addresses of the buffer. Each buffer, that represent antenna and OVS, has access space for 32 chips. After writing the 32 chip, the next is written to chip 0. **Figure 26-132** describes the IB data structure with Interpolation in window mode:

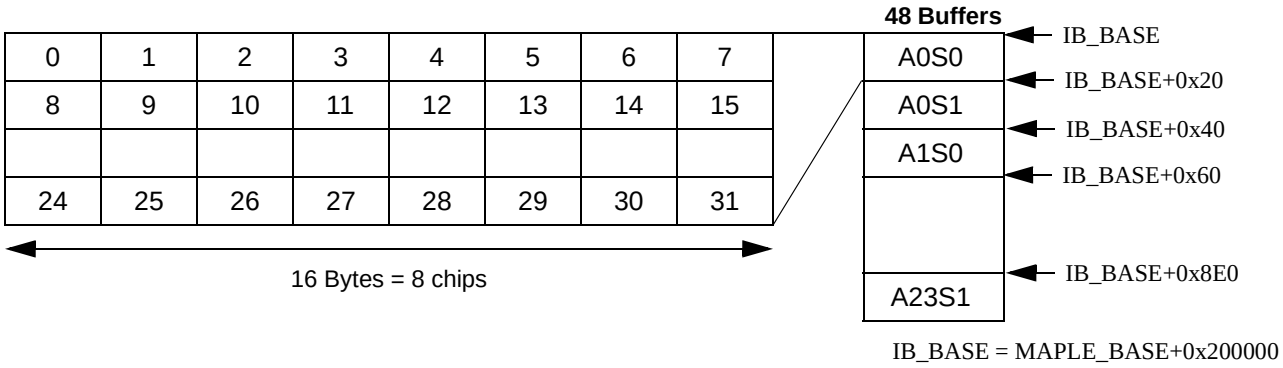


Figure 26-132. IB Data Structure Interpolation Enable

Note: The IB mode required when working in CPRI streaming mode or in interpolation bypass mode is the Window Mode (*ULFGCR[IBM]*=1*)*.

26.4.3.6.2.6.2 Input Data Structure—interpolation Bypass Mode

The IB is divided to buffers such that each buffer is a cyclic buffer which stores data of one antenna. The maximum number of antennas allowed and the number of chips in a buffer depend

on the Over Sampling rate. Interpolation bypass mode is set by setting the *ULFGCR[INB]* bit. The over sampling rate of x2, x4, x8 or x16 is set by setting *ULFGCR[OVS]* to 0, 1, 2 or 3 respectively.

Table 26-109 lists the number of maximum active antennas allowed (set by *ULFGCR[NOA]*) with respect to the over sampling rate (set by *ULFGCR[OVS]*):

Table 26-109. Maximum Number of Antennas in Interpolation Bypass Mode

OVS	Maximum NOA	Number of Bytes Per Chip	Size of IB per OVS per Ant
2	24	4	144 chips
4	24	8	72 chips
8	12	16	72 chips
16	6	32	72 chips

Note: When working with interpolation bypass mode, only IB Window Mode is supported (*ULFGCR[IBM]*=0). For details see **Section** , *Window Mode*, on page 26-240.

26.4.3.6.2.6.3 Input Data interface—Interpolation Bypass Mode

When the CRPE-ULF is working in Bypass mode the IB contains interpolated data and the interpolation filter is bypassed. The data is fetched by MAPLE-B2 from up-to 24 antenna cyclic buffers that located in system memory. The 24 Antenna cyclic buffers can be divided to up to 8 groups. Each group share the same write pointer which describe the buffer state of all group's antenna buffers.

The amount of groups, their size and the shared read pointer for all the groups are set in the *MCUFIBGCP* parameter (**Section 26.5.3.5.3.4**, *MAPLE CRPE-ULF Interpolation Bypass General Configuration Parameter (MCUFIBGCP)*). Each group is responsible for number of antennas according to the configuration of the *MCUFIBGAxP* parameter (**Section 26.5.3.5.3.6**, *MAPLE CRPE-ULF Interpolation Bypass Group Attributes <x> Parameter (MCUFIBGAxP)*). This parameter also includes the shared write pointer for the group.

Figure 26-133 illustrate 2 core groups that manage 2 antenna buffers. Each core group has its own write pointer and a mutual read pointer.

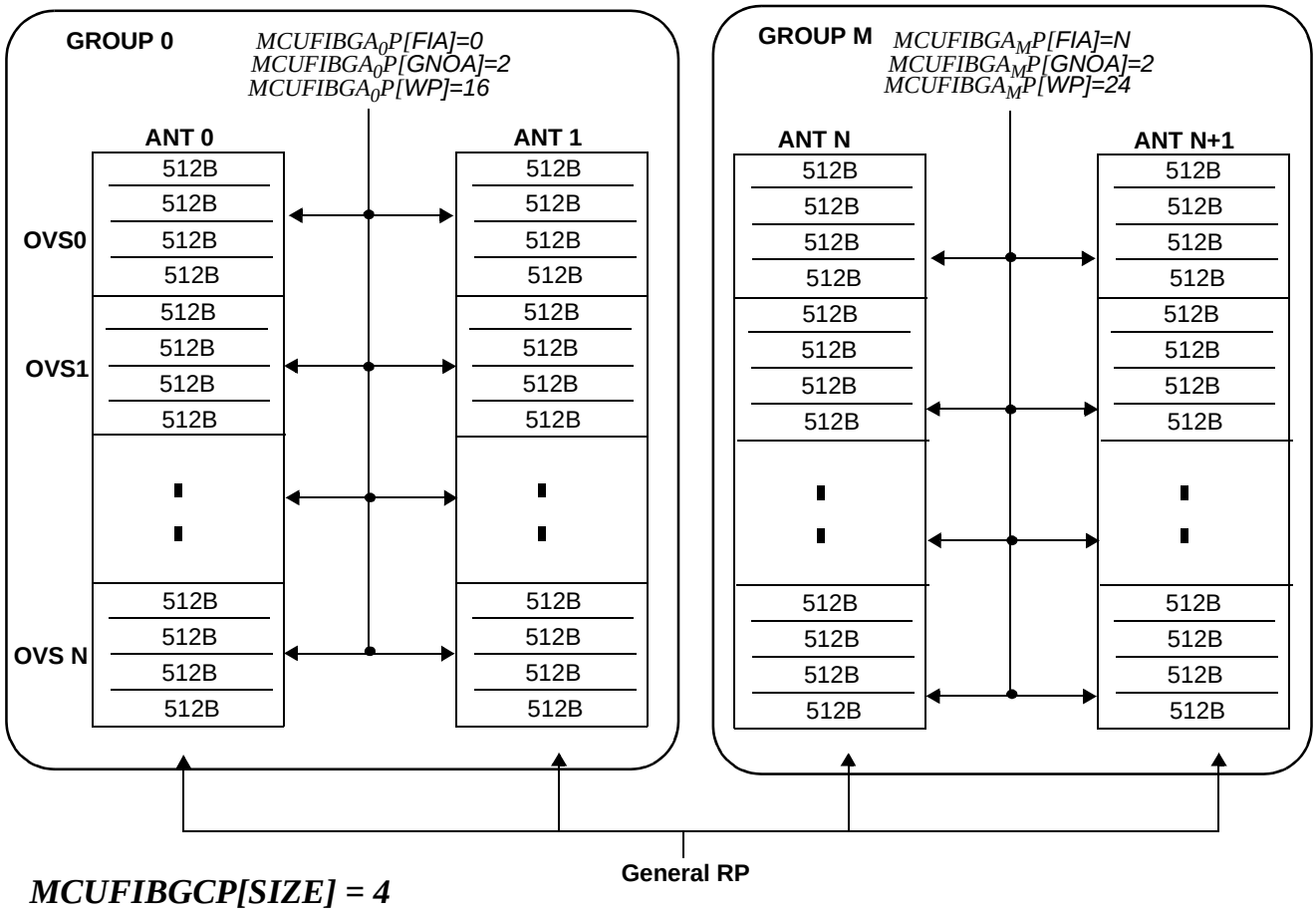


Figure 26-133. Antennas Group Input Data Structure

VCOP processes the next 64 bytes chunk of data for all antennas of all groups only if all the group's write pointers (*MCUFIBGAxP[WP]*) are bigger (cyclic pointers) than the global Read Pointer (*MCUFIBGCP[RP]*) and only after the previous data have been processed.

The MAPLE-B2 initiates the write/read pointers status check on the following internal events:

1. CRPE-ULF initiates an internal "Done" indication for processing the previous 32 chip data.
2. The CRPE-ULF timer indication event occur. The initial value of the timer is being configured in the *MCUFIBGCP[TMR]* field.

On every internal "Done" indication from the CRPE-ULF, the MAPLE-B2 check the pointers status. If there is no data to fetch, the MAPLE-B2 initiates the internal timer indicating the next time point to check the pointers. If in the next time point to check there is no data to fetch, the MAPLE-B2 continues with initiating the timer until data is valid to fetch.

Interpolation Bypass initialization

After MAPLE-B2 initialization, when configuring the CRPE-ULF to interpolation bypass mode and setting all relevant parameters, the host should trigger the `Maple_crpe_ulf_init` routine (see **Section 26.4.5**, *MAPLE-B2 Internal Task Control*, on page 26-312) to indicate the internal firmware to start fetching the data into the CRPE-ULF. Initiating this routine should be done once after initialization and only for Interpolation bypass mode.

26.4.3.6.2.7 Interpolation Processing

The interpolation logic is enabled if *ULFGCR[INB]* is cleared. Interpolation is implemented by an eight taps polyphase filter with programmable weights.

The interpolation logic calculates 8 phases for each sample. Each phase is calculated by eight programmable weights. Each weight is 8 bit Real which is multiplied by one of 4 predecessor or 4 successor samples. The phases are numbered from 0 to 7 such that interpolated phase 0 replaces a sample, phase 1 is closest to the four predecessor samples and phase 7 is closest to the 4 successor samples. **Figure 26-134** describes the interpolation of phase 4:

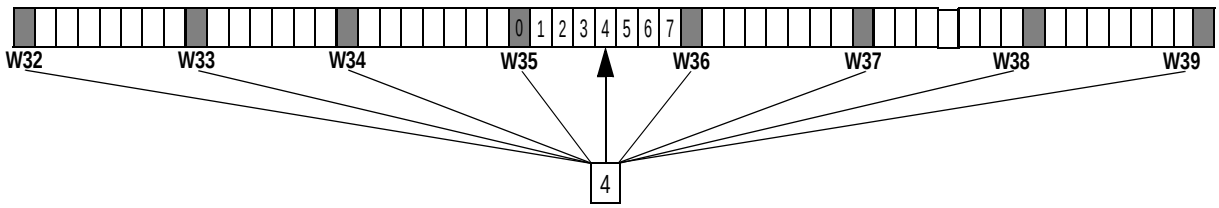


Figure 26-134. Interpolation of Sample 4

Table 26-110 describes the relation between the programmable weights and the interpolated sample:

Table 26-110. Interpolation Weights

Sample	Most far predecessor	Second Most far predecessor	Second closest predecessor	closest predecessor	closest successor	second closest successor	second most far successor	most far successor
0	ULFICR0[W0]	ULFICR0[W1]	ULFICR0[W2]	ULFICR0[W3]	ULFICR1[W4]	ULFICR1[W5]	ULFICR1[W6]	ULFICR1[W7]
1	ULFICR2[W8]	ULFICR2[W9]	ULFICR2[W10]	ULFICR2[W11]	ULFICR3[W12]	ULFICR3[W13]	ULFICR3[W14]	ULFICR3[W15]
2	ULFICR4[W16]	ULFICR4[W17]	ULFICR4[W18]	ULFICR4[W19]	ULFICR5[W20]	ULFICR5[W21]	ULFICR5[W22]	ULFICR5[W23]
3	ULFICR6[W24]	ULFICR6[W25]	ULFICR6[W26]	ULFICR6[W27]	ULFICR7[W28]	ULFICR7[W29]	ULFICR7[W30]	ULFICR5[W31]
4	ULFICR8[W32]	ULFICR8[W33]	ULFICR8[W34]	ULFICR8[W35]	ULFICR9[W36]	ULFICR9[W37]	ULFICR9[W38]	ULFICR9[W39]
5	ULFICR10[W40]	ULFICR10[W41]	ULFICR10[W42]	ULFICR10[W43]	ULFICR11[W44]	ULFICR11[W45]	ULFICR11[W46]	ULFICR11[W47]
6	ULFICR12[W48]	ULFICR12[W49]	ULFICR12[W50]	ULFICR12[W51]	ULFICR13[W52]	ULFICR13[W53]	ULFICR13[W54]	ULFICR13[W55]
7	ULFICR12[W56]	ULFICR12[W57]	ULFICR12[W58]	ULFICR12[W59]	ULFICR13[W60]	ULFICR13[W61]	ULFICR13[W62]	ULFICR13[W63]

Given 8 complex samples with 8 bits Real and 8 bits Imaginary, and given 8 corresponding weights (8 bit width) the interpolation is done according the following pseudo code:

```
Let R0,R1 ... R7 be the Real part of the complex samples
Let I0,I1 ... I7 be the Imaginary part of the complex samples
Let W0,W1 ... W7 be the corresponding weights
Let R be the real part of the interpolated sample
Let I be the Imaginary part of the interpolated sample
         R = (R0xW0 + R1xW1 + R2xW2 + R3xW3 + R4xW4 + R5xW5 + R6xW6 + R7xW7)>> ULFGCR[IS]
         I = (I0xW0 + I1XW1 + I2xW2 + I3xW3 + I4xW4 + I5xW5 + I6xW6 + I7XW7)>> ULFGCR[IS]
// Saturation:
         If R>127 then R=127
         If R<-128 then R = -128

         If I>127 then I=127
         If I<-128 then I= -128
```

26.4.3.6.2.8 Despreading and Descrambling (DD)

The DD operation refers to extracting a Soft Symbol of a specific finger out of a stream of chips. For E-/DPCCH which are spread with SF=256, the DD operation produces one Soft Symbol for each 256 chips. The DD is done by selecting 256 samples chips belong to the same Soft Symbol and correlating them with the PCH's scrambling code and OVSF. If the PILOT Soft Symbol is expected and *ULFGCR[PCE]* = 1, the chips are correlated with the expected pilot bit too.

26.4.3.6.2.8.1 Antenna Selection

The antenna interpolated input is selected by the *AID* field of the Finger Command (*FIC[AID]*) such that if *FIC[AID]*=0, correlation is done on antenna #0 data, if *FIC[AID]*=1, correlation is done on antenna #1 data,..., and if *IFC[AID]* = 23, correlation is done on antenna #23 data. *FIC[AID]* must be less than or equal to *ULFGCR[NOA]*. For details on the Finger Command data structure see **Section 26.4.3.6.2.11.1**, *Finger Command (FIC) Structure*.

26.4.3.6.2.8.2 Chip Selection.

The DD operation is done on the 256 chips. The selection of the chips depends on the internal Global Chip Counter (GCC), the PCH Command sub-slot start attribute (*PCHC[SBS]*) and the finger offset (*FIC[FOF]*). For details on the PCH Command data structure see **Section 26.4.3.6.2.11.2**, *Physical Channel Command Structure (PCHC)*

Figure 26-135 describes the chips selection for two fingers belong to the same PCH. The first finger is with offset FOF0 chips relative to *PCH[SBS]* and the second finger is with offset FOF1 chips relative to *PCH[SBS]*.

- For processing its first Soft Symbol (F0SS0), the first finger selects the chips (SBS*256 + FOF0) to (SBS*256 + FOF0 + 255)
- For processing its second Soft Symbol (F0SS1), the first finger selects the chips ((SBS+1)*256 + FOF0) to ((SBS+1)*256 + FOF0 + 255)
- For processing its first Soft Symbol (F1SS0), the second finger selects the chips (SBS*256 + FOF1) to (SBS*256 + FOF1 + 255).

- For processing its second Soft Symbol (F1SS1), the second finger selects the chips ((SBS+1)*256 + FOF1) to ((SBS+1)*256 + FOF1 + 255).

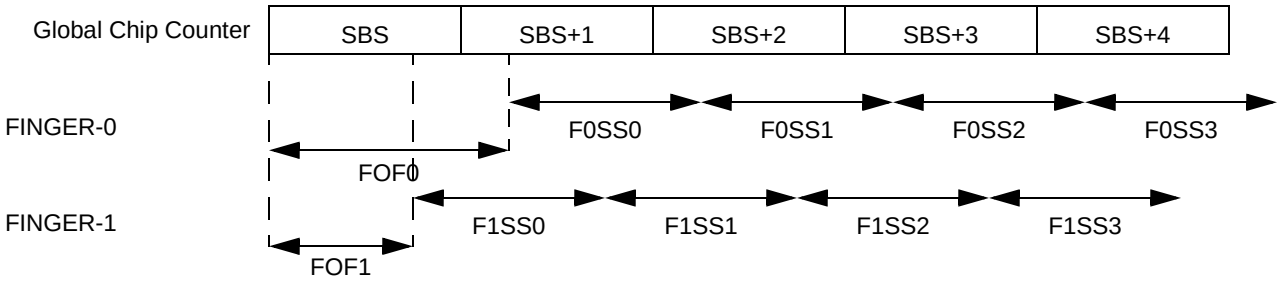


Figure 26-135. Chip Selection

26.4.3.6.2.8.3 Physical Channel (PCH) and Slot Format (SLF) Selection

DD processing supports 4 physical channel types: DPCCH, E-DPCCH, HS-DPCCH and PRACH.

- DPCCH produced 10 bits of data with the following structure:
  - 3 to 8 PILOT bits
  - 0, 2, 3 or 4 Transport Format Combination Indicator bits (TFCI)
  - 0 or 1 Feedback Information bits (FBI)
  - 2 or 4 Transport Power Control bits (TPC)
- E-DPCCH and HS-DPCCH produced 10 bits of data with no division to groups.
- RACH produces 8 PILOT bits and 2 TFCI bits.

The *PCHC[SLF]* field selects the Physical Channel type and the Slot Format as described in **Table 26-111**.

Table 26-111. ULF Supported Slot Formats

SLF	PCH	SBS0	SBS1	SBS2	SBS3	SBS4	SBS5	SBS6	SBS7	SBS8	SBS9
0	DPCCH	PILOT#0	PILOT#1	PILOT#2	PILOT#3	PILOT#4	PILOT#5	TFCI#0	TFCI#1	TPC#0	TPC#1
1	DPCCH	PILOT#0	PILOT#1	PILOT#2	PILOT#3	PILOT#4	TFCI#0	TFCI#1	TFCI#2	TPC#0	TPC#1
2	DPCCH	PILOT#0	PILOT#1	PILOT#2	PILOT#3	TFCI#0	TFCI#1	TFCI#2	TFCI#3	TPC#0	TPC#1
3	DPCCH	PILOT#0	PILOT#1	PILOT#2	PILOT#3	PILOT#4	PILOT#5	PILOT#6	PILOT#7	TPC#0	TPC#1
4	DPCCH	PILOT#0	PILOT#1	PILOT#2	PILOT#3	PILOT#4	TFCI#0	TFCI#1	FBI#0	TPC#0	TPC#1
5	DPCCH	PILOT#0	PILOT#1	PILOT#2	PILOT#3	TFCI#0	TFCI#1	TFCI#2	FBI#0	TPC#0	TPC#1
6	DPCCH	PILOT#0	PILOT#1	PILOT#2	TFCI#0	TFCI#1	TFCI#2	TFCI#3	FBI#0	TPC#0	TPC#1
7	DPCCH	PILOT#0	PILOT#1	PILOT#2	PILOT#3	PILOT#4	PILOT#5	PILOT#6	FBI#0	TPC#0	TPC#1
8	DPCCH	PILOT#0	PILOT#1	PILOT#2	PILOT#3	PILOT#4	PILOT#5	TPC#0	TPC#1	TPC#2	TPC#3
9	E-DPCCH	DATA#0	DATA#1	DATA#2	DATA#3	DATA#4	DATA#5	DATA#6	DATA#7	DATA#8	DATA#9
10	PRACH	PILOT#0	PILOT#1	PILOT#2	PILOT#3	PILOT#4	PILOT#5	PILOT#6	PILOT#7	TFCI#0	TFCI#1

Table 26-111. ULF Supported Slot Formats

SLF	PCH	SBS0	SBS1	SBS2	SBS3	SBS4	SBS5	SBS6	SBS7	SBS8	SBS9
11	HS-DPCCH	DATA#0	DATA#1	DATA#2	DATA#3	DATA#4	DATA#5	DATA#6	DATA#7	DATA#8	DATA#9
12	General Purpose #0	DATA#0	DATA#1	DATA#2	DATA#3	DATA#4	DATA#5	DATA#6	DATA#7	DATA#8	DATA#9
13	General Purpose #1	DATA#0	DATA#1	DATA#2	DATA#3	DATA#4	DATA#5	DATA#6	DATA#7	DATA#8	DATA#9

26.4.3.6.2.8.4 Scrambling Sequence Generation

The DD logic correlates the Rx buffer data with a complex-valued scrambling sequences. The scrambling sequence may be either a short scrambling sequence or a long scrambling sequence. The short and long scrambling sequences supported by DD logic complies with the long and short scrambling codes defines in *3GPP TS 25.211 v9.0.0 (2009-09)* sub clause 4.3.2.

A scrambling sequence is a series of 38400 complex numbers such that each complex number is either 1+j, 1-j, -1+j or -1 -j. There are 2^{24} different short scrambling sequences and 2^{24} different long scrambling sequences.

The DD logic uses the PCH Command (*PCHC*) attributes to select the PCH unique scrambling sequence, and together with the Finger Command (*FIC*) attributes and the Global Chip Counter, adjusts the scrambling sequence to the finger's offset.

The DD logic uses the Long/Short Scrambling sequence indication (*PCHC[LSS]*) and the Scrambling Sequence Number (*PCHC[SSN]*) to select the scrambling sequence:

- If *PCHC[LSS]* = 0, the DD logic uses *PCHC[SSN]* to select one of 2^{24} long scrambling sequences.
- If *PCHC[LSS]* = 1, the DD logic uses *PCHC[SSN]* to select one of 2^{24} short scrambling sequences.

In each processing period and for each finger the DD logic calculates 32 complex values which are correlated with the Rx data buffer. The complex values are calculated as followed:

```
Let S[i] be the i^th complex value of the selected scrambling sequence (i=0 to 38399)
Let S_pp[j] be the complex value to be correlate with the j^th chip of the PP. (j = 0 to 31)
Parameters:
GCC - internal Global Chip Counter
SBO - offset of the scrambling sequence in resolution of sub-slot and relative to GCC.
To be defined in the PCH Command (PCHC)
FOF - the finger's offset. To be defined in the Finger Command (FIC)
for(i=0 to 31)
        Spp[i] = S[(GCC+ i + SBO*256 - FOF)%38400]
```

Note: The scrambling sequence is shifted one chip to the left to support Early/On-time/Late (EOL) processing.

Figure 26-136 describes an example of scrambling sequences calculations with the following parameters:

- S1 is calculated with *PCHC[SBS]* = 63, *PCHC[SBO]* = 0 and *PCHC[FOF]*=0
- S2 is calculated with *PCHC[SBS]* = 63, *PCHC[SBO]* = 1 and *PCHC[FOF]*=0
- S3 is calculated with *PCHC[SBS]* = 63, *PCHC[SBO]* = 1 and *PCHC[FOF]*= 270

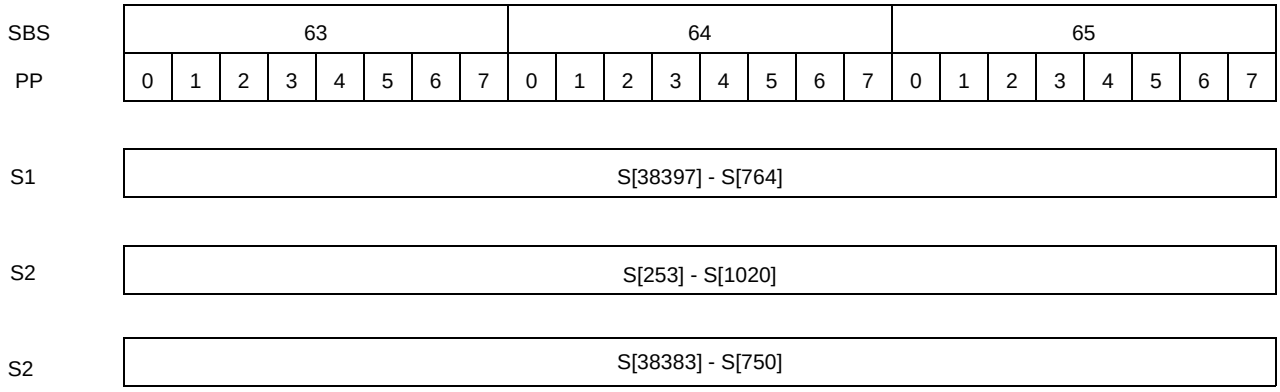


Figure 26-136. Scrambling Sequences Calculation Example

26.4.3.6.2.8.5 OVSF Code Generation

The **O**rthogonal **V**ariable **S**preading **F**actor (OVSF) code is a series of 256 real values which can be 1 or -1. CRPE-ULF supports the following OVSF codes:

- OVSF 256,0 for DPCCH.
- OVSF 256,1 for E-DPCCH.
- OVSF 256,16i+15 (i=0 to 15) for PRACH control part.
- OVSF 256,1, 256,31, 256,32 and 256,64 for HS-DPCCH

26.4.3.6.2.8.6 Pilot Code Generation

The DD logic correlates the Rx Buffer data with the pilot code when: *ULFGCR[PCE]* = 1, the PCH is DPCCH and during sub-slot when a pilot Soft Symbol is expected (as describes in **Table 26-111**). When *PCHC[SLF]*=0 or *PCHC[SLF]*= 8, there are 6 pilot bits per slot and total of 90 pilot bits in a frame. **Table 26-112** describes the pilot code when *PCHC[SLF]*=0 or *PCHC[SLF]*=8:

Table 26-112. PILOT Bits for *PCHC[SLF]*=0 or *PCHC[SLF]*=8

sub-slot/slot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0
2	1	0	1	0	0	1	1	0	1	1	1	0	0	0	0

Table 26-112. PILOT Bits for *PCHC[SLF]*=0 or *PCHC[SLF]*=8

sub-slot/slot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	0	0	0	1	0	0	1	1	0	1	0	1	1
5	0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
6–9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

When *PCHC[SLF]* =1 or *PCHC[SLF]*=4, there are 5 pilot bits per slot and total of 75 pilot bits in a frame. **Table 26-113** describes the pilot code when *PCHC[SLF]*=1 or *PCHC[SLF]*=4:

Table 26-113. PILOT Bits for *PCHC[SLF]*=1 or *PCHC[SLF]*=4

sub-slot/slot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0
1	1	0	1	0	0	1	1	0	1	1	1	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	0	0	0	1	0	0	1	1	0	1	0	1	1
4	0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
5–9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

When *PCHC[SLF]* =2 or *PCHC[SLF]* = 5, there are 4 pilot bits per slot and total of 60 pilot bits in a frame. **Table 26-114** describes the pilot code when *PCHC[SLF]*=2 or *PCHC[SLF]* = 5:

Table 26-114. PILOT Bits for *PCHC[SLF]*=2 or *PCHC[SLF]*=5

sub-slot/slot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0
2	1	0	1	0	0	1	1	0	1	1	1	0	0	0	0
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4–9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

When *PCHC[SLF]* =3 or *PCHC[SLF]* =10, there are 8 pilot bits pre slot and total of 120 pilot bits in a frame. **Table 26-115** describes the pilot code when *PCHC[SLF]*=3 or *PCHC[SLF]* =10:

Table 26-115. PILOT Bits for *PCHC[SLF]*=3, *PCHC[SLF]*=10

sub-slot/slot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	0	1	0	0	1	1	0	1	1	1	0	0	0	0

Table 26-115. PILOT Bits for *PCHC[SLF]*=3, *PCHC[SLF]*=10

sub-slot/slot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	0	0	0	1	0	0	1	1	0	1	0	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
8-9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

When *PCHC[SLF]* =6, there are 3 pilot bits per slot and total of 45 pilot bits in a frame. **Table 26-116** describes the pilot code when *PCHC[SLF]*=6:

Table 26-116. PILOT Bits for *PCHC[SLF]*=6

sub-slot/slot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0
1	1	0	1	0	0	1	1	0	1	1	1	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3-9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

When *PCHC[SLF]* =7, there are 7 pilot bits per slot and total of 105 pilot bits in a frame. **Table 26-117** describes the pilot code when *PCHC[SLF]*=7:

Table 26-117. PILOT Bits for *PCHC[SLF]*=7

sub-slot/slot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0
2	1	0	1	0	0	1	1	0	1	1	1	0	0	0	0
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	0	0	0	1	0	0	1	1	0	1	0	1	1
5	0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7-9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

In each processing period and for each finger the DD logic calculates 32 real values which are correlated with Rx Buffer data. The real value are calculated as followed:

```
Let PILOT[s][j][k] be the pilot bit for SLF=s, for the j^th slot and the k^th sub-slot.
Let P[i] be the real value correlate with the i^th chip of the processing period (i=0 to 31)
Parameters:
        GCC - Global Chip Counter
        SLF - Slot Format of the PCH
        FOF - Finger Offset
```

```
for(i=0 to 31)
        j = FLOOR((GCC + i - FOF)/2560)
        k = FLOOR((GCC + i - FOF)/256)
        if(PILOT[SLF][j][k] == 1)
           P[i] = 1;
        else
           P[i] = -1
```

Note: The Pilot sequence is shifted one chip to the left to support Early/On-time/Late (EOL) processing.

26.4.3.6.2.8.7 DPCCH/PRACH PILOT Correlation

The DPCCH PILOT correlation is a function which produces a 16 bits Real and 16 bits Imaginary complex-valued soft symbol. The DPCCH PILOT correlation is used when the PCH is DPCCH, the current Soft Symbol is pilot and *ULFGCR[PCE]* = 1. The correlation function gets as an input the following parameters:

- 256 chips represented by a complex number with 8 bits Real part and 8 bits Imaginary part.
- 256 scrambling complex numbers which may be 1+j, 1-j, -1+j or -1-j.
- 256 spreading numbers which can be either 1 or -1
- A pilot number which can be either 1 or -1

The correlation function is described in the following pseudo code:

```
Let A[i] + B[i]j be the i^th chip
Let C[i] + D[i]j be the i^th scrambling number
Let E[i] be the i^th spreading number
Let P be the pilot number
Let S_I be the Real part of the output soft symbol
Let S_Q be the Imaginary part of the output soft symbol

S_I = 0;
S_Q = 0;

# Real part and Imaginary part of (A[i]+B[i]j)*(C[i]-D[i]j)*E[i]*(-Pj)
for(i=0;i<256;i++){
        S_I += P*E[i]*B[i]*C[i] - P*E[i]*A[i]*D[i];
        S_Q += -P*E[i]*A[i]*C[i] -P*E[i]*B[i]*D[i];
}
S_I = S_I>>ULFGCR[DDS];
S_Q = S_Q>>ULFGCR[DDS];

if(S_I > 32767)
        S_I = 32767;
if(S_I < -32768)
        S_I = -32768;
if(S_Q > 32767)
        S_Q - 32767;
if(S_Q < -32768)
        S_Q = -32768;
```

26.4.3.6.2.8.8 E-DPCCH and HS-DPCCH Data Correlation.

The E/HS-DPCCH data correlation is a function which produces a 16 bits Real and 16 bits Imaginary complex-valued soft symbol. The correlation function gets as an input the following parameters:

- 256 chips represented by a complex number with 8 bits Real part and 8 bits Imaginary part.
- 256 scrambling complex numbers which may be 1+j, 1-j, -1+j or -1-j.
- 256 spreading numbers which can be either 1 or -1

The correlation function is described in the following pseudo code:

```
Let A[i] + B[i]j be the i^th chip
Let C[i] + D[i]j be the i^th scrambling number
Let E[i] be the i^th spreading number

Let S_I be the Real part of the output soft symbol
Let S_Q be the Imaginary part of the output soft symbol

S_I = 0;
S_Q = 0;

for(i=0;i<256'i++){
    S_I += E[i]*A[i]*C[i] + E[i]*B[i]*D[i]; # Real part of (A[i]+B[i]j)*(C[i]-D[i]j)*E[i]
    S_Q += E[i]*B[i]*C[i] - E[i]*A[i]*D[i]; # Imaginary of (A[i]+B[i]j)*(C[i]-D[i]j)*E[i]
}
S_I = S_I>>ULFGCR[DDS];
S_Q = S_Q>>ULFGCR[DDS];

if(S_I > 32767)
    S_I = 32767;
if(S_I < -32768)
    S_I = -32768;
if(S_Q > 32767)
    S_Q - 32767;
if(S_Q < -32768)
    S_Q = -32768;
```

26.4.3.6.2.8.9 Early/On-time/Late (EOL) Processing

EOL processing refers to processing a finger three times with fixed offset forward and backward relative to the offset set by *FIC[FOF]*. The EOL is enabled depending on the type of the soft symbol and on the corresponding *ULFGCR[ELDx]* bit:

- For PILOT Soft Symbol, EOL is enabled only if *ULFGCR[ELD0]* is reset.
- For TFCI Soft Symbol, EOL is enabled only if *ULFGCR[ELD1]* is reset.
- For FBI Soft Symbol, EOL is enabled only if *ULFGCR[ELD2]* is reset.
- For TPC Soft Symbol, EOL is enabled only if *ULFGCR[ELD3]* is reset.
- For ED-DPCCH Soft Symbol, EOL is enabled only if *ULFSCR[ELD4]* is reset.
- For HS-DPCCH Soft Symbol, EOL is enabled only if *ULFSCR[ELD5]* is reset.

- For General Purpose #0 Soft Symbol, EOL is enabled only if *ULFSCR[ELD6]* is reset.
- For General Purpose #1 Soft Symbol, EOL is enabled only if *ULFSCR[ELD7]* is reset.

When EOL is enabled DD processing is done on three fingers: Early finger, On-Time finger and Late finger. These three fingers produces three Soft Symbol respectively: Early Soft Symbol, On-Time Soft Symbol and Late-Soft Symbol.

The offset of the EOL fingers, in resolution of 1/16 chips, is set by *FIC[FOF]*, *FIC[SID]* and *PCHC[ELO]* as follows:

- The offset of the Early finger is *FIC[FOF]**16 + *FIC[SID]* - *PCHC[ELO]* - 1.
- The offset of the On-Time finger is *FIC[FOF]**16 + *FIC[SID]*.
- The offset of the Late finger is *FIC[FOF]**16 + *FIC[SID]* + *PCHC[ELO]* + 1.

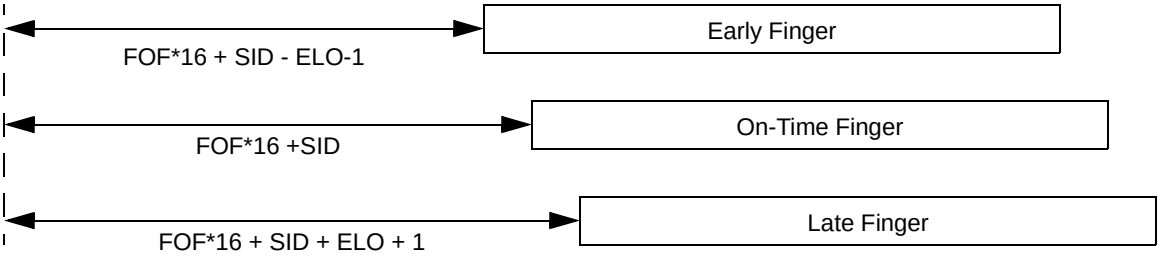


Figure 26-137. EOL Offset

Some notes:

- Due to EL offset the DD processing requires 34 chips during each processing periods.
- During the very first Processing Period (PP), the DD logic uses chips number –2 to chips number 31 such that chips –2 and –1 are zeroes.
- The On-Time finger is shifted one chip to the left, that is, in the very first PP, when ignoring *FIC[FOF]* it aligned to chips –1 to 30. The Early and Late fingers are with offset of *PCHC[ELO]* relative to the On-Time Finger.
- In each processing period the EOL fingers are progressed 32 chips forward.

Figure 26-138 describes the EOL fingers during the very first PP.

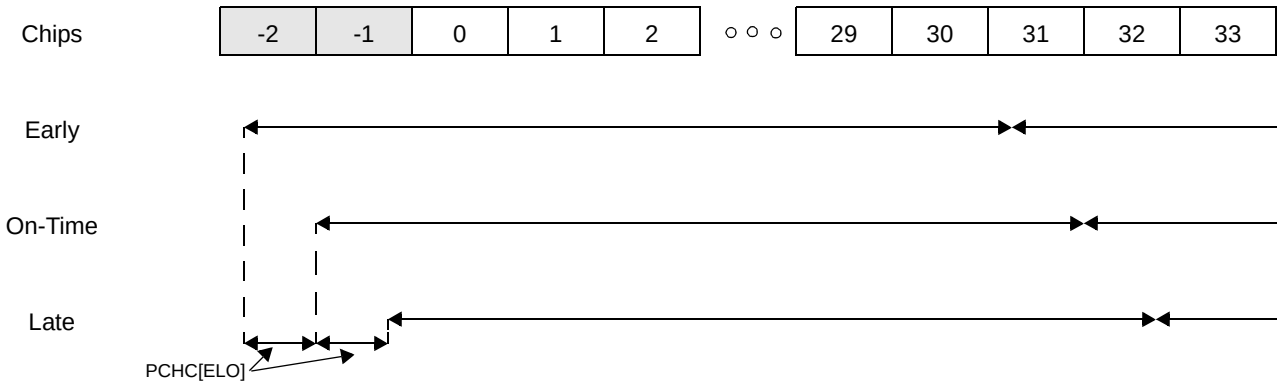


Figure 26-138. EOL Fingers in the Very First PP

26.4.3.6.2.8.10 Updating Finger Offset

The finger offset can be reprogrammed by issuing a Finger Command (FIC) with new Finger Offset (*FIC[FOF]*) and/or Sample ID (*FIC[SID]*) fields while preserving the old PCH ID (*FIC[PCHID]*), Antenna ID (*FIC[AID]*) and Finger Number (*FIC[FIN]*) values. The new offset applies in the sub-slot number set by *FIC[SBS]* field. When moving a finger forward, the DD logic produces the last Soft Symbol of the old finger (for sub-slot *FIC[SBS]*-1), stops processing during the gap between the old finger and the new finger, and restarts processing at the new start position of the new finger.

When moving a finger backward, the DD logic prematurely produces the last Soft Symbol of the old finger in the start position of the new finger. As a result the last Soft Symbol of the old finger (for sub-slot *FIC[SBS]*-1) would be less accurate. An alternative for moving a finger backward is killing the old finger after it produces the Soft Symbol for sub-slot SBS-1 and open a new finger with new offset to replace it from *FIC[SBS]* sub-slot index and on. In this case, the new finger must have new *FID*, may have new *FOF* and *SID*, and must have old *PCHID* and *FIN*.

Note: Eliminating a finger is done by setting its Antenna ID (*FIC[AID]*) to *ULFGCR[NOA]*.

Moving fingers options are described in **Figure 26-139**:

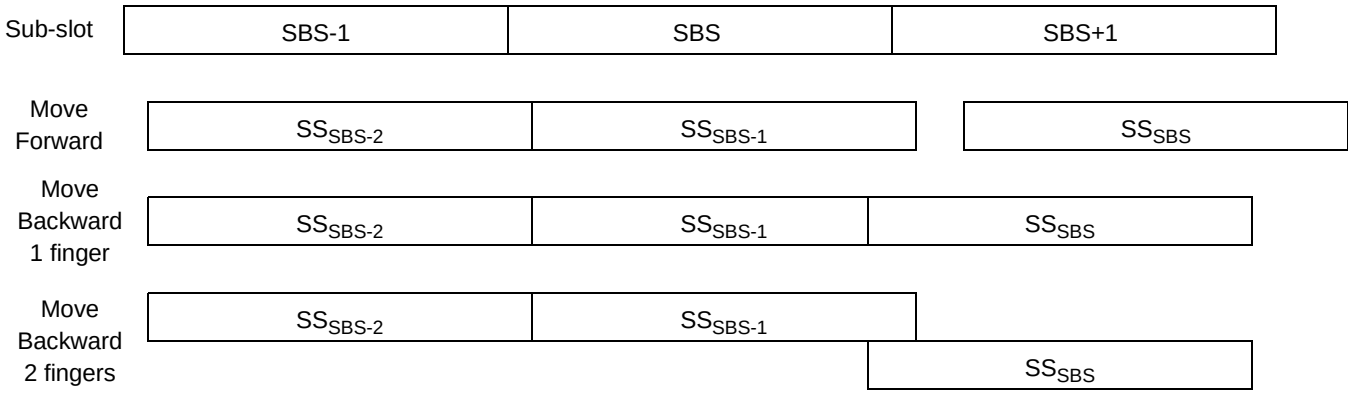


Figure 26-139. Moving Fingers options

If the finger Antenna ID (*FIC[AID]*) is changed on the fly the DD logic does not calculate the tail of the old Soft Symbol on the PP it starts the new Soft Symbol. Instead, the DD logic outputs the last intermediate result calculated in the previous PP.

The following example describes changing Finger's Antenna ID on the fly. The DD logic calculates SS0 and SS1 using data received on antenna #0 and calculates SS2 on data received on antenna #1. During SBS0 PP1, the DD logic calculates the tail of SS0 and outputs the symbol. It also starts to calculate SS1 and output the first intermediate result. During SBS1 PP1, the DD logic does not calculate the tail of SS1; instead, it outputs the last intermediate result for SS1 (which is calculated during SBS1 PP0). It also starts calculating SS2 and outputs its first intermediate result.

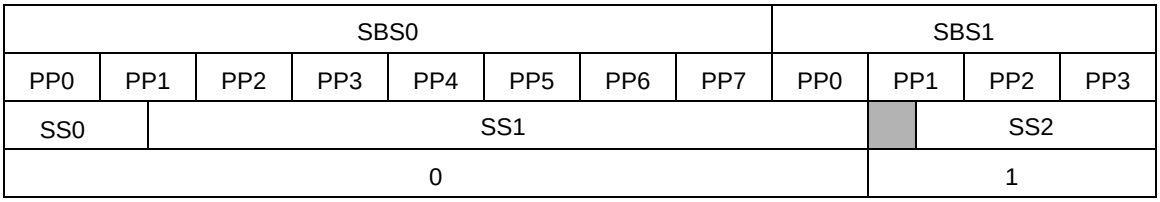


Figure 26-140. Changing AID On-the-Fly

26.4.3.6.2.9 Compressed PCHs

Compressed PCHs are transmitted on selected slots of a frame as indicated by *PCHC[SEx]*. The DD logic is activated only for a slot for which its corresponding *PCHC[SEx]* bit is set. The index of the slot is calculated with regard to the Global Clock Counter (GCC), the Sub-Slot Offset (*PCHC[SBO]*), the Sub-Slot Start indication (*PCHC[SBS]*), and the Finger Offset (*FIC[FOF]*) as described in the following pseudo code:

```
Let SI[i] be the SLOT index for the i^th chip of the processing period (i=0 to 31)
Parameters:
        GCC - Global chip counting.
        SBO - Sub-slot offset of the PCH
        SBS - Sub-slot when the PCH is activated
        FOF - Finger offset

for(i=0;i<32;i++)
        SI[i]= Floor((GCC + i - (SBS%150)*256 +SBO*256 - FOF)%38400/2560)
```

26.4.3.6.2.10 CRPE-ULF Output Data

The following subsections describe the output data structure for CRPE-ULF processing.

26.4.3.6.2.10.1 Soft Symbol Packet Structure

The Soft Symbol of all fingers belong to the same PCH are packed and written to system memory. There are two types of packet, the first type of packet (Type-0) is used when Aery/Late processing is disabled and the second type of packet (Type-1) is used when the EL is enabled.

- Type-0 packet consists of 64 bits header followed by On-Time Soft Symbols (SSs).
- Type-1 packet consists of 64 bits header followed by On-Time SSs, followed by Early SSs, followed by Late SSs.

The packets have the following characteristics:

- In both types of packets, a Soft Symbol is a complex number with 16 bits Real part and 16 bits Imaginary part.
- In both types of packets the Soft Symbol are ordered with respect to the finger number (*FIC[FIN]*) such that the first soft symbol belongs to the finger with *FIC[FIN]*=0, the second Soft Symbol belong to the finger with *FIC[FIN]*=1 and the last Soft Symbol belong to the finger with *FIC[FIN]* value equals to the *NOF* field of the Soft Symbol Packet Header (see **Section 26.4.3.6.2.10.2**, *Soft Symbol Packet Header*, on page 26-256).
- In both types of packets zero soft symbols are added after the last Soft Symbol such that the size of the packets is integer multiplication of 16 Bytes.

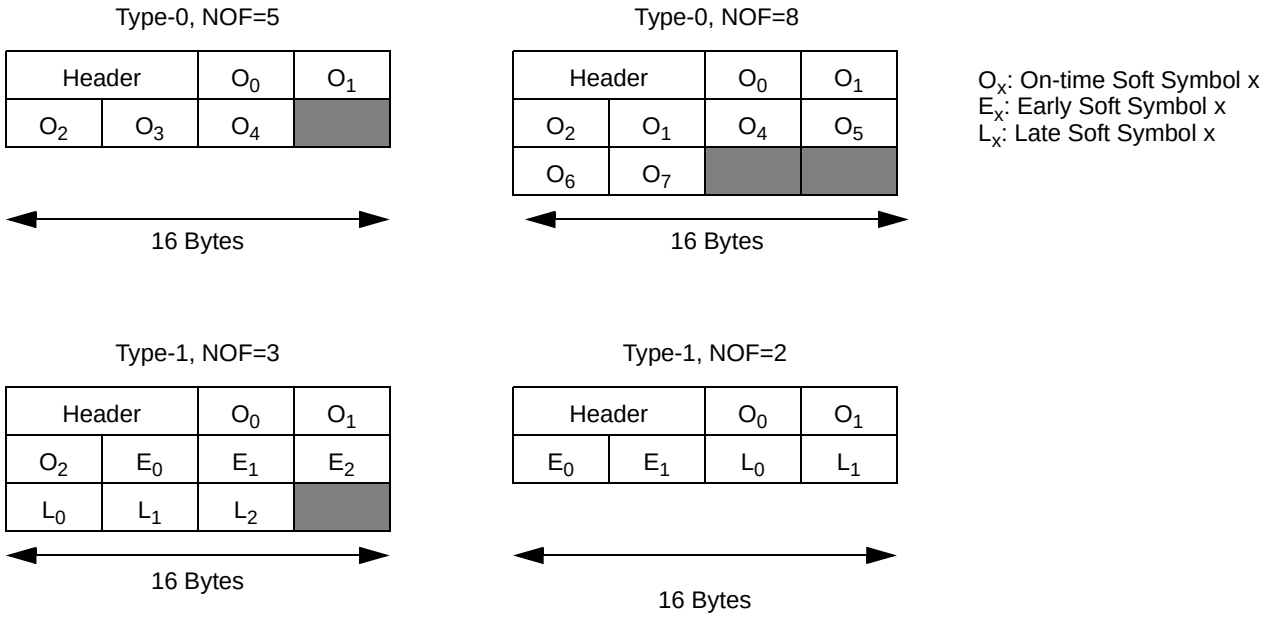


Figure 26-141. Type-0 and Type-1 Packets examples

26.4.3.6.2.10.2 Soft Symbol Packet Header

Figure 26-142 Describes the Soft Symbol Packet Header

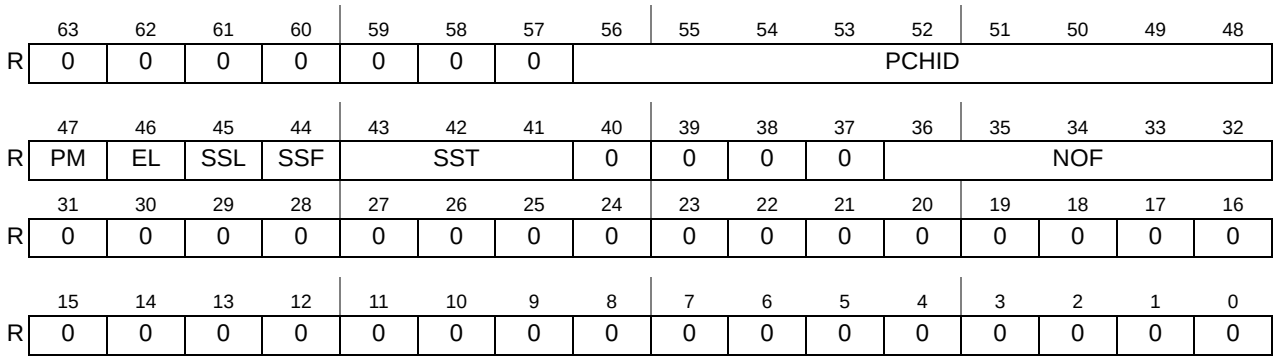


Figure 26-142. Soft Symbol Packet Header structure

Table 26-118 Describes the Soft Symbol Packet Header fields description:

Table 26-118. Soft Symbol Packet Header Field Descriptions

Field	Bits	Description
—	63–57	Reserved
PCHID	56–48	**PCHID**. The PCH ID to which this packet belong to.
PM	47	**Pilot Mark**. This bit is set if the current Soft Symbol type is PILOT and the index of the PILOT bit is equal to the *PMS* field in the PCH Command (PCHC).
EL	46	**Early Late**. Early/Late Indication. 0 - The packet does not contain Early/Late Soft Symbols 1 - The packet contains Early/Late Soft Symbols

Table 26-118. Soft Symbol Packet Header Field Descriptions (Continued)

Field	Bits	Description
SSL	45	**Soft Symbol Last**. Last soft symbol of a Type indication. 0 - The packet does not contain a last Soft Symbol of a Type 1 - The packet contains a last Soft Symbol of a Type
SSF	44	**Soft Symbol First**. First soft symbol of a Type indication. 0 - The packet does not contain a first Soft Symbol of a Soft Symbol Type 1 - The packet contains a first Soft Symbol of a Type
SST	43–41	**Soft Symbol Type**. 0: the packet contains DPCCH/PRACH PILOT Soft Symbol 1: the packet contains DPCCH/PRACH TFCI Soft Symbol 2: the packet contains DPCCH FBI Soft Symbol 3: the packet contains DPCCH TPC Soft Symbol 4: the packet contains E-DPCCH Soft Symbol 5: the packet contains HS-DPCCH Soft Symbol 6: the packet contains General Purpose #0 symbol 7: the packet contains General Purpose #1 symbol
—	40–37	Reserved
NOF	36–32	**Number Of Fingers**. The number of finger in the packet. 0 - The packet includes single finger. 1 - The packet includes 2 fingers. ... 31 - The packet includes 32 fingers.
—	31–0	Reserved

26.4.3.6.2.10.3 System Memory Output Buffers Structure

The Soft Symbol packets are written to up-to 18 cyclic buffers in the system memory. Each buffer has a programmable base address set by its corresponding *ULFOBxBCR[BASE]*. The base address must be 16 bytes aligned. The Size the buffers, are set by *ULFOBxACR[SIZE]* with granularity of 16 bytes. The first 16 bytes of a buffer stores the buffer's status information: Write Pointer (WP) and Time Stamp (TIME). The Write Pointer is initialized to *ULFOBxBCR[BASE]*+16 and it wraps back to *ULFOBxBCR[BASE]*+16 address when it reaches *ULFOBxBCR[BASE]+ULFOBxACR[SIZE]* address.

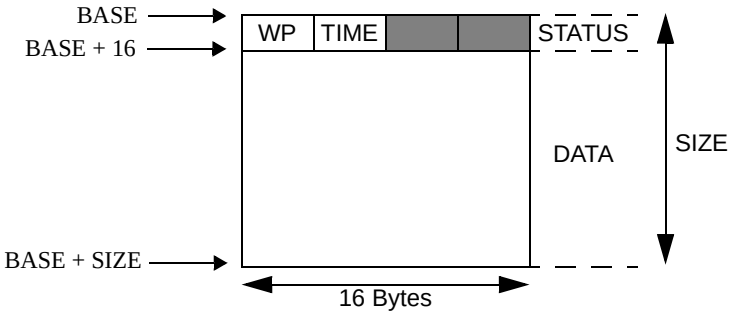


Figure 26-143. Output Buffer Structure

Each buffer includes a Group ID (*GID*) and Type Enable (*TEx*) configuration. A packet is directed to one of the buffers according to its Group ID and Type (*GID* and *SST* fields of the Packet Header), that is, a packet is directed to a certain buffer only if the packet and the buffer

share the same Group ID (*GID*) and if the type of the packet (as indicated by header's *SST* field) is enabled for this buffer (as indicated by *ULFOBxACR[TEx]*).

Note: It is the responsibility of the host to configure the CRPE-ULF output buffer such that a packet is directed to no less and no more than single output buffer.

Table 26-119 shows an example in which the PCHs are divided between 6 cores such that each core uses three output buffers. The first output buffer is used to store PILOT Soft Symbol, the second is used to store low latency Soft Symbol such as FBI and TPC and the third output buffer is used to store high latency Soft Symbol such as TFCI and E-DPCCH and HS-DPCCH.

Table 26-119. Output Buffer Programming, 6 groups, Fast/Slow

Buffer	GID	TE0 (PILOT)	TE1 (TFCI)	TE2 (FBI)	TE3 (TPC)	TE4 (E-DPCCH)	TE5 (HS-DPCCH)
0	0	1	0	0	0	0	0
1	0	0	0	1	1	0	0
2	0	0	1	0	0	1	1
3	1	1	0	0	0	0	0
4	1	0	0	1	1	0	0
5	1	0	1	0	0	1	1
6	2	1	0	0	0	0	0
7	2	0	0	1	1	0	0
8	2	0	1	0	0	1	1
9	3	1	0	0	0	0	0
10	3	0	0	1	1	0	0
11	3	0	1	0	0	1	1
12	4	1	0	0	0	0	0
13	4	0	0	1	1	0	0
14	4	0	1	0	0	1	1
15	5	1	0	0	0	0	0
16	5	0	0	1	1	0	0
17	5	0	1	0	0	1	1

26.4.3.6.2.10.4 Writing the Output Data Results

The CRPE-ULF uses a dedicated MBus Initiator (MI) interface (sharing MBus Master 3 with internal DMA engine) to write packets toward system memory. The transactions priority is set by configuring the *ULFGCR[MP]*.

The MI updates the Output Buffers (OB) status once, twice, four times or eight times in a processing period (32 chips) by writing the new Write Pointer (WP) and the new Time Stamp (TIME) to the base address of the OB as defined in *ULFOBxBCR[BASE]*. Controlling the update of the output buffers is done by setting the *ULFGCR[OBUR]*.

26.4.3.6.2.11 CRPE-ULF Command Flow

The CRPE-ULF is controlled by two types of commands:

- Physical Channel Commands (PCHC)
- Finger Commands (FIC).

A PCHC sets attributes which are common to all the fingers belong to the same PCH. A FIC set attributes which are specific to a finger. MAPLE-B2 is able to receive up-to 3200 FICs and up-to 512 PCHCs in time frame of a sub-slot. It can receive FIC commands to an already activated fingers and PCHC command to an already activated PCHs.

The new commands is activated according to the command's *SBS* (sub-slot start) field. MAPLE-B2 is able to receive one command for a Finger or PCH at a time, i.e writing a new command to a Finger/PCH while a previous command to the same Finger/PCH is pending is not allowed. The Commands are generated by the host (or multiple hosts) and written to dedicated queues in system memory as defined in the *MCUFCIBAxP[BASE]* fields. The size of each such queue is defined in *MCUFCIBAxP[SIZE]*. The MAPLE-B2 is responsible to fetch the commands from system memory queues as per CRPE-ULF internal requests. For details on the command fetching see **Section 26.4.3.6.2.11.3**, *Command Fetching Description*.

26.4.3.6.2.11.1 Finger Command (FIC) Structure

Figure 26-144 Describes the Finger Command (FIC) structure:

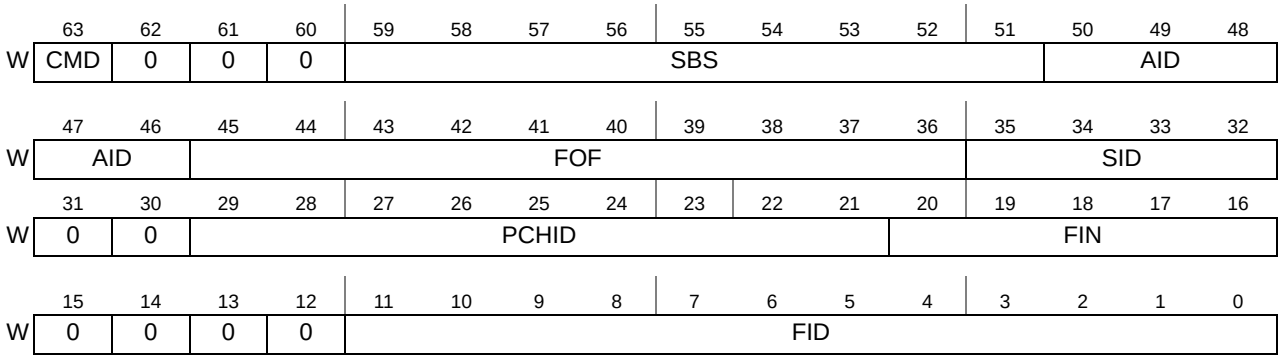


Figure 26-144. Finger Command Structure

Table 26-120 Describes the Finger Command field description:

Table 26-120. Finger Command Fields

Field	Bits	Description
CMD	63	**Command**. This bit should be set to 0 to indicate it's a Finger Command.
	62-60	Reserved
SBS	59–51	**Sub-Slot Start.** Indicates the activation time of the command in resolution of sub-slot. Valid range: 0 to 299
AID	50–46	**Antenna ID**. Selects the antenna input buffer. To disable the finger, AID should be set to *ULFGCR[NOA]*
FOF	45–36	**Finger Offset**. The finger's offset with regard to PCH start point and with resolution of 1 chip. If *ULFGCR[MDS]*=1, maximum offset allowed is 767. If *ULFGCR[MDS]*=0, maximum offset allowed is 511.

Table 26-120. Finger Command Fields

Field	Bits	Description
SID	35–32	**Sample ID**. Selects one sample out of 16 samples of a chip. If *ULFGCR[OVS]*=0, SID can be either 0 or 8. If *ULFGCR[OVS]*=1, SID can be either 0, 4, 8 or 12. If *ULFGCR[OVS]*=2, SID can be either 0, 2, 4, 6, 8, 10, 12 or 14. If *ULFGCR[OVS]*=3, SID can be any number between 0 to 15.
	31–30	Reserved.
PCHID	29–21	**PCH ID**. The ID of the PCH to which the finger belongs to. ***Note***: Once a PCHID is set it can't be reconfigure as long as the finger is active.
FIN	20–16	**Finger Number.** The index of the finger in the PCH packet. Valid Range: 0 to 31
	15–12	Reserved.
FID	11:0]	**Finger ID**. The ID of the finger to which the command refers to.

26.4.3.6.2.11.2 Physical Channel Command Structure (PCHC)

Figure 26-145 Describes the PCHC structure

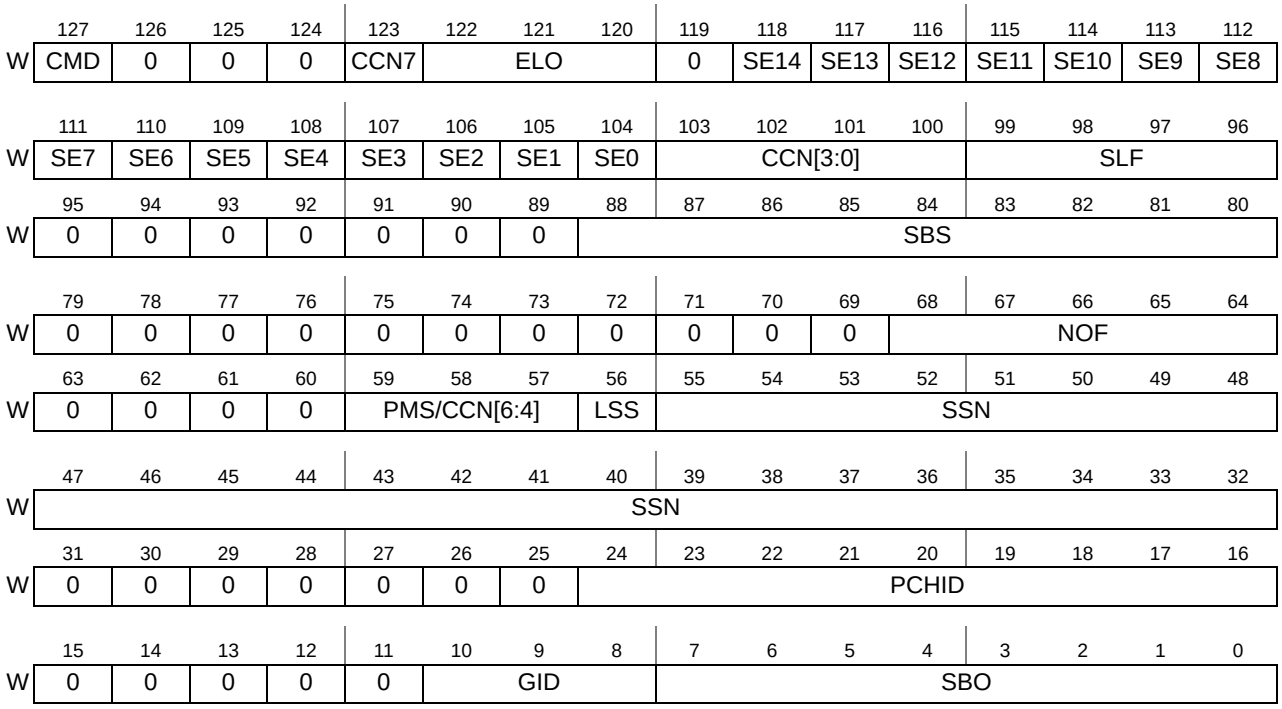


Figure 26-145. PCHC Structure

Table 26-121 Describes the PCH command field description:

Table 26-121. PCHC Field Descriptions

Field	Bits	Description
CMD	127	**Command ID**. Should be set to 1 to indicate PCH Command.
	126–124	Reserved.

Field	Bits	Description
CCN7	123	**Channelization Code Number bit 7**. This field together with *CCN[6:4]* and *CCN[3:0]* explicitly selects the channelization code number when using general purpose slot format (SLF=12 or SLF=13). For details see CCN[3:0] definition
ELO	122–120	**Early Late Offset**. 0: EL offset is 1/16, this value is allowed when *ULFGCR[OVS]* = 3. 1: EL offset is 2/16, this value is allowed when *ULFGCR[OVS]* > 1. 2: EL offset is 3/16, this value is allowed when *ULFGCR[OVS]* = 3. 3: EL offset is 4/16, this value is allowed when *ULFGCR[OVS]* > 0. 4: EL offset is 5/16, this value is allowed when *ULFGCR[OVS]* = 3. 5: EL offset is 6/16, this value is allowed when *ULFGCR[OVS]* > 1. 6: EL offset is 7/16, this value is allowed when *ULFGCR[OVS]* = 3. 7: EL offset is 8/16 ***Note:*** Once ELO is set it can't be reconfigure as long as the PCH is active (that is, as long as there are active fingers belong to this PCH).
	119	Reserved.
SE14	118	**Slot Enable 14**. 0 - The PCH is not transmitted on slot #14 1 - The PCH is transmitted during on slot #14
SE13	117	**Slot Enable 13**. 0 - The PCH is not transmitted on slot #13 1 - The PCH is transmitted during on slot #13
SE12	116	**Slot Enable 12**. 0 - The PCH is not transmitted on slot #12 1 - The PCH is transmitted during on slot #12
SE11	115	**Slot Enable 11**. 0 - The PCH is not transmitted on slot #11 1 - The PCH is transmitted during on slot #11
SE10	114	**Slot Enable 10**. 0 - The PCH is not transmitted on slot #10 1 - The PCH is transmitted during on slot #10
SE9	113	**Slot Enable 9**. 0 - The PCH is not transmitted on slot #9 1 - The PCH is transmitted during on slot #9
SE8	112	**Slot Enable 8**. 0 - The PCH is not transmitted on slot #8 1 - The PCH is transmitted during on slot #8
SE7	111	**Slot Enable 7**. 0 - The PCH is not transmitted on slot #7 1 - The PCH is transmitted during on slot #7
SE6	110	**Slot Enable 6**. 0 - The PCH is not transmitted on slot #6 1 - The PCH is transmitted during on slot #6
SE5	109	**Slot Enable 5**. 0 - The PCH is not transmitted on slot #5 1 - The PCH is transmitted during on slot #5
SE4	108	**Slot Enable 4**. 0 - The PCH is not transmitted on slot #4 1 - The PCH is transmitted during on slot #4
SE3	107	**Slot Enable 3.** 0 - The PCH is not transmitted on slot #3 1 - The PCH is transmitted during on slot #3
SE2	106	**Slot Enable 2**. 0 - The PCH is not transmitted on slot #2 1 - The PCH is transmitted during on slot #2

Field	Bits	Description
SE1	105	**Slot Enable 1**. 0 - The PCH is not transmitted on slot #1 1 - The PCH is transmitted during on slot #1
SE0	104	**Slot Enable 0.** 0 - The PCH is not transmitted on slot #0 1 - The PCH is transmitted during on slot #0
CCN[3:0]	103–100	**Channelization Code Number bits [3:0]**. – For slot format 0 to 9 this field is ignored. – For slot format 10 this field selects the channelization code number as follows: 0: channelization code number is 256,15 1: channelization code number is 256,31 2: channelization code number is 256,47 3: channelization code number is 256,63 4: channelization code number is 256,79 5: channelization code number is 256,95 6: channelization code number is 256,111 7: channelization code number is 256,127 8: channelization code number is 256,143 9: channelization code number is 256,159 10: channelization code number is 256,175 11: channelization code number is 256,191 12: channelization code number is 256,207 13: channelization code number is 256,223 14: channelization code number is 256,239 15: channelization code number is 256,255 – For slot format 11 this field selects the channelization code number as follows 0: channelization code number is 256,1 1: channelization code number is 256,31 2: channelization code number is 256,32 3: channelization code number is 256,64 – For slot format 12 and slot format 13 this field, together with *CCN[7]* and *CCN[6:4]*, explicitly selects the channelization code number.
SLF	99–96	**Slot Format**. This field sets the slot format and the PCH: 0: DPCCH Slot Format #0 1: DPCCH Slot Format #0A 2: DPCCH Slot Format #0B 3: DPCCH Slot Format #1 4: DPCCH Slot Format #2 5: DPCCH Slot Format #2A 6: DPCCH Slot Format #2B 7: DPCCH Slot Format 3 8: DPCCH Slot Format 4 9: E-DPCCH 10: PRACH control part 11: HS-DPCCH 12: General purpose slot format #0 13: General purpose slot format #1 14–15: Reserved.
	95–89	Reserved
SBS	88–80	**Sub-Slot Start**. Indicates the activation time of the command in resolution of sub-slot. Valid range: 0 to 299
	79–69	Reserved

Field	Bits	Description
NOF	68–64	**Number of Fingers**. The number of fingers belong to the PCH. 0 - The PCH include single finger 1 - The PCH include 2 fingers. ... 31 - The PCH include 32 fingers
	63–60	Reserved
PMS/ CCN[6:4]	59–57	**Pilot Mark Select** For slot formats 0 to 8 and for slot format 10, this field set the index of the pilot bit to which the PM bit in the packet header is set: 0: The Pilot Mark bit is set for PILOT #0 1: The Pilot Mark bit is set for PILOT #1 2: The Pilot Mark bit is set for PILOT #2 3: The Pilot Mark bit is set for PILOT #3 4: The Pilot Mark bit is set for PILOT #4 5: The Pilot Mark bit is set for PILOT #5 6: The Pilot Mark bit is set for PILOT #6 7: The Pilot Mark bit is set for PILOT #7 For slot formats 9, 11, 12 and 13, the PMS field is ignored. **Channelization Code Number bits [6:4]** For slot formats 12 and 13, this field together with CCN7 and CCN[3:0] explicitly selects the channelization code number.
LSS	56	**Long Short Select**. Selects between long and short scrambling sequences: 0 - A long scrambling code is selected for the PCH. 1 - A short scrambling code is selected for the PCH.
SSN	55–32	**Scrambling Sequence Number**. Indicates the PCH's scrambling sequence number.
	31–25	Reserved
PCHID	24–16	**PCH ID**. The ID of the PCH to which this command refers to. Valid range: 0 to 511
	15–11	Reserved
GID	10–8	**Group ID**. This field set the group ID to which this PCH belong to 0: The PCH belongs to group 0 1: The PCH belongs to group 1 2: The PCH belongs to group 2 3: The PCH belongs to group 3 4: The PCH belongs to group 4 5: The PCH belongs to group 5 6–7: Reserved
SBO	7–0	**Sub-Slot Offset**. Indicates index of the sub-slot where the command is activated. 0–149: Sub-Slot index. 150–255: Reserved.

26.4.3.6.2.11.3 Command Fetching Description

The MAPLE-B2 gets Finger and Physical channel commands by reading from up to 8 cyclic buffers in system memory. The buffers address and size are configured in the *MCUFCIBAxP* parameters (**Section 26.5.3.5.3.1**, *MAPLE CRPE-ULF Command Input Buffer Address <x> Parameter (MCUFCIBAxP)*, on page 26-364). The buffer defined by this address space is cyclic, which means that when the pointer equal to the size, it is set to zero.

The MAPLE-B2 fetches up to 1919 commands from the command buffers (up to 8) on every Processing Period (32 chips time), depend on the internal CRPE-ULF command fifo. The

MAPLE-B2 scan the external command buffers in a cyclic manner, that is, it starts by fetching all available commands from the first buffer, moving on to the next buffer and so on. After the last buffer is reached, it starts from the first buffer again. An external buffer is valid when its size (*MCUFCIBAxP[SIZE]*) is larger than zero and its Write Pointer (*MCUFCIBWPxP[WP]*) is greater (cyclic manner) than the Read Pointer (*MCUFCIBRPxP[RP]*). The host should update the Write Pointer when adding new commands in the system buffer, and the MAPLE-B2 update the Read Pointer when fetching the new commands. For details on the Write/Read Pointers see **Section 26.5.3.5.3.2**, *MAPLE CRPE-ULF Command Input Buffer Write Pointer <x> Parameter (MCUFCIBWPxP)*, on page 26-365 and **Section 26.5.3.5.3.3**, *MAPLE CRPE-ULF Command Input Buffer Read Pointer <x> Parameter (MCUFCIBRPxP)*, on page 26-366

26.4.3.6.2.11.4 Command Generation Restrictions

The following restrictions apply when issuing commands:

- The host should write a new PCH/Finger Command with a given sub-slot activation index (*SBS*) no later than 2 full sub-slots prior to its activation index, that is, *SBS* ≥ *ULFTSR[SSI]* +3. See example in **Figure 26-146**.
- The maximum pending time for a new command in the external system buffers is 146 sub-slots, that is, *SBS* ≤ *ULFTSR[SSI]*+146. See example in **Figure 26-146**.
- Command activation takes 2 or 3 sub-slots if *ULFGCR[MDS]* = 0 or 1 respectively. A new PCHC/FIC with a certain *PCHID*/*FID* can be issued only after the activation completion of the previous command with the same *PCHID*/*FID*. See example in **Figure 26-146**.

For example, if *ULFGCR[MDS]* = 0 and the command is set with *SBS* = 87, then the host can write the command to system memory no earlier than *ULFTSR[SSI]* = 241 and no later than *ULFTSR[SSI]* = 84. The earliest time the user can issue the next command (to the same PCH or FInger) is in sub-slot 89 with earliest activation time SBS = 92.

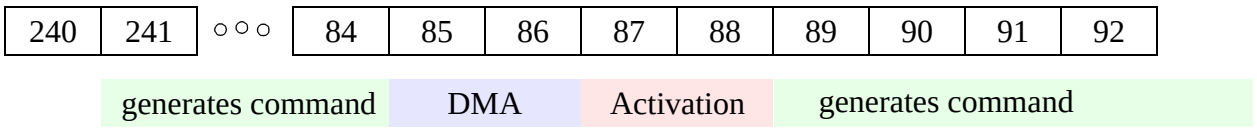


Figure 26-146. Setting a Command with SBS = *87 (ULFGCR[MDS]=0)*

26.4.3.6.2.11.5 Command Setting Rules

Use the following guidelines to issue commands:

- A PCH is active when there is at least one active finger with PCHID equal to the PCH ID. All PCH attributes except ELO can be set while a PCH is active. Fingers can be add to and remove from PCH while it is active.

- When *ULFGCR[MDS]* = 0, the finger belonging to a PCH must comply with maximum delay spread of 256 chips, that is, the difference between the latest finger offset and the earliest finger offset must not exceed 256 chips. In this mode, the maximum finger offset allowed is $255_{\text{(last chip index in current sub-slot)}}+256_{\text{(max delay spread)}} = 511$.
- When MDS = 0, the finger belonging to a PCH must comply with maximum delay spread of 512 chips, that is, the difference between the latest finger offset and the earliest finger offset must not exceed 512 chips. In this mode, the maximum finger offset allowed is $255_{\text{(last chip index in current sub-slot)}}+512_{\text{(max delay spread)}} = 767$.
- A finger is active when its *FIC[AID]* is less then *ULFGCR[NOA]*. All the finger attributes, except for the *FIC[PCHID]*, can be re-configured while the finger is active. To change the finger *FIC[PCHID]*, the finger must be removed and regenerated using a new *FIC[PCHID]*.
- Removing a finger is done by generating a finger command where *FID* is set to the ID of the finger that shall be removed, *AID* is set to *ULFGCR[NOA]*, SBS is set to the sub-slot index in which the finger shall be removed and other parameters (*SID*, *FOF*, *FIN* and *PCHID*) are set to the last value configured by the last finger command for this finger.
- A finger is moved forward when the new *FIC[FOF]* is greater than the old *FIC[FOF]*. The finger is moved backward when the new *FIC[FOF]* is less than the old *FIC[FOF]*. The finger can be moved forward and backward on the fly. There is no limit on moving a finger forward on the fly; however, when moving a finger backward on the fly it can only be moved up to 223 chips.
- To move a finger backward more than 223 chips, remove the finger and then regenerate it with a new offset.

26.4.3.6.2.12 CRPE-ULF Errors

The CRPE-ULF reports the following error types:

- **Command Error**: If a PCH/Finger Command is received and the command *SBS* field indicates that the command is early or late (see **Section 26.4.3.6.2.11.4**, *Command Generation Restrictions*), the *ULFESR[CE]* is set and the command attributes (PCH/Finger Command Type and Command ID) are written to *ULFCFSR*. Upon receiving a Command Error indication, the CRPE-ULF should be reset as described in **Section 26.4.3.6.4**, *CRPE Reset*.
- **Input Buffer Overflow Error**: If the input buffer overflows, the *ULFESR[IBO]* is set and the address of the transaction that caused the error is written to the Input Buffer Status Register (*ULFIBSR[ADDR]*). Upon receiving a Input Buffer Overflow indication, the CRPE-ULF should be reset as described in **Section 26.4.3.6.4**, *CRPE Reset*.

Upon assertion of any of the above events, the MAPLE-B2 asserts the general error interrupt (see **Section 26.4.3.1.2.2**, *General Error Event Interrupt*), indicating a CRPE-ULF error. Indication

as to the error type (Command Error or Input Buffer Overflow Error) is achieved by reading the *ULFESR*.

26.4.3.6.2.13 CRPE-ULF Initialization

The following steps describe the CRPE-ULF initialization (or re-initialized) flow :

1. MAPLE-B2 completes with its API initialization.
2. Host configure CRPE-ULF logic by writing to *ULFGCR* and *ULFSCR*.
3. Host (optionally) configure interpolation logic by writing to *ULFICRx*
4. Host configure at least one output buffers by writing to *ULFOBxBCR* and *ULFOBxACR*.
5. Host configure MAPLE CRPE ULF parameters indicating commands buffers.
6. Host generates MAPLE-B2 PCR Command (`Maple_crpe_ulf_init`) if Interpolation bypass is enabled.
7. Host wait for Antenna I/F synchronization.
8. Host enables the data stream from antenna I/F directly to CRPE-ULF in case of interpolation enabled mode or indirectly in case of interpolation bypass mode.
9. Host generates PCHCs and FICs and write them to system memory.

26.4.3.6.3 Chip Rate Downlink Processing Operation

The MAPLE-B2 supports UMTS (TS 25.211 and TS25.213) Chip Rate processing for Downlink as described in **Section 26.2**, *MAPLE-B2 Features*. These operations are executed using the Chip Rate- Downlink Processing Elements (CRPE-DL).

26.4.3.6.3.1 CRPE-DL Initialization Parameter

During the MAPLE-B2 initialization (see *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)), the CDOMCP parameter in the API must be initialized: See **Section 26.5.2.4**, *CRPE-DL Output Mode Configuration Parameter (CDOMCP)*, on page 26-326 for details.

26.4.3.6.3.2 CRPE Chip-Rate Downlink Processing

The Chip Rate Downlink processing is divided into the following areas of discussion:

- CRPE Downlink operation overview
- Channels Activation and Maintenance
- Input data
- Internal CRPE operation
  - TPC override
  - STTD encoding
  - Spreading operation

- Scrambling operation
- Gain operation
- Physical channels (PCHs) Combine operation
- Beam-forming operation

- Output data
- Initialization and Synchronization

26.4.3.6.3.3 CRPE Downlink Operation Overview

The CRPE Downlink PE operation is described schematically in **Figure 26-147**:

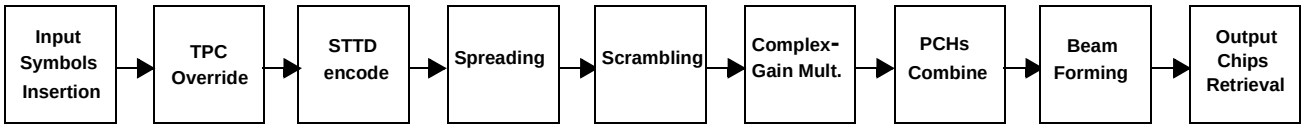


Figure 26-147. CRPE Downlink operation

The CRPE Downlink PE (CRPE-DL) accepts mapped I/Q symbols (optionally STTD encoded, see **Section 26.4.3.6.3.6.4**, *STTD Encoding*, on page 26-274) arranged in 1 or up to 1024 cyclic buffers of physical channels, and performs a set of operations on them:

- TPC override. See **Section 26.4.3.6.3.6.1**, *TPC Override*, on page 26-272.
- STTD encoding. See **Section 26.4.3.6.3.6.4**, *STTD Encoding*, on page 26-274.
- Spreading operation. See **Section 26.4.3.6.3.6.5**, *Spreading Operation*, on page 26-274.
- Scrambling operation. See **Section 26.4.3.6.3.6.6**, *Scrambling Operation*, on page 26-275.
- Multiplication by a complex gain operation. See **Section 26.4.3.6.3.6.7**, *Gain Operation.*, on page 26-275.
- Physical channels (PCHs) Combine operation. See **Section 26.4.3.6.3.6.8**, *Combine Operation*, on page 26-277.
- Beam-forming operation. See **Section 26.4.3.6.3.6.9**, *Beam Forming Operation*, on page 26-278.

The result are output of chips, arranged in a form of 16 antenna buffers, as described under **Section 26.4.3.6.3.7**, *Output Modes Of Operation*, on page 26-280.

26.4.3.6.3.4 Channels Activation and Maintenance

Prior to channel's processing by CRPE-DL, the host is required to program channel's parameters at MAPLE-B2 and CRPE-DL, and fill associated input buffer with input stream symbols of at least a single slot (at least 2 first channel's slots for fast channels).

Out of the 512 supported channels, up to 200 channels can be treated as fast channels. Out of the 200 channels, up to 100 channels are allowed to be active in the same sub-slot. The difference

between fast channels and the rest of the channels is that fast channels activation can be performed at a latency of a single sub-slot before required processing time (see **Figure 26-149**), while the rest of the channels are activated at a latency of at least slot before required processing time (see **Figure 26-148**). Note that all channels (including fast channels) require termination or compressed mode change programming at a latency of at least slot before processing time. All channels can be programmed up to a single frame before actual processing time.

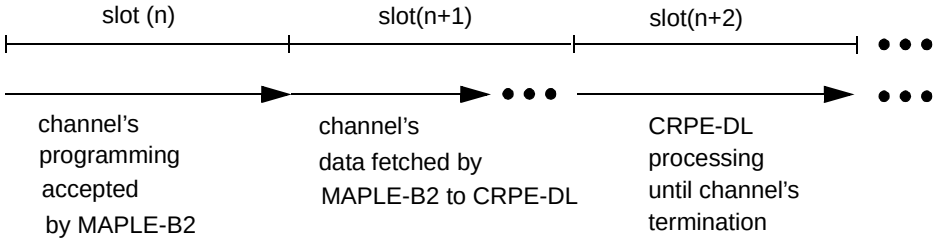


Figure 26-148. Channel Programming Latency Illustration

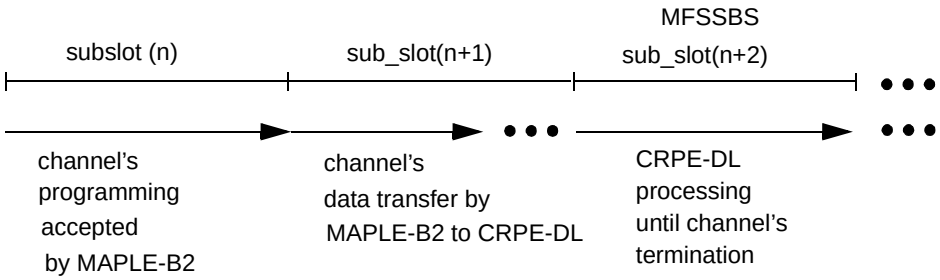


Figure 26-149. Sub-Slot Latency Channel Programming Illustration

For normal channels, MAPLE-B2 fetches channel data that requires processing in the next slot at the current slot time. For fast channels, during the slot of fast channel activation, MAPLE-B2 fetch channel's data that require processing at current slot and next slots.

Channels activation, termination, and compressed status update is programmed using *MCDLFCxP0*, *MCDLFCxP1* and *MCDLFCxP2* parameters for fast channels and *MCDLSCxP0*, *MCDLSCxP1* and *MCDLSCxP2* parameters for the rest of the channels. It is not allowed to activate the same channel using both sets of registers. The *MCDLNOCLP* parameter defines a limit to the maximum number of used slot channels, and a limit to the maximum number of used fast channels. The limit numbers are used as MAPLE-B2's configuration information that is used to reduce MAPLE-B2 processing time by indicating the maximum number of channel's parameters which requires attending.

The processing of a slot channel starts at slot number and sub-slot number as defined by *MCDLSCxP0[MSSL]* and *MCDLSCxP1[MSSBS]*, or *MCDLFCxP0[MFSSL]* and *MCDLFCxP1[MFSSBS]* for fast channels, respectively. The same fields also determine compressed mode status change points.

MCDLSCxP1[MCMPC] and *MCDLFCxP1[MFCMPC]* defines the compressed mode status up until the current slot and sub-slot (but does not include the current slot and sub-slot) for the slot/fast channels, and *MCDLSCxP1[MCMPN]* and *MCDLFCxP1[MFCMPN]* defines the compressed mode status starting from the current slot and sub-slot (including the current slot and sub-slot) for the slot/fast channels.

When *MCDLSCxP1[MCONT]* (or *MCDLFCxP1[MFCONT]* for fast channels) is cleared, the processing of a channel terminates at the slot and sub-slot defined by *MCDLSCxP0[MESL]* and *MCDLSCxP1[MESBS]*, or *MCDLFCxP0[MFESL]* and *MCDLFCxP1[MFESBS]* for fast channels, respectively. Otherwise, channel processing continues from one frame to the next.

Other parameters that are programmed in these registers, apart from system buffer parameters, indicate whether the channel is a synchronization channel (spreading and scrambling bypass) by using *MCDLSCxP1[MSYNC]* (or *MCDLFCxP1[MFSYNC]* for fast channel) and defines the channel spreading factor and spreading factor reduction in compressed mode, using *MCDLSCxP1[MSF]* and *MCDLSCxP1[MSFR]*, or *MCDLFCxP1[MFSF]* and *MCDLFCxP1[MFSFR]* for fast channels, respectively.

When these parameters are updated by the host, the host should set the *MCDLSCxP0[MVALID]* (or *MCDLFCxP0[MFVALID]* for fast channels), indicating to MAPLE-B2 that parameters are updated. Upon channel activation, the host also must clear the *MCDLSCxP0[MSTARTED]* (or *MCDLFCxP0[MFSTARTED]* for fast channel).

Each of the *MCDLSCxP(0–2)* parameters has a dedicated address per channel ID #x. *MCDLFCxP(0–2)*, however, is a joint and limited pool of parameters for all channels where the *MCDLFCxP1[MFCID]* defines to which of the possible 1024 channels (512 channels plus 512 associated external STTD channels) this set of parameters is associated. When a couple of channels are used as STTD pair (channel IDs #x and #(x+512)), the channel of external STTD (channel #x+512) must be programmed at *MCDLFCxP(0-2)* entry that is greater than its coupled channel’s entry.

Figure 26-150 illustrates concatenation of channel termination, channel parameters reprogramming and new channel activation. Since same channel index is used for both old and new parameters, a pipe cleaning period of 2 slots is required from the slot of channel termination by the host, until the slot of new channel activation by the host.

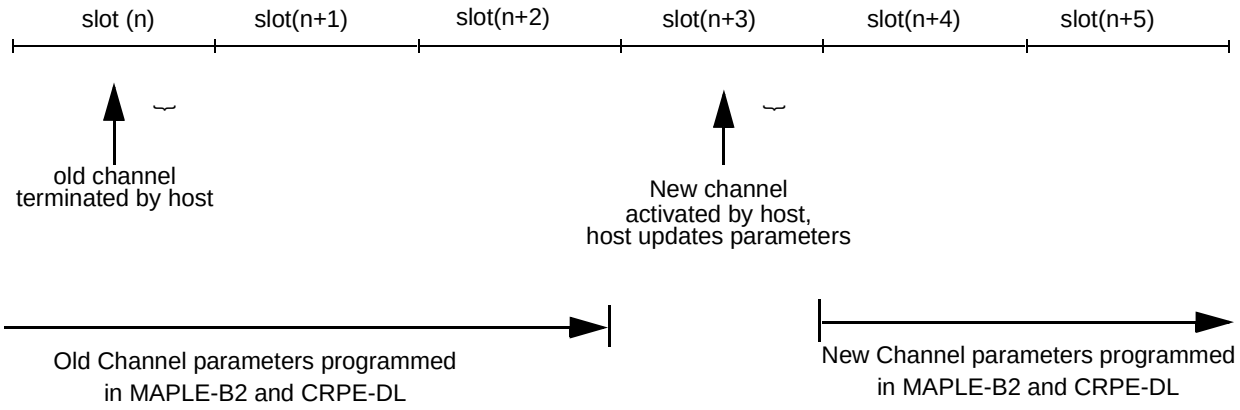


Figure 26-150. Channel Reprogramming Illustration

The following items describe the sequence of actions required for CRPE-DL Physical Channels (PCHs) activation and maintenance:

1. Initial Configuration:
   - Update Scrambling Initialization LUT Memory. See **Section 26.4.3.6.3.6.6**, *Scrambling Operation*
   - Update Slot Format LUT Memory. See **Section 26.5.5.7.2**, *CRPE-DL Slot Format Look-Up Table (SFLUT)*
   - Update CRPE-DL Beam Forming Coefficients Values Control Registers (*CDBFCVLRx*).
   - Update CRPE-DL Beam Forming Coefficients Timing Command Control Registers (*CDBFCTCRLx*).
   - Activate CRPE-DL by writing to CRPE-DL Start Control Register (*CDSLR*)
2. Channel Activation and Termination:
   - Update CRPE-DL General Command Control Register (*CDGCLR*).
   - Activate channel at MAPLE-B2's CRPE-DL Slot/Fast Channel Parameters (*MCDLSCxP(0-2)/MCDLFCxP(0-2)*)
   - Update compressed status change in the Slot/Fast Channel Parameters (*MCDLSCxP(0-2)/MCDLFCxP(0-2)*) upon every change of compressed mode.
   - Terminate channel at in the Slot/Fast Channel Parameters (*MCDLSCxP(0-2)/MCDLFCxP(0-2)*).
3. Channel Maintenance:
   - Update CRPE-DL TPC Command Control Register (*CDTCLR*). It can be updated up to once per slot.
   - Update CRPE-DL Virtual Antenna Gains Control Registers (*CDVAGLR(0-1)*). It can be updated up to once per slot.
   - Update CRPE-DL Virtual Antennas Gain Command Control Register (*CDVAGCLR*). It can be updated up to once per slot.

— Update CRPE-DL Idle Period Control Registers (*CDIPLRx*). It can be updated once per idle period per virtual antenna.
— Update CRPE-DL Beam Forming Coefficients Values Control Registers (*CDBFCVLRx*). This operation is optional
— Update CRPE-DL Beam Forming Coefficients Timing Command Control Register (*CDBFCTCRLx*). This operation is optional.

26.4.3.6.3.5 CRPE-DL Input Data Structure

The number of supported input streams is up to 1024, each potentially sourced from a different system buffer. An input stream is associated with a single Physical Channel (PCH). If STTD operation is performed externally, 2 input streams are associated with the same PCH. Channels indexes 512–1023 are reserved to input streams of external STTD operation. In the case of external STTD operation, channel #x and channel #x+512 (x=0,1...,511) are coupled, where channel #x+512 is the external STTD channel.

Figure 26-151 illustrates the CRPE-DL supported input data structure:

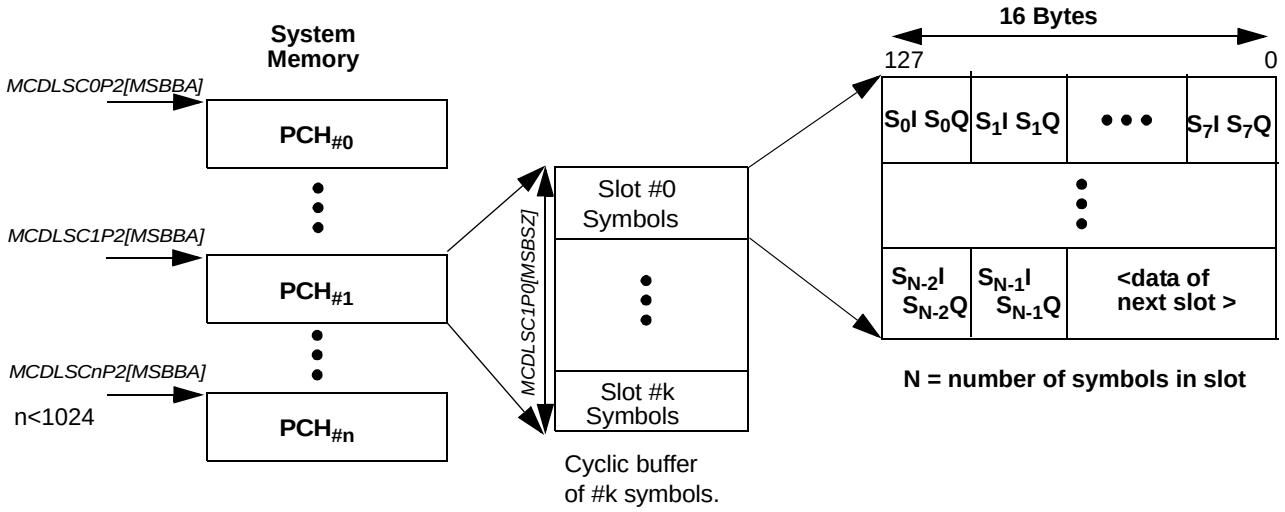


Figure 26-151. CRPE-DL input buffer

Each cyclic buffers contains complex symbols (8 bits I and 8 bits Q pair), 8 complex symbols per 16 bytes address, as illustrated in **Figure 26-151**.

Each buffer contains a number of slots, and each slot contains a number of bytes, depending on the Spreading Factor of the $PCH_{\#x}$, as detailed in **Table 26-122**. The total number of bytes per buffer is defined by *MCDLSCxP0[MSBSZ]* (or *MCDLFCxP0[MFSBSZ]* for fast channels). The base address of the buffer is defined by *MCDLSCxP2[MSBBA]* (or *MCDLFCxP2[MFSBBA]* for fast channels).

Table 26-122. Slots Size with respect to Spreading Factor of $PCH_{\#x}$

SF of PCHx value	Slot Size (bytes)
4	1280
8	640
16	320
32	160
64	80
128	40
256	20

Slots that belong to compressed frame with Spreading Factor (SF) reduction contain the number of bytes as indicated by **Table 26-122** according to the reduced SF. Zero valued complex symbols in the buffer represent symbols that need not be transmitted due to compressed mode. In addition, synchronization channel slots are supported with 256 complex symbols per slot at SF = 1 (512 bytes per slot).

The base address of each input buffer at 256 bytes unit is defined by *MCDLSCxP2[MSBBA]* (or *MCDLFCxP2[MFSBBA]* for fast channel), while the size at 2 bytes unit is defined by *MCDLSCxP0[MSBSZ]* (or *MCDLFCxP0[MFSBSZ]* for fast channel). The slots in each buffer are placed in subsequent addresses. All symbols in the buffer are sequential. First symbol of slot #k is placed in the address that immediately follows last symbol of slot #k-1.

Each input buffer is associated with internal read-pointer: *MCDLSCxRPP* (or *MCDLFCxRPP*, for fast channel) which is updated by MAPLE-B2, indicating the last address that have been fetched by the MAPLE-B2 and can be used by the host to monitor CRPE-DL processing progress and to prevent buffer overruns while adding new data into the buffer. Typically, the host is synchronized to global ‘frame’, ‘slot’ and ‘sub-slot’ timing of the system and is not required to monitor read-pointer to avoid buffer overrun or under-run.

26.4.3.6.3.6 Internal CRPE Operation

This section describes the required details on how to setup a PCH to perform the various CRPE-DL operations.

26.4.3.6.3.6.1 TPC Override

The CRPE-DL is capable of overriding DPCCH TPC fields with updated TPC input symbol generated by the host prior to slot transmission. The following section describes the process details.

26.4.3.6.3.6.2 Slot Format Look Up Table

Before setting up a PCH, the host must create/update the entry in the *Slot Format LUT memory* to which the PCH is associated with. This entry is used for the TPC commands, as specified in **Section 26.4.3.6.3.6.3**, *TPC Value Override*. Each Slot Format LUT entry contains information

that defines the distribution of input symbols to up to five different fields in a slot, and may identify one of the fields as TPC. **Section 26.5.5.7.2**, *CRPE-DL Slot Format Look-Up Table (SFLUT)* describes the SFLUT information.

The settings listed in the SFLUT are constant throughout the entire usage of the PCH. Changing an entry is only possible after tearing down the PCHs associated with it. The Slot Format LUT memory allows up to 90 concurrent different slot formats.

26.4.3.6.3.6.3 TPC Value Override

During CRPE-DL operation, TPC override of a PCH#x and it STTD pair (PCH#x+512) is done by direct accessing to the CRPE-DL TCP Command Control Register (*CDTCLR*). TPC override is turned on by setting *CDTCLR[TPCOV]*. The updated TPC complex symbol is defined by the *CDTCLR[TPCSI]* and *CDTCLR[TPCSQ]* fields. This symbol may override correlating input symbols. The TPC field may contain more than a single symbol, but all TPC symbols of TPC field in a single slot have the same value. The overriding symbol value assumes no STTD operation. CRPE-DL manipulates the overriding symbol value in case an STTD operation affects the input stream. In this case, CRPE-DL changes the overriding TPC value to match post STTD operation prior to overriding the associated input buffer symbols.

The TPC override programming event is allowed to occur anytime during the PCH#x and PCH#x+512 transmission period. Its value takes effect at the following event of processing first symbol of TPC field by CRPE-DL.

Figure 26-152 illustrates a 5 field slot in which the second field is a TPC field. TPC update takes affect for the current slot only if it arrives before processing any TPC symbol of the PCH at the current slot.

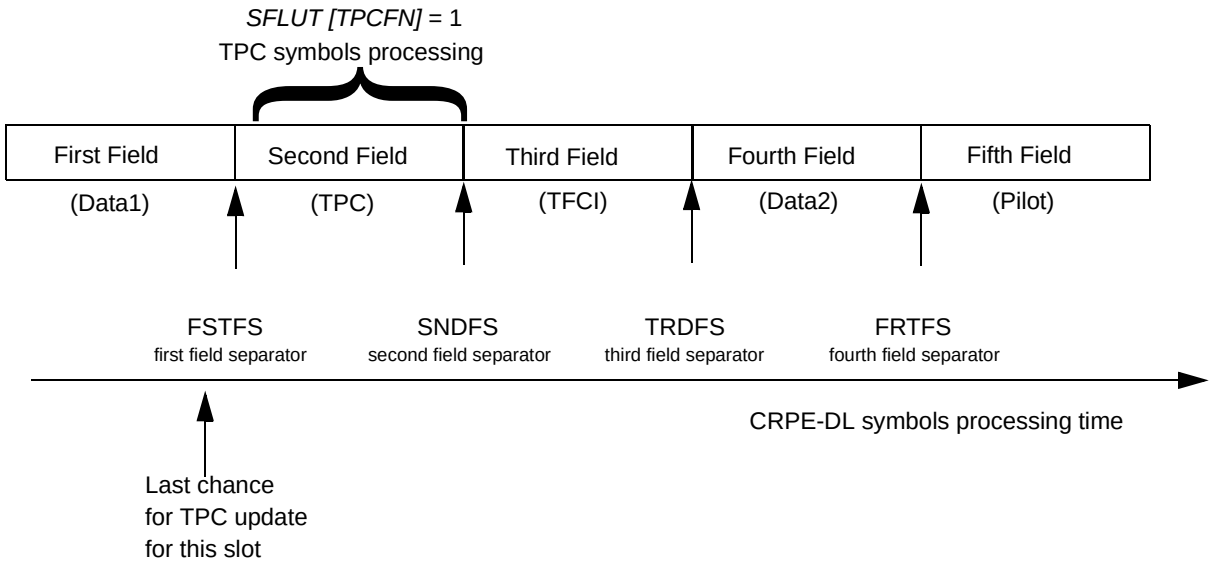


Figure 26-152. TPC update timing

A TPC update command that occurs at least 16 chips (about 4.17 µs) prior to the start of TPC field processing, affects current slot. TPC commands that occur 15 or less chips prior to the start of the TPC field processing may affect the current or the next slot.

26.4.3.6.3.6.4 STTD Encoding

When the *CDGCLR[STTD]* field is set with the value of 2, indicating internal STTD calculations, the CRPE-DL performs STTD operation internally on relevant channel. The input stream involved with such a channel is processed twice. First time data is processed without STTD operation and targeted to virtual antenna as pointed by *CDGCLR[VANTA]*. Second time data is encoded as illustrated by **Figure 26-153**, processed and targeted to virtual antenna as pointed by *CDGCLR[VANTAB]*. When working with internal STTD enable (*CDGCLR[STTD]*=2) the *CDGCLR[VANTABEN]* value should be 1.Otherwise CRPE-DL does not perform STTD operation on the associated channel.

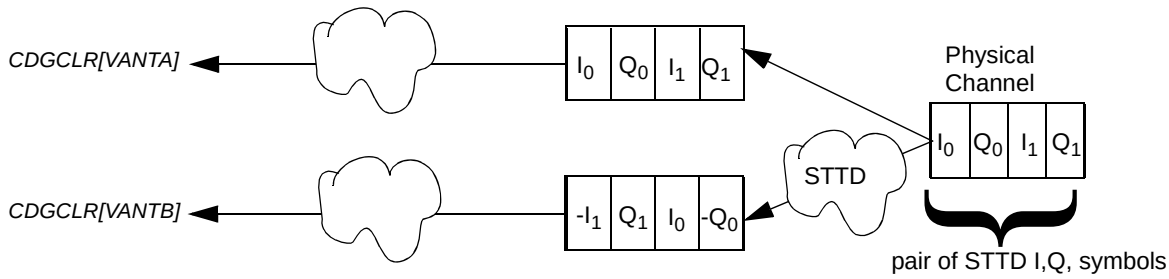


Figure 26-153. CRPE-DL STTD Diversity Operation

CRPE-DL can perform STTD encoding only for channels for which uniform operation for all symbols of the slot (as illustrated in **Figure 26-153**) is suitable for the channel. In addition, CRPE-DL does not support internal STTD operation for channels with a Spreading Factor of 256.

Channels which require STTD encoding which is not supported by CRPE-DL should be encoded externally. In this case, two input streams must be assigned as two separate CRPE-DL PCHs: one original assigned to channel #x (x=0,1,...,511), and a second coupled external STTD with diversity assigned to channel #x+512. The *CDGCLR[STTD]* field associated with the STTD diversity stream that was encoded externally should get the value 3.

26.4.3.6.3.6.5 Spreading Operation

CRPE-DL performs channelization as specified in 3GPP TS version 9.9.0 (release 9) 25.213 standard section 5.1.2.

I and Q branches of the physical channel (except in case of synchronization channel, identified by setting *CDGCLR[SCH]*) are separated to chip rate by the same real-valued channelization code. The channelization code of compressed/non-compressed frames is defined by the

CDGCLR[OVSF]/CDGCLR[CMOVSF] fields respectively, and by the *MCDLSCxP1[MSF]* field (or *MCDLFCxP1[MFSF]* for fast channels).

The channelization code pattern is generated internally by CRPE-DL, in accordance with the standard. **Figure 26-154** illustrates the repeating channelization code patterns $C_{ch,MSF,OVSF}$ of several selected *OVSF* and *MSF* values. Note that Spreading Factor value must always be greater than correlating OVSF value.

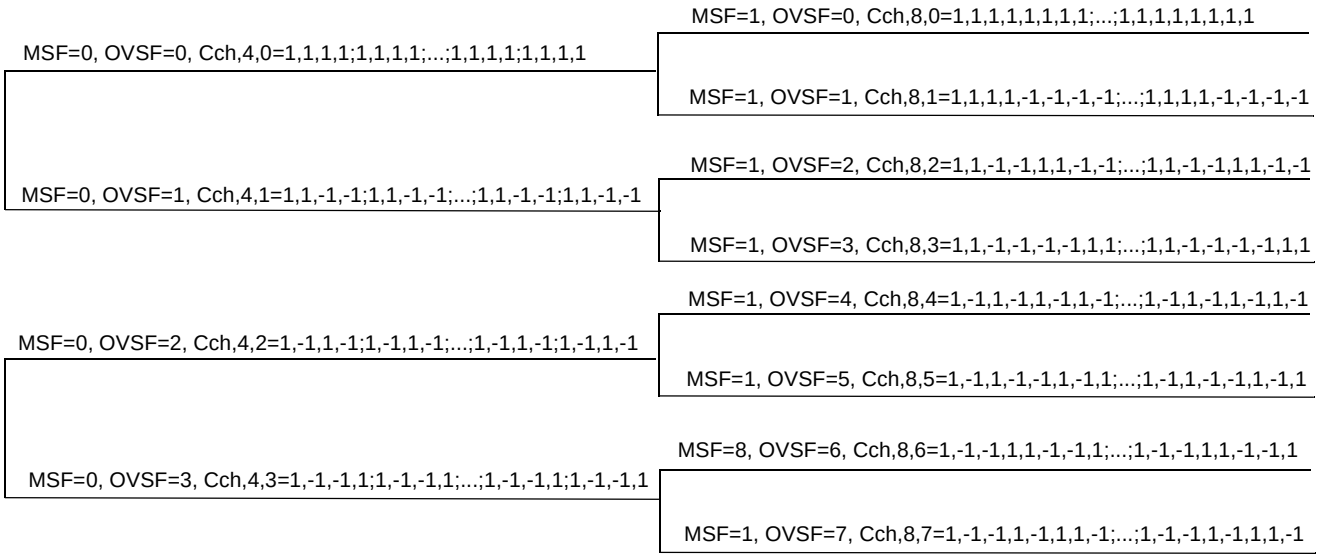


Figure 26-154. Channelization Code Patterns for MSF = 4,8 Illustration

26.4.3.6.3.6.6 Scrambling Operation

The CRPE-DL supports up to 32 different simultaneous scrambling codes, associated with non compressed frames. Scrambling is performed according to 3GPP TS 25.213 section 5.2.2. The CRPE-DL is generating the spreading code internally using an internal Scrambling Initialization Look-Up Table (*SCRILUT*). Each entry in the *SCRILUT* contains the parameter SN_x. This parameter, 13 bits wide, selects a single scrambling code, ranging from 0 to 8191 and associated with non compressed frames. Each PCH contains a pointer to one of the 32 possible entries in the LUT, identified by the *CDGCLR[SCRIX]* field. In addition, the *CDGCLR[CMSCR]* field selects the scrambling code of compressed frames to be either the same as the scrambling code of non compressed frames, or associated left or right alternative scrambling code.

The *SCRILUT* description is detailed in **Section 26.5.5.7.3**, *CRPE-DL Scrambling Initialization Look-Up Table (SCRILUT)* and should be initialized during CRPE-DL initial configuration phase.

26.4.3.6.3.6.7 Gain Operation.

Performing Gain operation and Phase shift adjustments to the output chips is done by controlling each PCH parameters as follows:

Each field out of five possible fields in a slot, as defined by Slot Format LUT (see **Figure 26-152**), is associated with a single 16 bit real gain factor out of four possible real gain factors per channel. The *SFLUT[FSTGF]* (in the Slot Format LUT) associates the first, second, third or forth real gain factor with first field; while *SFLUT[SNDGF]*, *SFLUT[TRDGF]*, *SFLUT[FRTGF]* and *SFLUT[FFTGF]* associate real gain factors with second, third, fourth and fifth fields respectively.

The real gain factors are 2 bytes signed real numbers. In addition to the real gain factors, *CDVAGLR1[IGA]* and *CDVAGLR1[QGA]* represent a complex factor for antenna A; while *CDVAGLR1[IGB]* and *CDVAGLR1[QGB]* represent a complex factor for antenna B. The complex factors are 4 bits each, 2 bits are assigned for the real part of the complex factor and 2 bits are assigned for the imaginary part of the complex factor. Each 2 bits may represent signed values from the range of 1, 0 or –1. The real gain factors along with the complex factors are used for gain and phase shift of each channel separately.

The values of the four real factors of a channel are associated with both virtual antennas A and B, and programmed at the *CDVAGLR0* register fields per PCH. Complex factors are programmed at *CDVAGLR1* registers (also per PCH). Each PCH has a dedicated registers' address space for the programming of its gains.

Gains Usage synchronization

The *CDVAGLR(0-1)* registers, containing the actual gain factors per channel, are treated as shadow registers and do not necessarily become effective immediately. The *CDVAGCLR[GUS]* and *CDVAGCLR[GUN]* fields, as programmed in the CRPE-DL Virtual Antennas Gain Command Control Register, point to the slot from which programmed gains become effective. The *CDVAGCLR[GUS]* points to a specific slot, while *CDVAGCLR[GUN]* indicates that new gains should be used at the earliest slot in the channel frame.

CRPE-DL uses previously programmed gain factors for all slots preceding the slot pointed by *CDVAGCLR[GUS]* or *CDVAGCLR[GUN]*. For the programming of *CDVAGCLR[GUS]* value, the user need not take into account channel sub-slot offset. The actual update is performed automatically when processing the first symbol of a field pointed by *SFLUT[GUF]* at the Slot Format LUT. Gain is also updated when the first symbol of a new channel is processed.

If gain weights are re-programmed before becoming effective, the overridden weights are lost and never used. If *CDVAGCLR[GUS]* or *CDVAGCLR[GUN]* are re-programmed before CRPE-DL reaches the slot pointed by them, the overridden *CDVAGCLR[GUS]* or *CDVAGCLR[GUN]* are lost and never used. **Figure 26-155** illustrates the usage of updated gains.

Gain update latency is allowed to be as low as 16 chips chunk before the affected slot processing starts. An update of gain weights values, indicating the required usage of a new gain, must occur before CRPE-DL started processing the first IQ symbol of the slot pointed by *CDVAGCLR[GUS]*.

The processing state of CRPE-DL, indicating progress of chips processing, is accessible to the system by reading CRPE-DL Processing State Status Register. *CDPSSR[FI]*, *CDPSSR[SI]*, *CDPSSR[BI]* and *CDPSSR[CI]* indicate the index of currently processed frame, slot, sub-slot and chunk respectively.

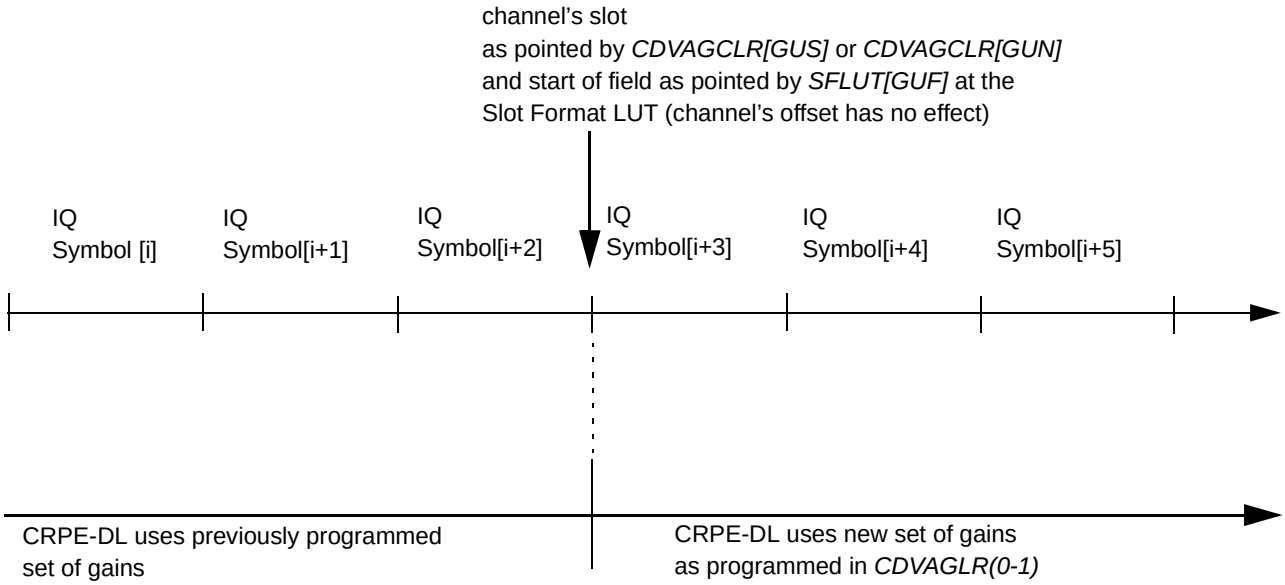


Figure 26-155. Usage Of Updated Gains Illustration

26.4.3.6.3.6.8 Combine Operation

CRPE-DL processes 16-chip chunks from all active channels (one channel after the other) and accumulates the channels chips at the combine stage. Sixteen chips per virtual antenna are accumulated at the combine stage. A complex chip in the combine stage is represented by real and imaginary parts of 32 bits each. Chip values are limited between the upper and lower saturation values. Each virtual antenna has its own entry in the combine stage.

After accumulating the chips at the combine stage, the representation of the chip is reduced from 64 bits (32 real and 32 imaginary) to 32 bits (16 real and 16 imaginary). The *CDCCSCLRx[CCSNUM]* field determines the number of shift right operations to perform for the combined chips. The *CDCCSCLRx[CCSNUM]* is a shadow parameter that takes effect at the slot and sub-slot indicated by *CDCCSCLRx[CCSUSL]* and *CDCCSCLRx[CCSUSS]*, respectively, or at the next earliest sub-slot if *CDCCSCLRx[CCSUN]* is set to 1.

Each channel is targeted to up to 2 virtual antennas out of 16 virtual antennas, indicated by *VANTA* and *VANTB* as programmed in the *CDGCLR* register. The *CDGCLR[VANTABEN]* field, when set, enables virtual antenna B in addition to virtual antenna A. Otherwise, the processed chips of the channel are targeted only to virtual antenna A. When *CDGCLR[STTD]* equals 2, the channel should be directed to both antennas, while the STTD diversity stream, which is generated internally by CRPE-DL, is targeted to virtual antenna B.

26.4.3.6.3.6.9 Beam Forming Operation

The CRPE-DL is capable of performing beam forming operations. The output of beam forming operations are 16 beam output antennas. Each beam antenna is a unique complex linear combination of 4 virtual antennas associated with the beam antenna.

Each beam is formed independently. The virtual antennas are separated to 4 groups of 4 virtual antennas, and the beam antennas are also separated to 4 groups of 4 beam antennas. The groups are coupled to pairs such that each group of virtual antennas forms beams at a single group of beam antennas. **Figure 26-156** illustrates the correlation between virtual antennas and beam antennas. The figure illustrates 4 groups of 4 virtual antennas each group, generating 4 groups of 4 beams. The following objects are shown:

- VA_i represents 4 bytes chips of virtual antennas i at the combine stage.
- BF_{mn} is a complex coefficient, used for the multiplication of virtual antenna m while forming beam antenna n.
- BA_i represents 4 bytes chip of beam antenna i at the CRPE-DL output buffer.

The values of the complex coefficients BF_{mn} are programmed in the CRPE-DL Beam Forming Control Registers (*CDBFCVLR(0-7)*) using the following equation:

$$BF_{mn} = (BF_{mn}I) + j \times (BF_{mn}Q)$$

The BF_{mn} coefficients are written to shadow registers. The latest programmed coefficients take effect starting from the slot and sub-slot as specified in *CDBFCTCLRx[BFUSL]* and *CDBFCTCLRx[BFUSS]* fields respectively. *CDBFCTCLRx[BFUN]* may be used instead of *CDBFCTCLRx[BFUSL]* and *CDBFCTCLRx[BFUSS]*, indicating that coefficients should take effect at the next earliest sub-slot.

The *CDBFCTCLR0* register correlates with *CDBFCVLR(0–1)* registers, *CDBFCTCLR1* register correlates with *CDBFCVLR(2–3)* registers, *CDBFCTCLR2* register correlates with *CDBFCVLR(4–5)* registers and *CDBFCTCLR3* register correlates with *CDBFCVLR(4–5)* registers.

The most basic case of beam forming is the one at which each virtual antenna is directly mapped to the correlating output antenna (virtual antenna *n* to output antenna *n*, where *n* = 0 to 15). The beam forming coefficients need to be programmed for the basic case as well as more complex case prior to CRPE-DL operation.

After beam-forming operation, the chips are internally represented by 54 bits (27 real and 27 imaginary). Final output chips are reduced to representation of 32 bits (16 real and 16 imaginary). The *BFSNUM* field in *CDBFCTCLR(0-3)* determines the number of shift right operations to perform for the chips after beam forming operation. The *BFSNUM* is a shadow field that takes affect at slot and sub-slot as pointed by *CDBFCTCLRx[BFSUSL]* and *CDBFCTCLRx[BFSUSS]* respectively, or at the next earliest sub-slot if *CDBFCTCLRx[BFSUN]* is set to 1.

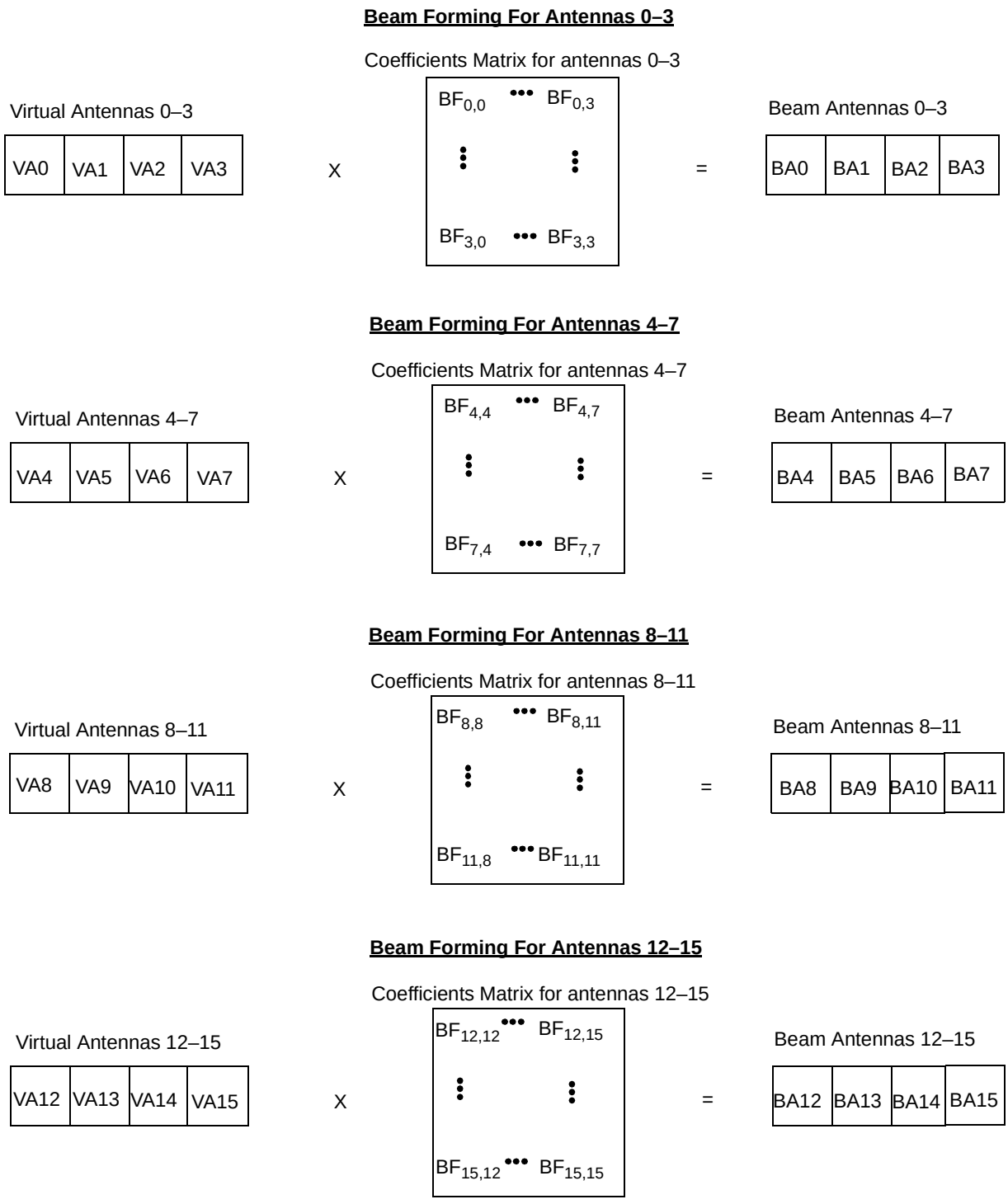


Figure 26-156. Virtual Antennas And Beam Antennas Correlation Illustration

26.4.3.6.3.6.10 Virtual Antennas Idle Period

Each virtual antenna is associated with its own Idle Period Control Register (*CDIPLR*). This register defines periods of time in which all coefficients associated with the virtual antenna are considered as zero, regardless their actual values in *CDBFCVLR(0-7)* registers.

The first slot and sub-slot of the idle period of virtual antenna i are defined by *CDIPLR[IPSSL$_i$]* and *CDIPLR[IPSSBS$_i$]* respectively. Start time is included in the idle period.

The last slot and sub-slot of the idle period of virtual antenna i are defined by *CDIPLR[IPESL$_i$]* and *CDIPLR[IPESBS$_i$]* respectively. End time is not included in the idle period.

The programming of *CDIPLR* register takes affect for a single idle period. When programming the register, the host should set *IPV$_i$* bit and CRPE-DL clears the bit once an idle period is performed from start to end.

26.4.3.6.3.7 Output Modes Of Operation

The MAPLE-B2 supports the following two operation modes with respect to CRPE-DL outputs:

- CPRI output mode in which output stream is accessible at the output buffer of the CRPE-DL through the MBus slave port for CPRI direct access.
- MAPLE-B2 output mode in which MAPLE-B2 actively transfers output stream to external system buffers.

During MAPLE-B2 output mode, MAPLE-B2 transfers data from CRPE-DL output buffer to system buffers each time a new processing chunk (16 chips per antenna) is available. In CPRI output mode data is fetched from CRPE-DL output buffer by an external master (CPRI) independently.

26.4.3.6.3.7.1 MAPLE-B2 Output Mode

MAPLE-B2 output mode is selected when *CDOMCP[MOM]* field in the CRPE-DL Output Mode Parameter is set. The *MCDLOBxBAP* parameters along with *MCDLOBxSP* parameters define the system output cyclic buffers to which MAPLE-B2 transfers the output chips. The *MCDLOBxBAP[MOBBA]* field defines the base address of the antenna's output system buffer at granularity of 128 bytes. The *MCDLOBxSP[MOBSZ]* field defines the size of the antenna's output system buffer at granularity of 128 bytes. The MAPLE-B2 transfers 256 chips per antenna at time period of a single sub-slot.

The MAPLE-B2 maintains an output buffer write pointer for each output buffer in the *MOWR_PTR* field of the *MCDLOBxWPP* parameter. Each antenna is associated with its own dedicated write pointer parameter. This parameter can be monitored to verify MAPLE-B2 transfer progress; however it is not required to do so, because it can be assumed that MAPLE-B2 transfer rate is sufficient for real time requirements. The write pointer parameter can also be used to identify the completion of first data chunk transfer after system initialization.

The system output buffers are filled by MAPLE-B2 cyclically; when MAPLE-B2 reaches the end of a buffer, it wraps back to the base address of the buffer. **Figure 26-157** illustrates output system buffer example of antenna #0.

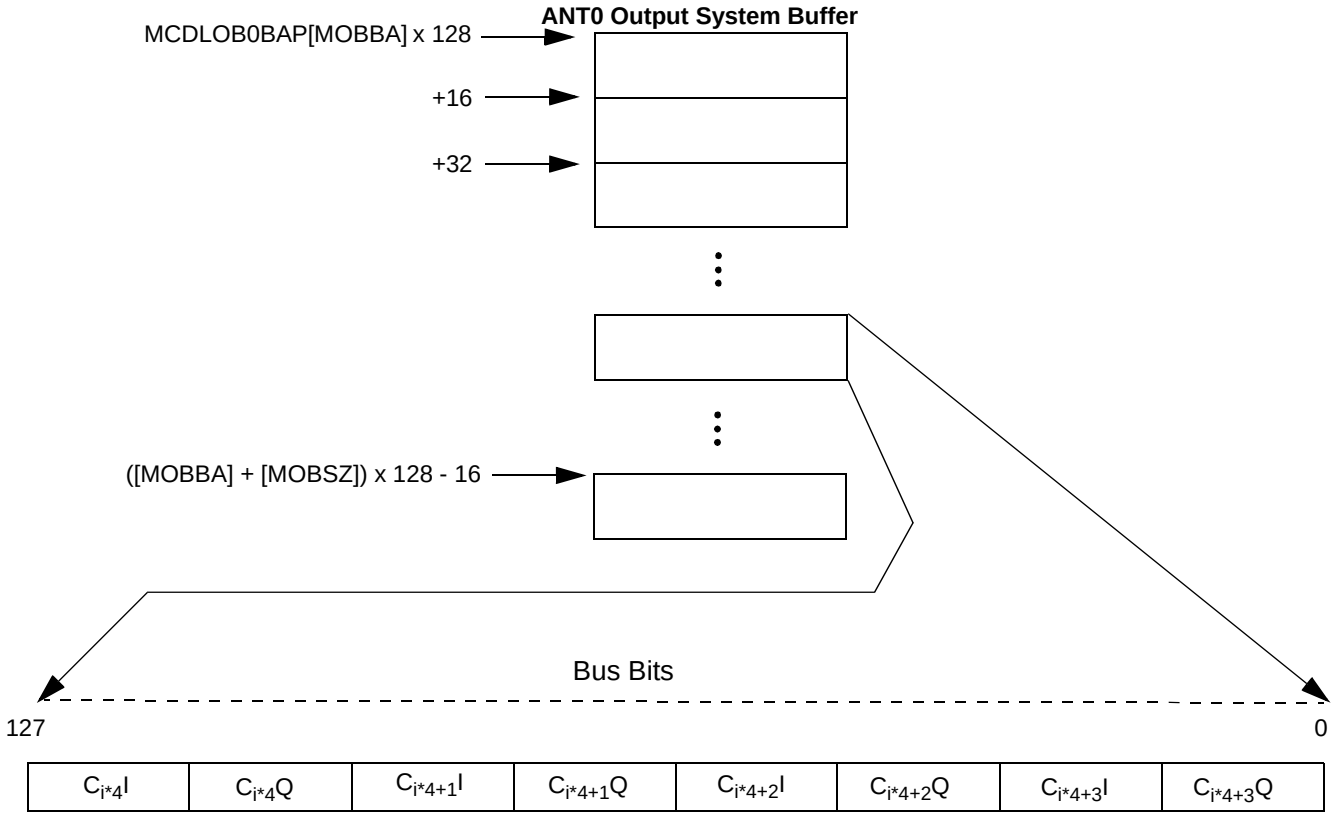


Figure 26-157. Output System Buffers Illustration

26.4.3.6.3.7.2 CPRI Output Mode

CPRI output mode is selected when the *CDOMCP[MOM]* field is cleared. In this mode, MAPLE-B2 expects that CRPE-DL output buffer is read directly by an external host such as CPRI DMA. The data structure of the CRPE-DL output buffer is illustrated at **Figure 26-158** and is detailed in **Section 26.5.5.7.1**, *CRPE-DL Chips Output Data Table (CDCODT)*

The CRPE-DL output buffer contains 2 entries per antenna. Each entry contains 16 chips data of a single antenna. An output chip is a complex number, represented by 2 bytes of real part and 2 bytes of imaginary part.

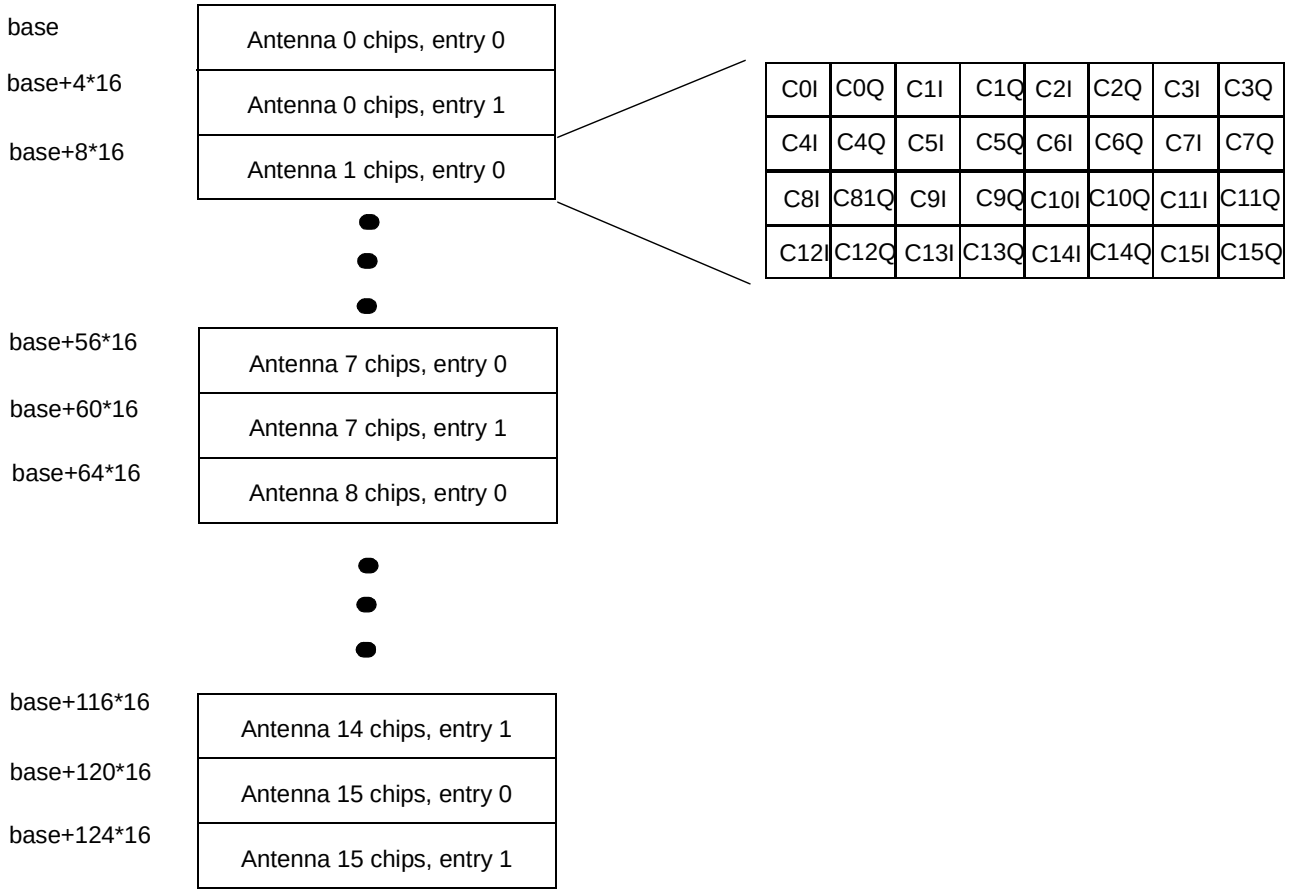


Figure 26-158. CRPE-DL Output Buffer Entry Data Organization Illustration

26.4.3.6.3.7.3 Output Buffer Accesses Constraints

The following is a list of required assumptions and constraints needed to maintain proper use of the CRPE-DL output buffer:

- For Output Buffer Overrun Prevention the following must be maintained:
  - Accesses to CRPE-DL output buffer are monitored for the purpose of CRPE-DL rate control and overrun prevention. Chips of all enabled antennas at current buffer entry should be read before chips from next entry are accessed.
  - Enabling *A0EN* to *A15EN* bits of the *CDOMCP* parameter enables the activation of output antennas 0 to 15 respectively.
  - Every chip should be read a single time.
  - The size of any CRPE-DL output buffer access is a multiple of 16 bytes.
  - Addresses of all CRPE-DL accesses are aligned to 16 bytes.
  - Output buffer reads are at a rate of 3.84 Mega chips per second per antenna. Since each buffer entry includes 16 chips per antenna, chips of all activated antenna must be read from buffer entry at time period of no longer than 4.166 µs.

- For Output Buffer Under-Run Prevention, the following must be maintained:
  - The 2 buffer entries shell accessed in a cyclic manner: read all chips of enabled antennas from first entry, then read all chips of enabled antennas from second entry, then read all chips of enabled antennas from first entry an so on.
  - Output buffer reads are at a rate of 3.84 Mega chips per second per antenna. Since each buffer entry includes 16 chips per antenna, an attempt to accesses more than one buffer entry at the same 4.166 µs time period by an external master is forbidden. During MAPLE-B2 Output Mode the rate of buffer entry change is triggered by an internal CRPE-DL event; in this case there is no restriction of 4.166 µs period.

26.4.3.6.3.8 Initialization of CPRI Output Mode

The following describes the initialization sequence of the CRPE-DL assuming CPRI Output Mode:

1. Host activates CPRI module.
2. CPRI start synchronization period:
   a. Providing frame and sub-slot synchronization events periodically.
   b. Sending zeros to antenna without reading CRPE-DL output buffer.
3. Host writes channels' symbols to system buffers.
4. Host starts polling CPRI synchronization status, until CPRI is synchronized.
5. Host initializes channels at MAPLE-B2 parameters and CRPE-DL registers:
   a. Initialization of LUT Memories.
   b. Initialization of beam forming coefficients.
   c. Initialization of channel activation and parameters.
   d. Initialization of frame, slot and sub-slot counters by performing a write access to CRPE-DL Start Control Register (*CDSLR*).
6. Host start polling the *CDESR[OBS]* status indication, waiting for completion of processing of two chunks.
7. CRPE-DL starts processing when receiving new sub-slot event that follows *CDSLR* write. CRPE-DL fills first 2 chunks of all antennas at output buffer.
8. CRPE-DL sets Output Buffer Status event (*CDESR[OBS]*) to indicate that first 2 chunks of CRPE-DL output buffer are ready to be fetched since *CDSLR* was written.
9. Host can configures CPRI to start fetching data.
10. CPRI fetches first chunk of CRPE output data and then second chunk of output data. CPRI continues to fetch CRPE output chunks at steady state synchronized to the frame start.
11. CRPE-DL continue to fill output buffer at steady state. CRPE-DL monitors CPRI accesses to output buffer to prevents internal output buffer data override.

26.4.3.6.3.9 CRPE-DL Rate Control

CRPE-DL is designed to work at a rate that is faster than real-time requirements. The actual processing rate of CRPE-DL, therefore, requires external control. CRPE-DL rate control is based on two mechanisms: output chip override prevention and controlled processing task per processing period.

26.4.3.6.3.9.1 Output Chip Override Prevention

CRPE-DL writes its output chips to an internal output buffer. At CPRI output mode (*CDOMCP[MOM]*=0) the chips from the output buffer are directly read by an external host such as CPRI, while at MAPLE-B2 output mode (*CDOMCP[MOM]*=1) chips are transferred by MAPLE-B2 to a system buffer.

CRPE-DL monitors the read accesses from its internal output buffer, and prevents overrun of chips that are still not read from the buffer. If a risk of chips overrun at CRPE-DL output buffer occurs, the CRPE-DL halts its processing until relevant chips are read from the output buffer.

The output buffer is a 16 chips double buffer per antenna. At a steady state, CRPE-DL writes chips to one buffer per antenna while chips are read from the other buffer.

26.4.3.6.3.9.2 Controlled Processing Task Per Processing Period

CRPE-DL performs a defined task according to the selected processing period. The processing period can be a 16 chips chunk period or a 256 chips sub-slot period according to the configuration of the *RSEL* field in the *CDOMCP* parameter. Once the defined task processing is complete, CRPE-DL automatically halts its operation and waits for an external event, indicating that a new processing period has just started.

The external event source is selected to be an interrupt if *CDOMCP[RCS]* = 1, or write access to the CDRLR register if *CDOMCP[RCS]* = 0. In the latter case, writing 1 to *CDRLR[CSE]* is used to represent new chunk event, while writing 1 to *CDRLR[BSE]* is used to represent new sub-slot. *CDRLR* is cleared automatically by the CRPE-DL.

CRPE-DL uses new sub-slot event to begin its initial operation after *CDSLR* is written.
CRPE-DL uses new sub-slot event to start execution of a new sub-slot if *CDOMCP[RSEL]* = 1.

CRPE-DL uses a new chunk event to start execution of a new 16 chips if *CDOMCP[RSEL]* = 0.

26.4.3.6.4 CRPE Reset

The MAPLE-B2 supports internal reset assertion for the CRPE modules only (CRPE-ULF, CRPE-DL and CRPE-ULB). Such internal reset assertion may be required in the case the Antenna I/F (CPRI) and the CRPE modules loose synchronization hence should be reset. Reset assertion for the CRPE modules is done by initiating the `Maple_reset_crpe` routine as

described in **Section 26.4.5**, *MAPLE-B2 Internal Task Control*. Once initiated, the MAPLE-B2 performs the following:

1. Set the *MCRRCIP[CR_RST]* (see **Section 26.5.3.5.1.1**) bit indicating that the CRPE reset routine is initiated.
2. Disable all CRPE modules internal interrupts and related routines.
3. Wait to the completion of all running CRPE modules related routines which were initiated prior to reset routine assertion.
4. Assert the internal CRPE hard reset input.
5. Start with CRPE configuration initialization as described in the API parameters.
6. Reset the *MCRRCIP[CR_RST]* bit indicating the internal CRPE reset is completed.

Once internal reset is completed, each of the CRPE modules should be initialized as described in each module's initialization explanation. Note that the initialization part which is executed in the API during MAPLE-B2 initialization is already executed during the `Maple_reset_crpe` routine as described above.

26.4.3.7 CRC Operation

The MAPLE-B2 supports CRC checks and CRC calculations for block sizes of up to 16 KBytes and with eight optional CRC polynomials. These operation are executed using internal CRCPE instantiated in PSIF2.

Note: The CRC processing executed by the CRC Processing Elements is not standard related and, therefore, is not affected by the MAPLE-B2 operation mode.

26.4.3.7.1 CRCPE Buffer Descriptor Structure

A detailed description of the CRC Buffer Descriptor structure expected by the host for the CRCPE BD rings is provided in **Section 26.5.4.7**, *CRCPE Buffer Descriptor Structure*, on page 26-481.

26.4.3.7.2 CRC Input Data Structure

After parsing the CRC Buffer Descriptor, MAPLE-B2 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. The CRC input buffer size must be specified in the [CRC_BS] field of the CRC

BD. **Figure 26-159** shows an example of a 1224 bits block and how it should be placed in the system memory to be input for the MAPLE-B2 CRC processing:

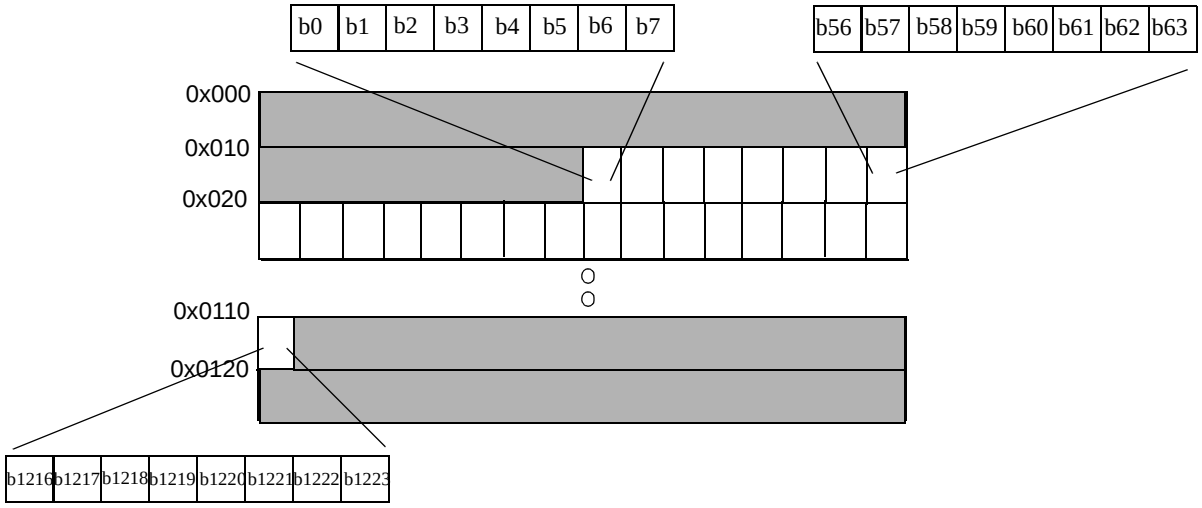


Figure 26-159. Example of Placing a Block of 1224 Bits in the System Memory for CRC Calculation

If CRC calculations are required to be executed on blocks which are not 8 bits aligned, zero padding bits must be added at the beginning of the input buffer, before the first data bit. **Figure 26-160** shows an example of a 1220 bits block and how it should be placed in the system memory to be input for the MAPLE-B2 CRC processing:

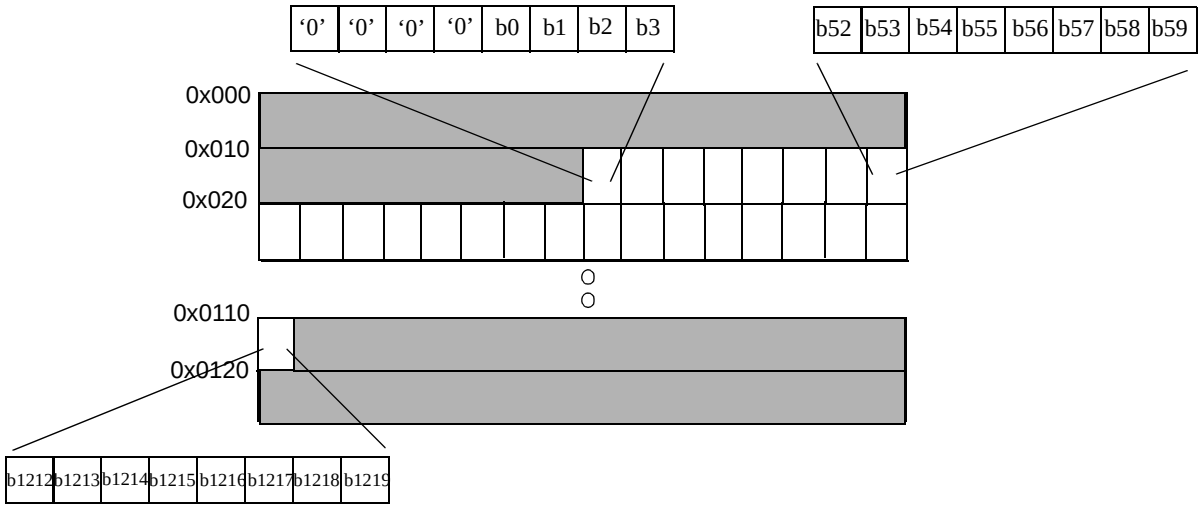


Figure 26-160. Example of Placing a Block of 1220 Bits in the System Memory for CRC Calculation

The MAPLE-B2 CRCPE performs the following two types of CRC processing:

- *CRC calculations*: If the [CHECK] bit in the CRC BD is reset, the MAPLE-B2 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-B2 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-B2, the CRC check failed. If the [FAIL] bit reset, the CRC check passed.

26.4.3.7.3 Byte Reverse CRC Processing

The CRCPE is capable of performing byte reverse CRC processing for both CRC calculations and CRC checks. By resetting the [RVRS_IN] bit in the CRC BD, the MAPLE-B2 performs byte reverse CRC processing on the input data. **Figure 26-161** describe the CRC processing of a 64 bits vector with respect to the [RVRS_IN] bit configuration:

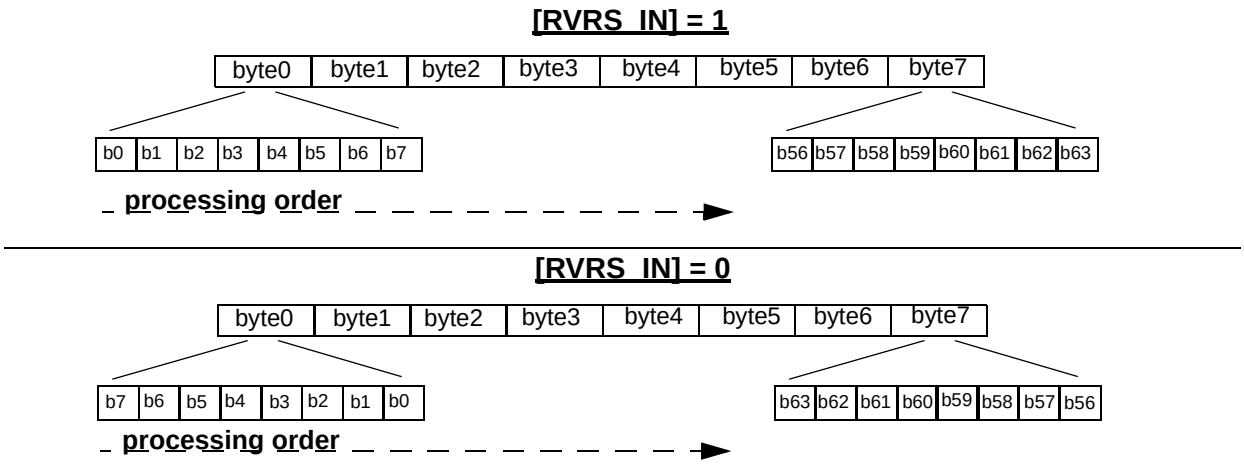


Figure 26-161. CRC Processing Order with Respect to [RVRS_IN] Bit

26.4.3.7.4 CRC Initial 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-B2 copies the value into the CRCPE machine before the beginning of the CRC processing.

The configured value of the [CRC_INIT] field must be with the same size as the CRC polynomial. For example, if the configured polynomial is CRC16-CCITT or CRC16 [CRC_POLY] = 010 or 011 respectively), the valid bit which is copied by MAPLE-B2 into the CRCPE is CRC_INIT[15:0] only. In this case, bits 31–16 of the [CRC_INIT] files remain not valid.

26.4.3.7.5 CRC Polynomials

The CRCPE supports eight types of CRC polynomials. The control of the CRC processing polynomial is done using the [CRC_POLY] field of the CRC BD according to **Table 26-123**:

Table 26-123. Supported CRC Polynomials with respect to [CRC_POLY] Field

[CRC_POLY]	CRC Type	CRC Polynomial
'000'	CRC24	$D^{24} + D^{23} + D^6 + D^5 + D + 1$
'001'	CRC24	$D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1$
'010'	CRC16-CCITT	$D^{16} + D^{12} + D^5 + 1$
'011'	CRC16	$D^{16} + D^{15} + D^2 + 1$
'100'	CRC32	$D^{32} + D^{26} + D^{23} + D^{22} + D^{16} + D^{12} + D^{11} + D^{10} + D^8 + D^7 + D^5 + D^4 + D^2 + D + 1$
'101'	CRC18	$D^{18} + D^{17} + D^{14} + D^{13} + D^{11} + D^{10} + D^8 + D^7 + D^6 + D^3 + D^2 + 1$
'110'	CRC12	$D^{12} + D^{11} + D^{10} + D^8 + D^5 + D^4 + 1$
'111'	CRC6	$D^6 + D^5 + D^3 + D^2 + D^1 + 1$

26.4.3.7.6 CRC Processing Results

The MAPLE-B2 is capable of performing the following processing of the CRC results:

26.4.3.7.6.1 Reverse Output Operation

When resetting the [RVRS_OUT] bit of the CRCPE BD, the MAPLE-B2 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. **Figure 26-162** describe the output reverse operation on a 16 bits CRC result:

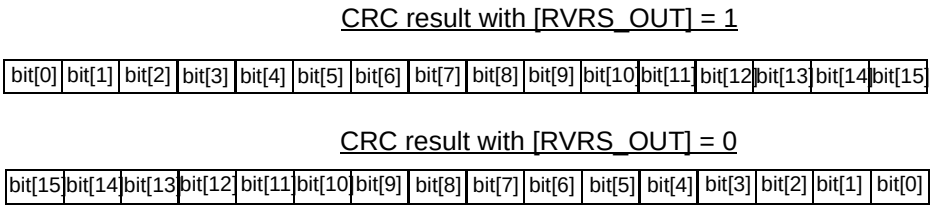


Figure 26-162. Reverse Output operation description

26.4.3.7.6.2 Inverse Output Operation

When setting the [INV_OUT] bit of the CRCPE BD, the MAPLE-B2 performs bit wise inverse operation on the CRC result. **Figure 26-163** describes the output Inverse operation on a 16-bit CRC result:

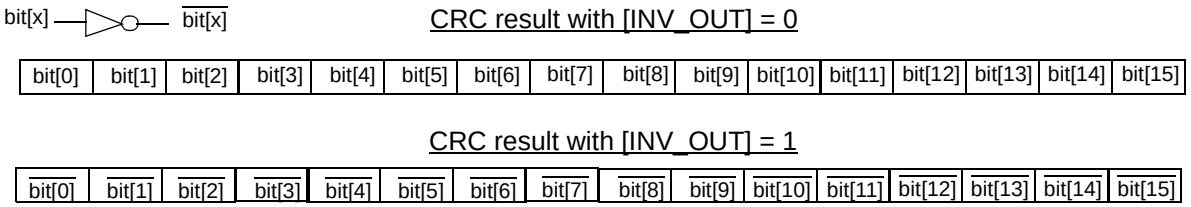


Figure 26-163. Inverse Output Operation Description

26.4.3.7.6.3 CRC Result Check/Calculate

If CRC check is required ([CHECK] = 1), MAPLE-B2 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 ([CHECK] = 0), MAPLE-B2 returns the CRC calculation result in the [CRC_RSLT] field of the CRC BD. By setting the [UPDATE] bit in the CRC BD, MAPLE-B2 also updates the CRC result at the end of the input buffer in the system memory. **Figure 26-164** describes an example of input buffer after the CRC BD completion when the [UPDATE] bit is set:

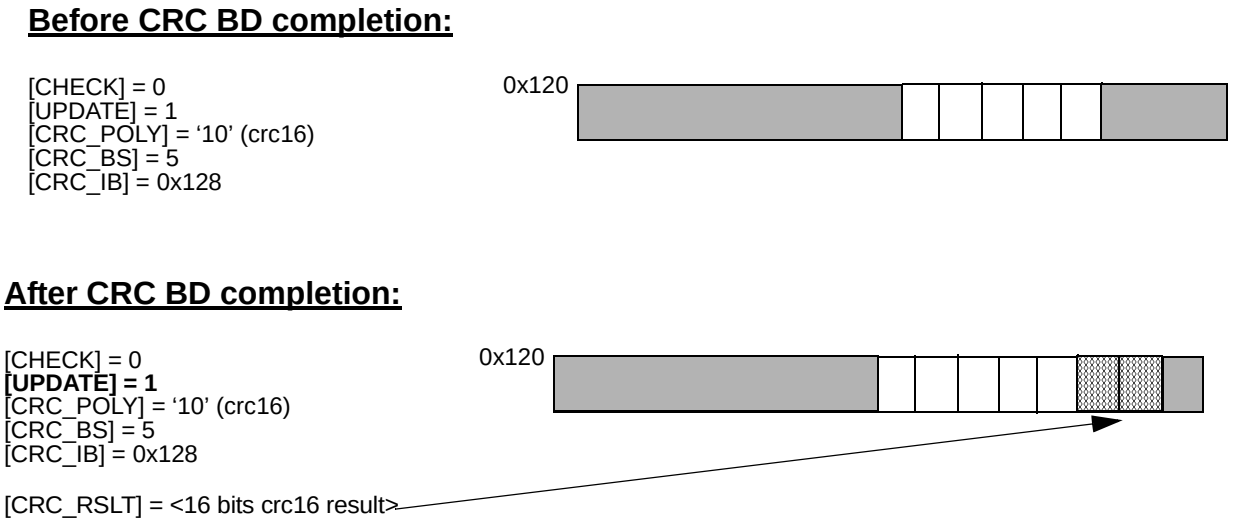


Figure 26-164. System Memory Update with CRC Result When [UPDATE] Bit is Set

26.4.3.8 Code Generation Operation

The MAPLE-B2 supports WCDMA Scrambling Code Generation acceleration. It is design to generate Uplink and Downlink scrambling and Hadamard codes, and to calculate their multiplication results. These calculation are executed using the CGPE instantiated in the Chip-Rate processing Element (CRPE).

26.4.3.8.1 CGPE Buffer Descriptor Structure

A detailed description of the Code generation Buffer Descriptor structure expected by the host for the CGPE BD rings is provided in **Section 26.5.4.8**, *CGPE Buffer Descriptor Structure*, on page 26-486.

26.4.3.8.2 CGPE Processing

The Codes Generator is responsible of generating uplink or downlink complex-valued scrambling codes and dedicated OVSF codes to be scrambled with uplink physical channels, as defined in 3GPP TS 25.213 (release 9):

- clause 4.3 - “Uplink spreading and modulation”
- clause 5 - “Downlink spreading and modulation”

The following sub-sections details its functionality.

26.4.3.8.2.1 OVSF Codes Generation

The **O**rthogonal **V**ariable **S**preading **F**actor (**OVSF**) codes are used as channelisation codes to preserve the orthogonality between a user’s different physical channels.

Figure 26-165 illustrates OVSF codes generation. Each level in the code tree defines OVSF codes of length SF, corresponding to a spreading factor of SF.

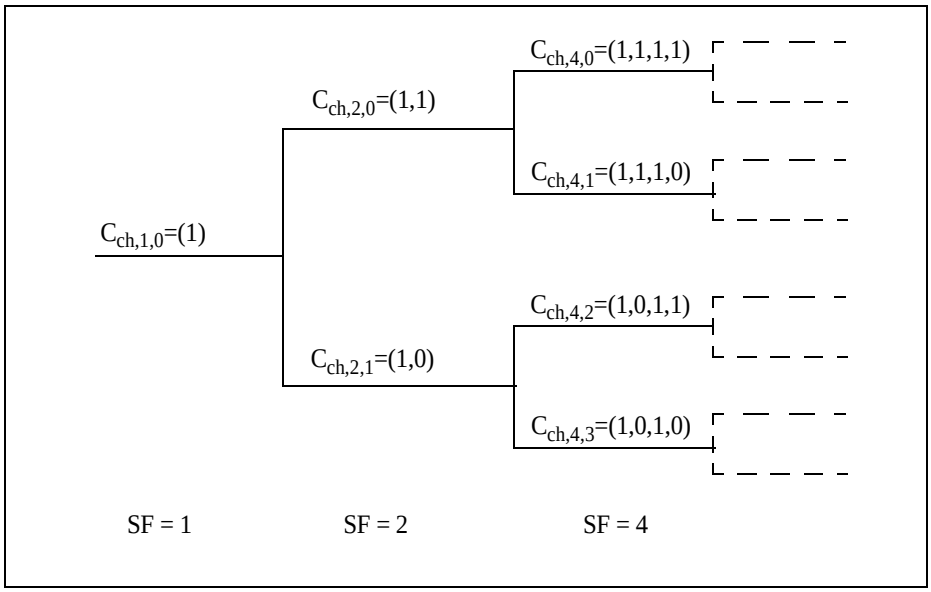


Figure 26-165. Code Tree for generation of Orthogonal Variable Spreading Factor codes

The OVSF codes are uniquely described as $C_{ch,SF,k}$, where SF is the Spreading Factor of the code and k is the code number, $0 \leq k \leq$ (Spreading Factor - 1). Controlling the OVSF value in the CGPE is done by the *OVSF_NUM* value of the CGPE BD.

The CG supports the following subset of Spreading Factor values: 1, 2, 4, 8, 16, 32, 64, 128 and 256 and is controlled in the CGPE by the *SF* field of the CGPE BD. The generation method for the channelisation code is defined in *3GPP TS 25.213 V9.1.0 (release 9) - clause 4.3.1 - "Channelisation codes"*

26.4.3.8.2.2 Uplink Scrambling Codes Generation

All uplink physical channels are subjected to scrambling with a complex-valued scrambling code. In WCDMA uplink transmissions, the scrambling code can either be short or long. Configuring the CGPE to generate uplink code is by setting the *UL_CODE* field of the CGPE BD (*UL_CODE* = 1) and controlling whether this generated code is long or short is done by setting the *LONG_MODE* field of the CGPE BD.

26.4.3.8.2.2.1 Long Complex Scrambling Codes Generation

The long scrambling sequences $C_{long,1,n}$ and $C_{long,2,n}$ are constructed from position wise modulo 2 sum of 38400 chip segments of two binary m-sequences generated by means of two generator polynomials of degree 25.

Let x, and y be the two m-sequences respectively.

The x sequence is constructed using the primitive (over GF(2)) polynomial $X^{25}+X^3+1$.

The y sequence is constructed using the polynomial $X^{25}+X^3+X^2+X+1$.

The sequence $C_{long,2,n}$ is a 16,777,232 chip shifted version of the sequence $C_{long,1,n}$. *PN_NUM[23:0]* (of the CGPE BD) is the 24 bit binary representation of the scrambling sequence number n.

The x sequence depends on the chosen scrambling sequence number n and is denoted x_n, in the sequel. Furthermore, let $x_n(i)$ and y(i) denote the i_{th} symbol of the sequence x_n and y, respectively.

The m-sequences xn and y are constructed as:

Initial conditions:

$x_n(0)=PN\_NUM[0]$, $x_n(1)= PN\_NUM[1]$, ... ,$x_n(23)= PN\_NUM[23]$, $x_n(24)=1$.
$y(0)=y(1)=...=y(23)= y(24)=1$.

Recursive definition of subsequent symbols:

$x_n(i+25) =x_n(i+3) + x_n(i)$ modulo 2, $i=0,..., 2^{25}-27$.
$y(i+25) = y(i+3)+y(i+2) +y(i+1) +y(i)$ modulo 2, $i=0,..., 225-27$.

Define the binary Gold sequence Z_n by:

$z_n(i) = x_n(i) + y(i)$ modulo 2, $(i = 0, 1, 2, ..., 2^{25}-2)$.

The real valued Gold sequence Z_n is defined by:

$$Z_n(i) = \begin{cases} +1 \text{ if } Z_n(i) = 0 \\ \\ -1 \text{ if } Z_n(i) = 1 \end{cases} \qquad \text{for } i = 0,1,...,2^{25}-2$$

Now, the real-valued long scrambling sequences $C_{long,1,n}$ and $C_{long,2,n}$ are defined as follows:

$C_{long,1,n}(i) = Z_n(i)$, $(i = 0, 1, 2, ..., 2^{25}-2)$
$C_{long,2,n}(i) = Z_n((i + 16777232)$ modulo $(2^{25}-1))$, $i = 0, 1, 2, ..., 2^{25}-2$.

Finally, the complex-valued long scrambling sequence $C_{long,\,n}$, is defined as:

$$C_{long,n}(i) = Z_{long.1,n}(i)\ (1 + j(-1)^i\ C_{long,2,n}(2\lfloor \frac{i}{2} \rfloor))\ ,\quad i = 0, 1,..., 2^{25}-2$$

Figure 26-166 illustrates the Uplink scrambling sequence generator:

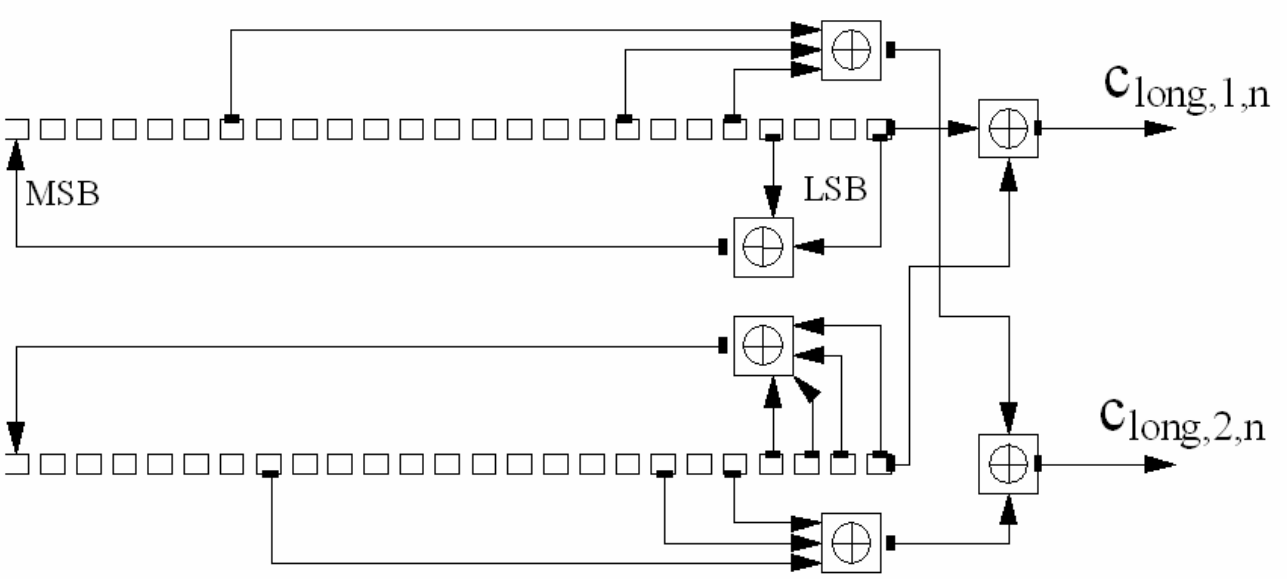


Figure 26-166. Configuration of the Uplink Long Scrambling Sequence Generator

26.4.3.8.2.2.2 Short Complex Scrambling Codes Generation

The short scrambling sequences $C_{short,1,n}(i)$ and $C_{short,2,n}(i)$ are defined from a sequence from the family of periodically extended *S(2)* codes.

PN_NUM[23:0] is the 24 bit binary representation of the code number n.

The n_{th} quaternary *S(2)* sequence $Z_n(i)$, 0 <= n <= 16777215, is obtained by modulo 4 addition of three sequences, a quaternary sequence *a(i)* and two binary sequences *b(i)* and *d(i)*, where the initial loading of the three sequences is determined from the code number n.

The sequence $Z_n(i)$ of length 255 is generated according to the following relation:

$$Z_n(i) = a(i) + 2b(i) + 2d(i)\ \text{modulo}\ 4,\ i = 0, 1,..., 254$$

The quaternary sequence *a(i)* is generated recursively by the polynomial

$$g0(x) = x^8+3x^5+x^3+3x^2+2x+3$$

as:

$$a(0) = 2 * PN\_NUM[0] + 1\ \text{modulo}\ 4$$
$$a(i) = 2 * PN\_NUM[i]\ \text{modulo}\ 4,\ i = 1, 2,..., 7$$
$$a(i) = 3a(i-3) + a(i-5) + 3a(i-6) + 2a(i-7) + 3a(i-8)\ \text{modulo}\ 4,\ i = 8, 9,..., 254$$

The binary sequence *b(i)* is generated recursively by the polynomial

$$g1(x) = x^8+x^7+x^5+x+1$$

as:

```
b(i) = PN_NUM[8+i] modulo 2, i = 0, 1,..., 7
b(i) = b(i-1) + b(i-3) + b(i-7) + b(i-8) modulo 2, i = 8, 9,..., 254
```

The binary sequence *d(i)* is generated recursively by the polynomial

$g2(x) = x^8+x^7+x^5+x^4+1$

as:

```
d(i) = PN_NUM[16+i] modulo 2, i = 0, 1,..., 7
d(i) = d(i-1) + d(i-3) + d(i-4) + d(i-8) modulo 2, i = 8, 9, ..., 254
```

The sequence $Z_n(i)$ is extended to length 256 chips by setting $Z_n(255) = Z_n(0)$.

Table 26-124 describes the mapping from $Z_n(i)$ to the real-valued binary sequences $C_{short,1,n}(i)$ and $C_{short,2,n}(i)$, i = 0, 1,..., 255

Table 26-124. Mapping from $Z_n(i)$ to $C_{short,1,n}(i)$ and $C_{short,2,n}(i)$

$Z_n(i)$	$C_{short,1,n}(i)$	$C_{short,2,n}(i)$
0	+1	+1
1	-1	+1
2	-1	-1
3	+1	-1

Finally, the complex-valued short scrambling sequence $C_{short,n}$, is defined as:

$$C_{short,n}(i) = C_{short.1,n}(i \bmod 256)\ (1 + j(-1)^i\ C_{short,2,n}(2\lfloor \frac{i \bmod 256}{2} \rfloor))\ ,\quad i = 0,1,2,...$$

Figure 26-167 illustrates the implementation of the short scrambling sequence generator for the 255 chip sequence to be extended by one chip.

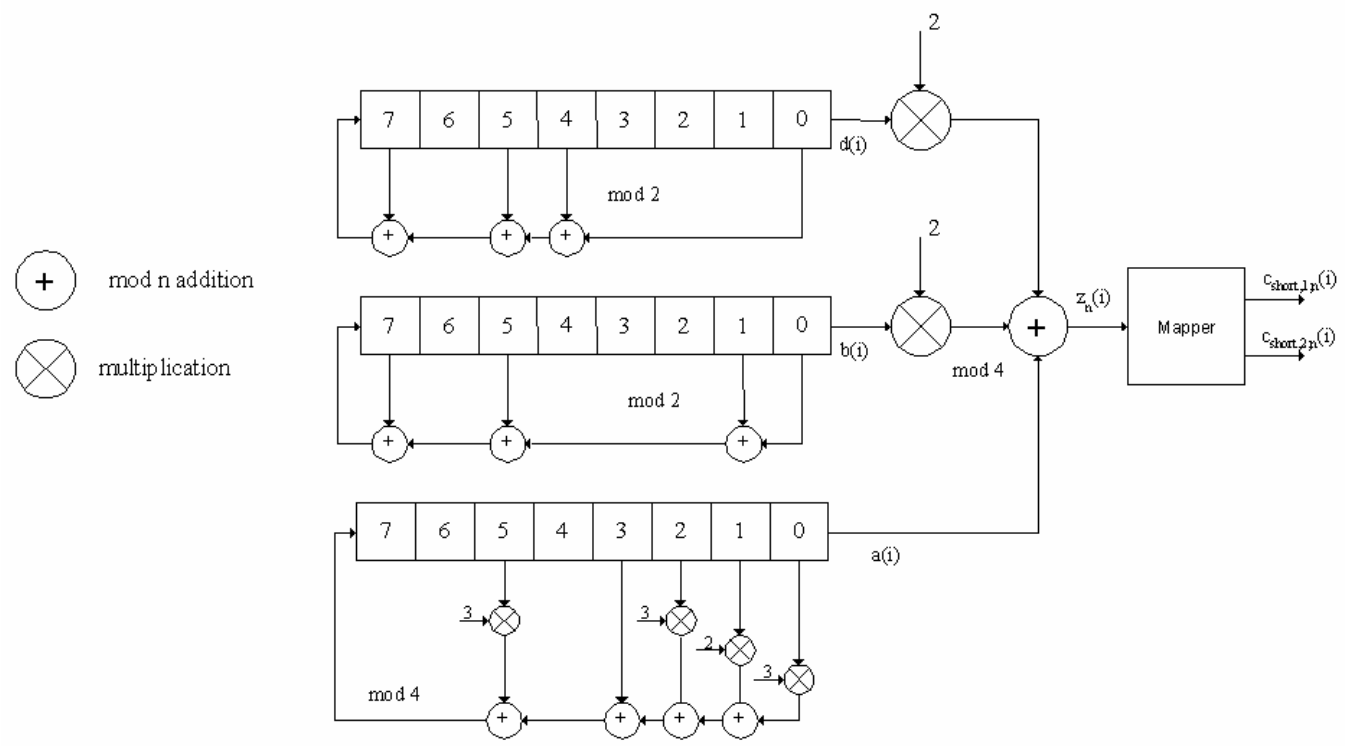


Figure 26-167. Uplink Short Scrambling Sequence Generator for 255 Chips

26.4.3.8.2.3 Downlink Scrambling Codes Generation

Configure the CGPE to generate downlink code by resetting the *UL_CODE* field of the CGPE BD (*UL_CODE* = 0).

The sequence of complex valued chips are scrambled (complex chip-wise multiplication) by a complex-valued scrambling code $S_{dl,n}$.

The scrambling code sequences are constructed by combining two real sequences into a complex sequence.

Each of the two real sequences are constructed as the position wise modulo 2 sum of 38400 chip segments of two binary m sequences generated by means of two generator polynomials of degree 18.

Let x and y be the two sequences respectively.

The x sequence is constructed using the primitive (over GF(2)) polynomial:

$1+X^7+X^{18}$.

The y sequence is constructed using the polynomial:

$1+X^5+X^7+ X^{10}+X^{18}$.

The sequence depending on the chosen scrambling code number n is denoted Z_n, in the sequel. Furthermore, let *x(i)*, *y(i)* and $Z_n(i)$ denote the i_{th} symbol of the sequence x, y, and Z_n, respectively.

The m-sequences x and y are constructed as:

Initial conditions:

$x(0)=1,\ x(1)=x(2)=\ldots=x(16)=x(17)=0.$

$y(0)=y(1)=\ldots=y(16)=y(17)=1.$

Recursive definition of subsequent symbols:

$x(i+18) = x(i+7) + x(i) \text{ modulo } 2,\ i=0,\ \ldots,\ 2^{18}-20.$

$y(i+18) = y(i+10)+y(i+7)+y(i+5)+y(i) \text{ modulo } 2,\ i=0,\ldots,\ 2^{18}-20.$

The n_{th} Gold code sequence Z_n, n=0,1,2,...,2^{18}-2, is then defined as:

$Z_n(i) = x((i+n) \text{ modulo } (2^{18}-1)) + y(i) \text{ modulo } 2,\ i=0,\ldots,\ 2^{18}-2.$

These binary sequences are converted to real valued sequences Z_n by the following transformation:

$$Z_n(i) = \begin{cases} +1 & \text{if } Z_n(i) = 0 \\ -1 & \text{if } Z_n(i) = 1 \end{cases} \quad \text{for } i = 0,1,\ldots,2^{18}-2$$

Finally, the n_{th} complex scrambling code sequence $S_{dl,n}$ is defined as:

$S_{dl,n}(i) = Z_n(i) + j\ Z_n((i+131072) \text{ modulo } (2^{18}-1)),\ i=0,1,\ldots,38399.$

Note that the pattern from phase 0 up to the phase of 38399 is repeated.

Figure 26-168 describe the Downlink Scrambling Code Generation.

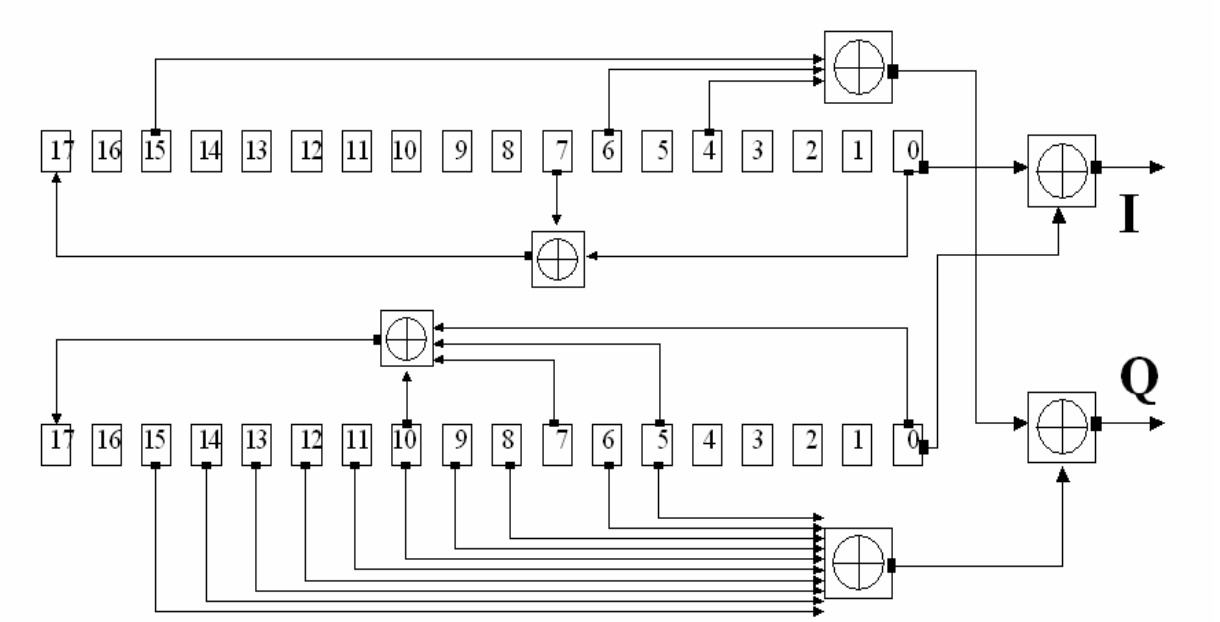


Figure 26-168. Configuration of Downlink Scrambling Code Generation

26.4.3.8.2.4 Code Generation Offset

The CGPE support generating code starting at any offset in the frame. This allows generating the scrambling code in portions where for each portion all which is required is to set the chip offset of the frame. The CG in return calculates internally the initial state for the given chip offset and generates the code starting the calculated initial state.

Controlling the chip offset for each CGPE job is done by setting the *START_OFFSET* field of the CGPE BD. The *START_OFFSET* field represent Sub-Slot alignment (256 chips alignment).

26.4.3.8.3 Output Data Structure

The CGPE support two types of generated output codes:

1. Each I and Q pair are represented by a pair of bits. A 128 bits of generated code include 64 such pairs.
2. Each I and Q pair are represented by 16 bits (8 bits I, and 8 bits Q). A 128 bits of generated code include 8 such pairs.

In case 16 bits representation for I,Q pair is chosen, it is up to the host to set the values of the 8 bits ‘0’ representation and 8 bits ‘1’ representation.

Controlling the generated code output type is done by setting the *OUT_MODE* field of the CGPE BD. If 16 bits representation for I,Q pair is chosen, the *CFG_VAL_0* and *CFG_VAL_1* fields of the CGPE BD determines the values to represent ‘0’ and ‘1’ respectively.

The CGPE also supports conjugating the output results. By setting the *CONJ_EN* bit of the CGPE BD, the output results is conjugated.

The size of the generated code is determined by the *CODE_SIZE* field of the CGPE BD. Its value (plus 1) represent the amount of 128 bits chunks to be generated by the CGPE. For example a value 5 indicates that the generated code size is: (5+1)*128 = 768 bits.

The generated code is placed (by MAPLE-B2) in the system memory in the address pointed by the *CODE_RSLT_PTR* field of the CRPE BD.

Figure 26-169 describe two examples of the output results of the CGPE per given BD configuration:

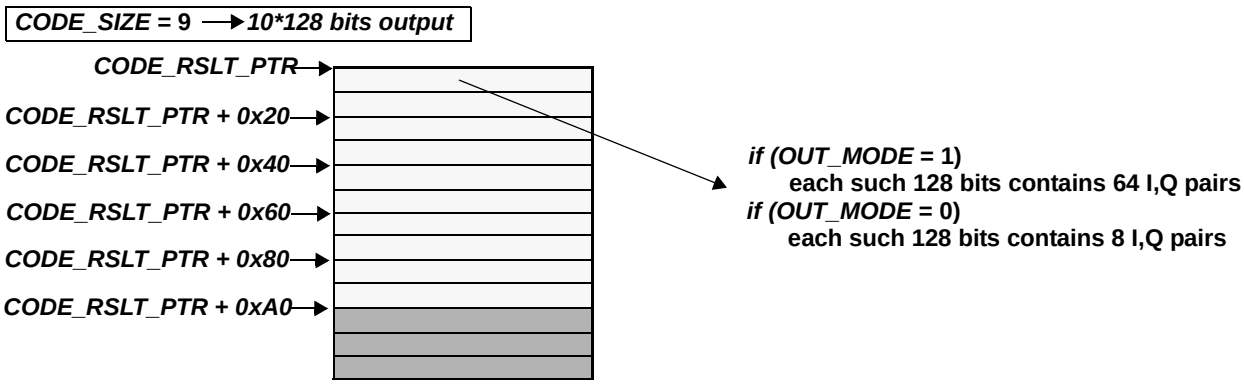


Figure 26-169. CGPE Output Data Structure Examples

26.4.3.9 CONVPE Convolution and Correlation Processing Operation

The MAPLE-B2 supports convolution and correlation tasks with implementation in frequency domain. This is done using the virtual processing element CONVPE. The CONVPE is an abstracted element which combines Fourier Transforms, Multiply/Accumulate, and associated control utilizing the following:

- 3 eFTPE engines.
- FDU logic (dedicated logic located in the EQPE block and utilizing its memory. See **Section 26.4.3.5.7**, *CONVPE Support—FDU Processing*, on page 26-186).
- Internal DMA
- RISC engines.
- MBus DMAs

A data flow diagram of CONVPE processing shown in **Figure 26-170**:

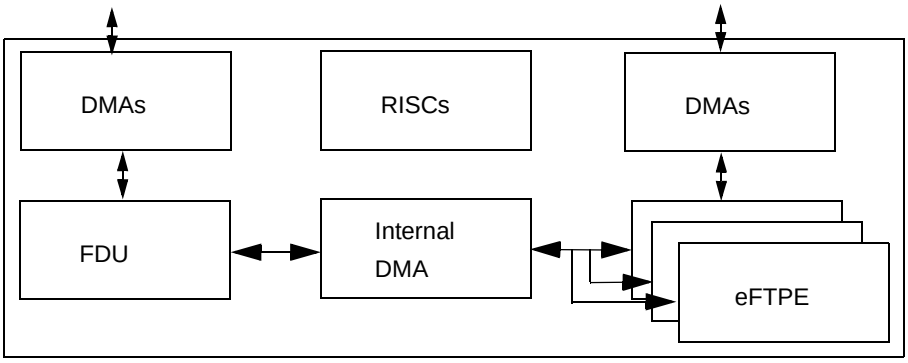


Figure 26-170. CONVPE Data Flow

Main processing flow of CONVPE based on FFT followed by Multiply-Accumulate functionality executed in FDU unit and followed by IFFT. First portion of the data flow is optional, and data streams which are available in frequency domain from system memory can be

fetched by the MAPLE-B2 system DMAs directly into FDU memories omitting this way portions of FFT processing.

26.4.3.9.1 CONVPE Programming

The CONVPE programming is based on Buffer Descriptors (BDs) that are populated with high level task parameters. These parameters include a pointer to a detailed Task Descriptor (TD) which is located in system memory. The high level tasks parameters embedded in the CONVPE BDs are minimized to the information required for task scheduling aspects and description of the TD type, location and size. **Figure 26-171** depicts this concept.

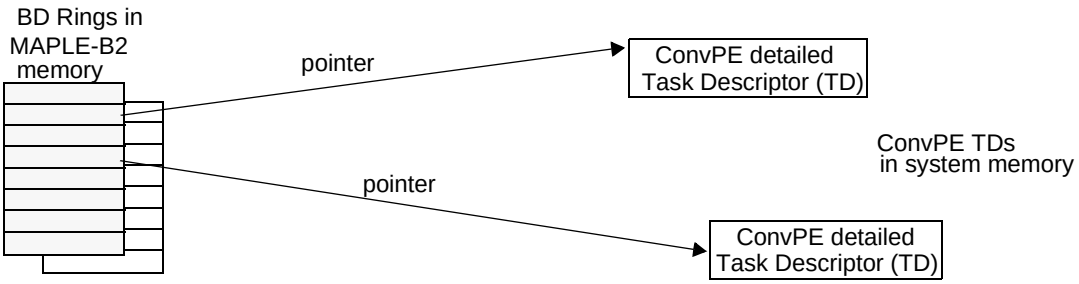


Figure 26-171. CONVPE Task Descriptors Concept

26.4.3.9.1.1 CONVPE Buffer Descriptor Structure

Section 26.5.4.9, *CONVPE Buffer Descriptor Structure*, on page 26-490 contains a detailed description of the CONVPE Buffer Descriptors structure as it should be written to the CONVPE BD rings by the host.

26.4.3.9.1.2 CONVPE Detailed Tasks Descriptors

The following sections describe in details each of the possible tasks as described in the *TASK_TYPE* field of the CONVPE BD.

26.4.3.9.1.3 RACH Preamble Correlations Task Description

The UMTS RACH task performs correlations between RACH preamble signatures and antenna data diversity branches. Number of RACH signatures and number of antenna data diversity branches are parameters of the task which can be configured by the host. In addition, coherent correlation window size and data overlaps for frequency domain correlation implementation are parameters that provide the user-required flexibility for this task. **Section 26.5.4.10**, *CONVPE RACH Preamble Correlations Task Descriptor*, on page 26-491 provides a detailed description of the Task Descriptor (TD) layout used for RACH Preamble Correlation.

The RACH correlation task is performed on a data stream with overlaps due to nature of frequency domain implementation. The overlap size is typically defined based on Base Station deployment cell radius. **Figure 26-172** demonstrates and example of correlation task with the relevant parameters

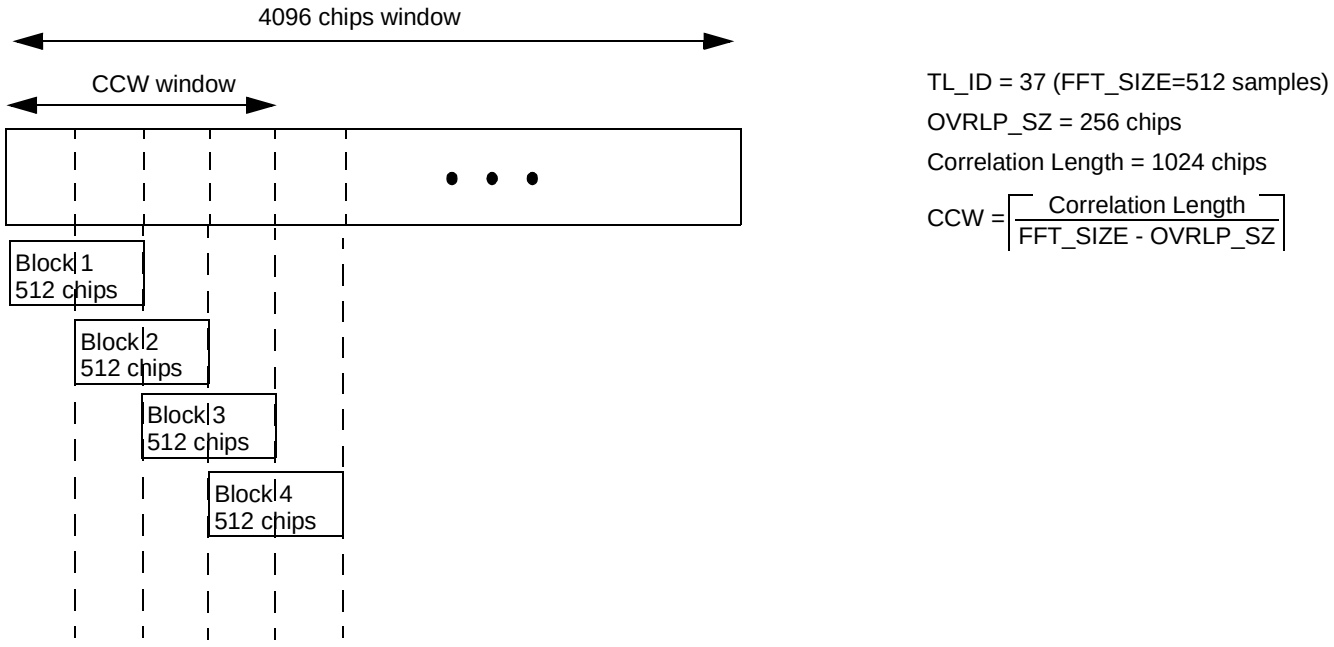


Figure 26-172. RACH Correlation Task Input Data Parameters

26.4.3.9.1.3.1 RACH Preamble Task Limitations

The following configuration limitation applies to the RACH preamble correlation task:

```
BLOCKS_NUM = Sum(SIGN_INDX[i]) x ANT_DIV_BCH ; i = 0,...,15
BLOCK_SIZE = 4 x FFT_SIZE[TL_ID] ; //FFT_SIZE is the number of samples in the FFT transform
1) CCW x BLOCK_SIZE x ANT_DIV_BCH /2 ≤ 16384
2) if (BLOCKS_NUM > 4)
     if ((BLOCKS_NUM-1)%8 ≥ 4)
          (Ceiling(BLOCKS_NUM/8)+1) x BLOCK_SIZE ≤ 12288
     else
          Ceiling(BLOCKS_NUM/8) x BLOCK_SIZE ≤ 12288
```

Following is an example of a valid task:

- Task's parameters:
  - *TL_ID* = 37 (FFT_SIZE=512)
  - *ANT_DIV_BCH* = 2
  - *SING_INDX* = 0x0351
  - *CCW* = 8
- Calculations:
  - BLOCKS_NUM = 5 x 2 = 10
  - BLOCK_SIZE = 4 x 512 = 2048

- Limitations check:
  - 1) 8 x 2048 x 2 / 2 = 16384 ≤ 16384
  - 2) (10 > 4) ==> ((10 - 1) % 8 < 4) ==> Ceiling(10/8) x 2048 = 4096 ≤ 12288

26.4.3.9.1.3.2 RACH Preamble Input Data Structure - Antenna data

The input antenna data can be in frequency or in time domain. The data structure is based on a continuos buffer with start address and constant offset between data vectors. Based on the *ANT_FD* field, the data vector consists of chips in time domain with 8 bit I, 8 bit Q representation (*ANT_FD* = 0) or in frequency domain with 16 bit I, 16 bit Q representation (*ANT_FD* = 1). For a given RACH CONVPE task the data structure can not be mixed between frequency and time domain representation. **Figure 26-173** shows an example of this data structure.

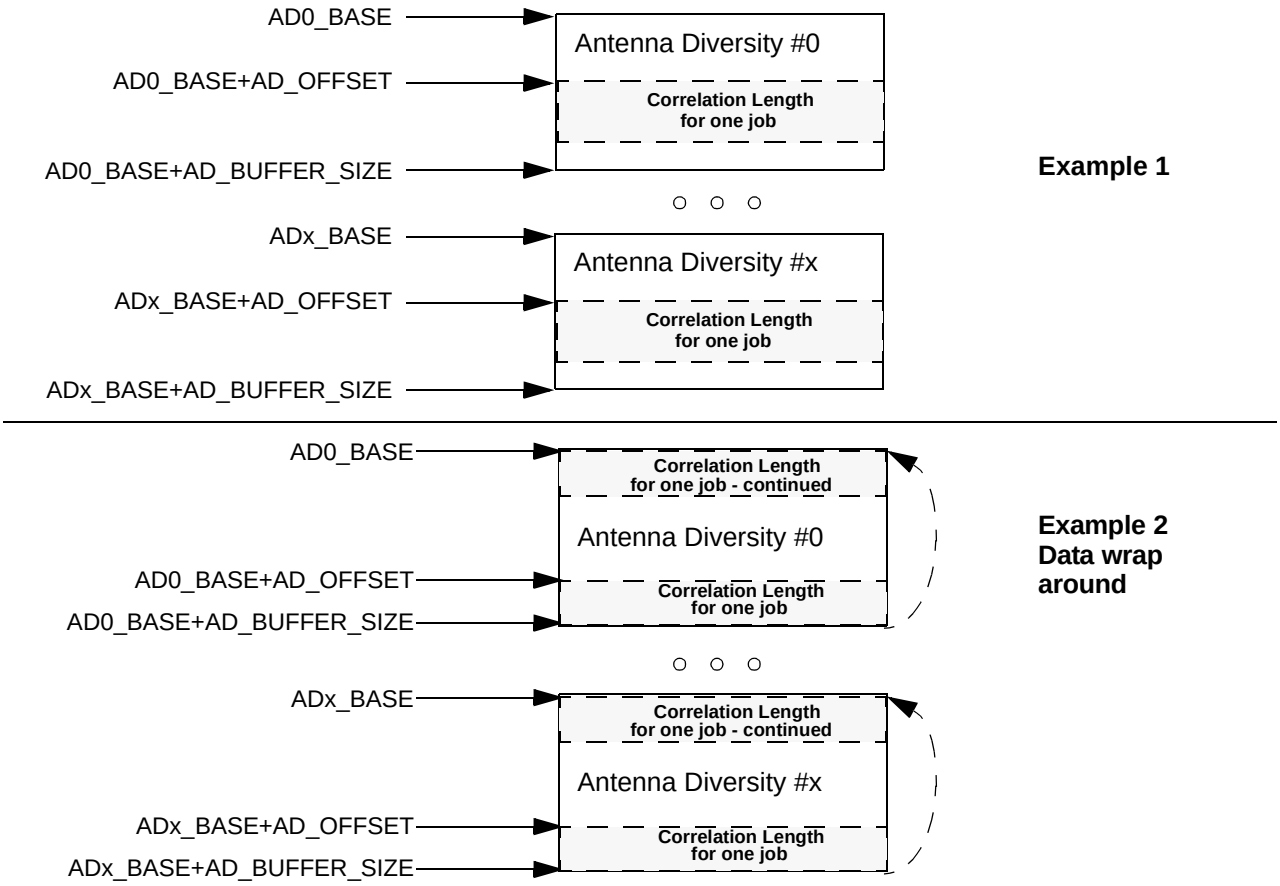


Figure 26-173. Input Antenna Data Structures

The base address *ADx_BASE* is 8 bytes aligned pointer in system memory. The offset is represented by a 32 bit word providing flexibility to allocation of the different antenna data in system memory. The order of antenna stream diversity branches is arbitrary selected by the host, and represents the order of the results located based on *CORR_RSLT_ADDR* base address.

If the input antenna data is available in the frequency domain, its representation should be in block floating point. The block floating point exponent per each diversity branch data vector is 8 bit wide representing a range of -127 to 128. The exponents are concatenated one after another and located at the address pointed by *AD_FD_SCALE_ADDR*.

26.4.3.9.1.3.3 RACH Preamble Input Data Structure - Signatures

The MAPLE-B2 expects the input signatures in frequency domain. These signatures should be generated in frequency domain prior to calling the RACH task with zero padding equals the overlap size (*OVRLP_SZ*) of the RACH task per frequency domain block. **Figure 26-174** shows the signatures expected data structure in system memory. The signatures are located based on *SIGN_ADDR* field. Signatures relevant to the specific RACH job are indicated by *SIGN_INDX* field. Every bit which in *SIGN_INDX* field represents one of 16 different signatures in use if bit is set.

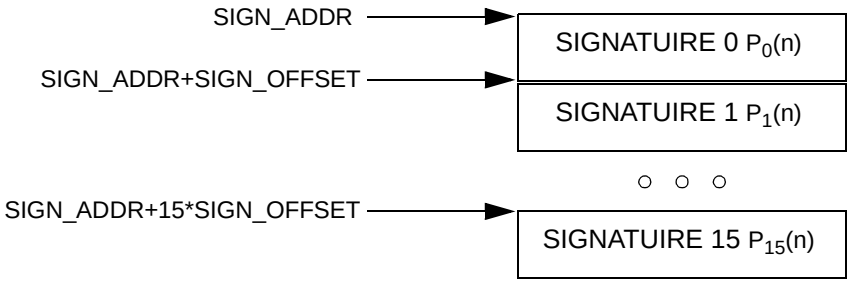


Figure 26-174. Signatures Data Structure in system memory

26.4.3.9.1.3.4 RACH Preamble Output Data Structure

Figure 26-175 shows the structure of the results as placed in system memory. RACH preamble correlation results are located in system memory based on *CORR_RSLT_ADDR*.

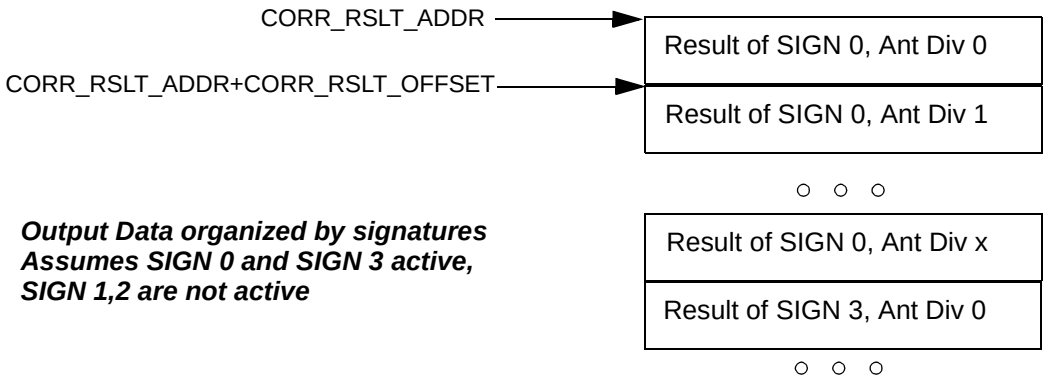


Figure 26-175. RACH Preamble Correlation Results Data Structure in system memory

The correlation results per antenna diversities for each signature are in the same order as the antenna diversities inputs. The size of each result vector is based on the Overlap Size (*OVRLP_SZ)* field. Results are represented in 16I, 16Q format, thus results are offset from each other by 4*[IFFT Transform Length] bytes.

If the antenna data is input to the CONVPE RACH preamble correlations task in the time domain (*ANT_FD* = 0), and the frequency domain antenna data is required in system memory for future tasks, *ANT_FFT_WBE* field should be set. In that case the MAPLE-B2 outputs the frequency domain antenna data into system memory. These frequency domain antenna data results are written to address as described in the *ANT_DATA_FD_ADDR* pointer and are organized in the same order as the input data. The output results data structure is based on 16I,16Q values with block floating point exponent located based on *AD_FD_SCALE_ADDR* pointer with 8 bit exponent for each FFT segment. The following figure describe the frequency domain antenna diversities outputs:

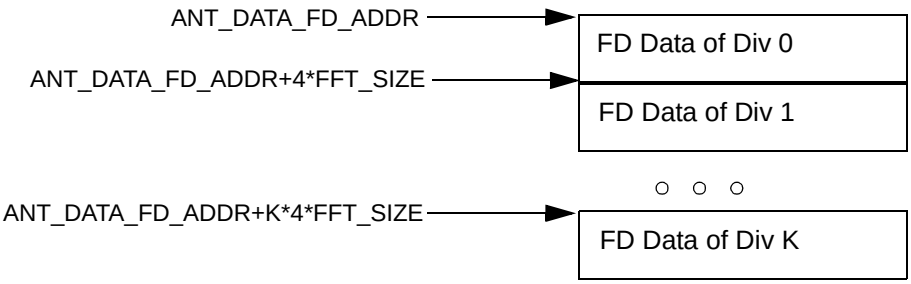


Figure 26-176. Antenna Data optional outputs in Freq. domain example for RACH task

Note: If the input data for CONVPE RACH preamble correlation task is already in frequency domain, *ANT_DATA_FD_ADDR* pointer and respective data structure are not used. The input data in frequency domain should be based on *AD_BASE* pointer and relative data structure as shown in **Figure 26-173**.

26.4.3.9.1.4 Path Searcher Correlations Descriptor

The MAPLE -B2 executes Path Searcher correlations in frequency domain utilizing CONVPE functionality. The correlations are performed between antenna diversity branches and pilot sequences per user. The correlation is performed on maximum coherent window equivalent to one radio slot.

The pilot sequences are generated, by the CONVPE, in time domain and transformed to frequency domain as part of the correlation task. Per user parameters of scrambling sequence, type of code, frequency correction parameters, slot format and offsets are supplied as part of the CONVPE Path Searcher task.

The task is executed with multiple antenna diversities over number of users in a single descriptor. Thus, the descriptor contains two portions, first portion is task generic, with shared information for all users. This portion describes the antenna data and correlation parameters. The second portion is user specific, which defines the per-user parameters of the task, focusing on scrambling sequence, slot format, offsets and frequency correction parameters. Path search of up to 16 users can be executed in a single task.

Section 26.5.4.11, *CONVPE Path Searcher Task Descriptor*, on page 26-495 has a detailed description of the CONVPE UMTS Path searcher task fields as they should be written by the host in system memory pointed by the CONVPE Buffer Descriptor *TASK_ADDR* field.

26.4.3.9.1.4.1 CONVPE Path Search Task limitations

The following configuration limitation applies to the Path Search correlation task:

```
BLOCKS_NUM = (NUSERS+1) x ANT_DIV_BCH ;
BLOCK_SIZE = 4 x FFT_SIZE[TL_ID] ; //FFT_SIZE is the number of samples in the FFT transform
1) CCW x BLOCK_SIZE x ANT_DIV_BCH / 2 ≤ 16384
2) if (BLOCKS_NUM > 4)
     if ((BLOCKS_NUM-1)%8 ≥ 4)
          (Ceiling(BLOCKS_NUM/8)+1) x BLOCK_SIZE ≤ 12288
     else
          Ceiling(BLOCKS_NUM/8) x BLOCK_SIZE ≤ 12288
```

Following is an example of a valid task:

- Task’s parameters:
  - *TL_ID* = 37; // (FFT_SIZE=512)
  - *ANT_DIV_BCH* = 2;
  - *CCW* = 8;
  - *NUSERS* = 7
- Calculations:
  - BLOCKS_NUM = 8 x 2 = 16
  - BLOCK_SIZE = 4 x 512 = 2048
- Limitations check:
  - 1) 8 x 2048 x 2 / 2 = 16384 ≤ 16384
  - 2) (16 > 4) ==> (15%8 ≥ 4) ==> 3 x 2048 = 6144 ≤ 12288

Similar to RACH preamble correlations, input antenna data can be in frequency or in time domain. For details of the data structure see **Figure 26-173**.

CONVPE generates internally the pilot sequences in frequency domain, using the scrambling code of long or short type based on *LS* filed of task descriptor.

Path searcher correlation results are located in system memory based on the *U_CORR_RSLT_ADDR* field of the Task Descriptor. The results are located in consecutive form

in the same order as input data. Details of this data structure are shown in **Figure 26-175**, but instead of per-signature data structure, it is organized on per-user basis.

The size of each result vector is based on the overlap size (*OVRLP_SZ* field). Results are represented in 16I, 16Q format, thus results are offset from each other by 4**OVRLP_SZ* bytes. The results include a scaling exponent per vector, which is located based on the *AD_FD_SCALE_ADDR* field.

26.4.3.9.2 FDU Processing

As part of CONVPE processing (see **Section 26.4.3.9**, *CONVPE Convolution and Correlation Processing Operation*), the EQPE includes a dedicated logic (FDU logic) which assists with vector multiplication processing. The FDU logic, which is not functional during Equalization processing (LNEQ, MLEQ, TS and MIV modes), is targeted for CONVPE processing assist only. As both Equalization logic and FDU logic share the same EQPE internal resources (memories), they cannot work in parallel, that is, the EQPE is either in FDU processing mode or Equalization processing mode.

The FDU logic cannot be directly controlled by the host therefore has no programming model interface of its own. It can only be controlled by MAPLE-B2 firmware as part of the CONVPE processing. The following sections describe the FDU logic processing and functionality.

26.4.3.9.2.1 FDU Operation

The FDU operation in the job is defined on vectors of samples where each sample in the vector is 4 bytes of size (16I,16Q) in 1Q15 representation. In addition, each vector has one scaling factor, the scaling factor is 8-bit signed.

The FDU includes the following main buffers for samples:

1. A buffer. Size: 8KByte.
2. B buffer. Size: 32KByte.
3. C buffer. Size: 96KByte.
4. Output Buffer. Size: 32KByte.

and the following buffers which holds Scale values:

1. A Scale buffer
2. B Scale buffer
3. C Scale buffer
4. Output Scale buffer

In general, the FDU executes vectors Multiply/Accumulate operations. The following figure describe the high level functionality of the FDU:

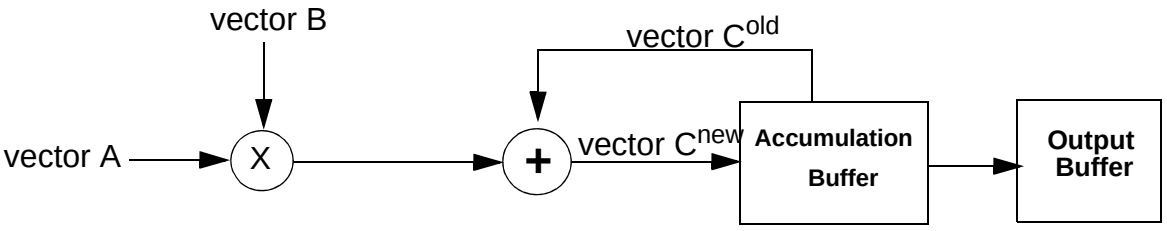


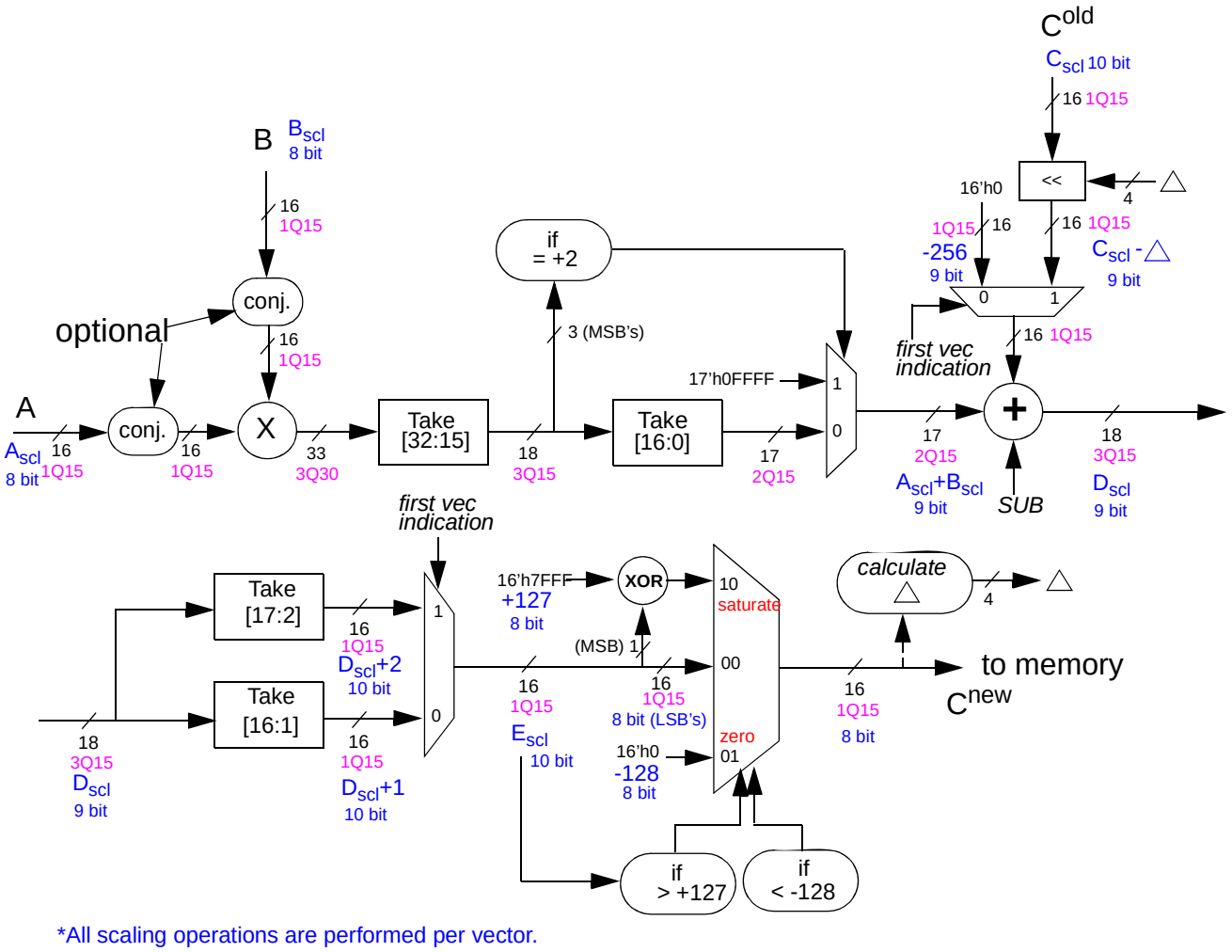
Figure 26-177. FDU High Level Functionality

In addition to the basic functionality described in **Figure 26-177**, the FDU supports the following:

— Optional conjugation operation on A input vector or/and B input vector
— Optional subtraction operation instead of accumulate, that is, $C^{new} = A \times B - C^{old}$
— Saturation of overflown samples.

26.4.3.9.2.2 FDU Arithmetic Scheme

The FDU operates in block floating point arithmetic, which includes one scaling factor for each vector. The FDU detailed calculation implementation is described in **Figure 26-178**.



*All scaling operations are performed per vector.

Figure 26-178. FDU Calculation Implementation[1]

During the writing of the C^{new} samples to the memory, the FDU gathers statistics on the mantissa to locate the position of the leading '1' of the samples - this is called 'delta' sample (△) and it represents the number of shift-left operations that can be performed on the data without overflow. When the block is finished, the FDU writes the delta sample, to the delta buffer. This delta value is then used to adaptively scale up the mantissa to gain maximum precision.

Each delta samples is 8-bits with values 0-15.

1. The illustration describes the calculation when MODE=0, when MODE=1 the calculation is identical with B/C vectors are in opposite places.

26.4.3.9.2.3 FDU Scaling

The FDU performs block-floating-point arithmetic and therefore requires and produces one scaling factor for each vector. The FDU should get 2 types of scaling factors: A scaling factor and B scaling factor. On FDU processing completion the C scaling factor is available for further processing. As the FDU is capable of working on multiple vectors each of the scale values is kept in a dedicated buffer which can hold multiple values as per FDU requirements.

26.4.4 External Masters Support Using Serial RapidIO Doorbell

The MAPLE-B2 is capable of working with masters external to its DSP device, using the Serial RapidIO interface.

26.4.4.1 Serial RapidIO Doorbell Parameters Configuration

If an external master, which is connected via the Serial RapidIO port, wishes to master the MAPLE-B2, the following parameters must be initialized:

- **M<pe>[1]BR(H/L)PBxP[HOST_ID]**—16-bit parameter Host ID used by MAPLE-B2 for WORD3[EDID:DID] fields in the Type10 Outbound Doorbell Descriptor in the eMSG unit. For details, see **Section 16.3.9.1,** *Type10 Outbound Doorbell Descriptor Format*.
- **M<pe>BR(H/L)PBxP[P_TR_IF]**—4-bit parameter for the Port Target Interface for RapidIO Ports 1 and 2. This field is used by the MAPLE-B2 for the WORD4[TINT] field in the Type10 Outbound Doorbell Descriptor in the eMSG unit. For details, see **Section 16.3.9.1,** *Type10 Outbound Doorbell Descriptor Format*.
- **M<pe>BR(H/L)PBxP[EXT_MST]**—2 bit parameter which describes whether the MAPLE-B2's current ring's master is connected directly to one of the MAPLE-B2's regular interrupt lines or the ring's master is external and connected to the DSP device via the Serial RapidIO port.
- **HSPxBAP[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 HSPxBAP[HS_EN] is set.
- **HSPxBAP[HS_EN]**—1 bit parameter which describes for the MAPLE-B2 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-B2. This feature is added to avoid any coherency problems due to multiple masters accessing simultaneously to a Doorbell controller.
- **MDHSIDCP[TV_HS_ID]**—8 bit parameter of eTVPE External Hardware Semaphore ID, used by the MAPLE-B2 when accessing the Hardware Semaphore register for eTVPE doorbell generation.

1. <pe> stands for TV, FF or DF.

- **MDHSIDCP[DF_HS_ID]**—8 bit parameter of DFTPE External Hardware Semaphore ID, used by the MAPLE-B2 when accessing the Hardware Semaphore register for DFTPE doorbell generation.
- **MDHSIDCP[FF_HS_ID]**—8 bit parameter of FFTPE External Hardware Semaphore ID, used by the MAPLE-B2 when accessing the Hardware Semaphore register for FFTPE doorbell generation.
- **MDGCP[BD_PR]**—2 bit parameter field used by the MAPLE-B2 to set the doorbell transaction priority in WORD4[FLOWLVL] field in the Type10 Outbound Doorbell Descriptor in the eMSG unit. For details, see **Section 16.3.9.1,** *Type10 Outbound Doorbell Descriptor Format*.
- **MDGCP[RET]**—8 bit parameter which is used by the MAPLE-B2 for the WORD0[RETRY] field in the Type10 Outbound Doorbell Descriptor in the eMSG unit. For details, see **Section 16.3.9.1,** *Type10 Outbound Doorbell Descriptor Format*. The *RET* field determines the number of times the doorbell controller attempts to transmit the doorbell.

Figure 26-179 describes a possible multiple DSP device system with Serial RapidIO connectivity:

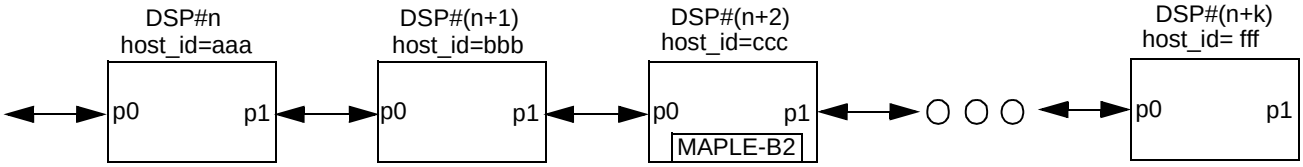


Figure 26-179. Possible Multiple DSP device Platform Connectivity

The example in **Figure 26-179** describes a possible platform with one DSP device containing an active MAPLE-B2 which is mastered by an external master located in any of the other DSP devices. If a master in DSP#n needs to master a BD ring in the MAPLE-B2 of DSP#(n+2), the values of the relevant parameters should be:

- *MBDR(H/L)PBxP[HOST_ID] = 0xaaa*. The host_id value of DSP#n
- *MBDR(H/L)PBxP[P_TR_IF] = 0x0*. The port target interface is RapidIO Port 1.
- *MBDR(H/L)PBxP[EXT_MST] = 0x1*. This parameter directs MAPLE-B2 to the door-bell controller of RapidIO Port 1 of the DSP.

If the MAPLE-B2 in Soc#(n+2) is not the only master for Doorbells, then:

- *HSP0BAP[HS_EN] = '1'*. This assures that whenever the MAPLE-B2 (or any other master in the DSP device) wishes to access the Doorbell controller, it first must lock the Semaphore, thus avoiding any coherency problem.
- *HSP0BSP[BA]* must be initialized with the address of the HS register to be used when trying to use the Doorbell.

- *MDHSIDCP[TV_HS_ID] = 0xA0*. User definition. Describes the MAPLE-B2's ID when accessing the Semaphore register for eTVPE rings.
- *MDHSIDCP[DF_HS_ID] = 0xA1*. User definition. Describes the MAPLE-B2's ID when accessing the Semaphore register for DFTPE rings.
- *MDHSIDCP[FF_HS_ID] = 0xA2*. User definition. Describes the MAPLE-B2's ID when accessing the Semaphore register for FFTPE rings.
- *MDGCP[BD_PR] =0x1*. Describes the transaction priority level. Written by MAPLE-B2 into WORD4[FLOWLVL] field in the Type10 Outbound Doorbell Descriptor.
- *MDGCP[RET] = 0x5*. User definition. Describes the number of times the message processor tries to re-send the Doorbell before asserting its interrupt. Written by MAPLE-B2 into the WORD0[RETRY] field in the Type10 Outbound Doorbell Descriptor.

Note: If the *HSP0BAP[HS_EN] = '0'*, that is, the MAPLE-B2 is the only doorbell master and therefore there is no need for external Semaphore, the MAPLE-B2 implements an internal semaphore to avoid contention between its internal RISC engines. This internal semaphore implementation requires no user configuration.

26.4.4.2 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. Place all input data related to the new jobs for the MAPLE-B2 in a system memory that is memory mapped with respect to the MAPLE-B2. The MAPLE-B2 can fetch its data only from its mapped system memory.
2. Place the new BDs in the external master's BD ring in MAPLE-B2 internal memory.

Upon job completion, the MAPLE-B2 outputs the relevant data to the system memory; and if the [INT_EN] bit in the BD is set, the MAPLE-B2 initiates a Doorbell interrupt. The following flow describes the MAPLE-B2's actual accesses on the MBus interface to issue a Doorbell interrupt and assumes the above parameters are configured:

- If *HSPxBAP[HS_EN]* is set:
  - Write access to the External HS register in address *HSPxBSP[BA]* and the data is *MDHSIDCP['pe'_HS_ID]* according to the PE.
  - Read access from the External HS reg in address *HSPxBSP[BA]*. If the read data equals *MDHSIDCP['pe'_HS_ID]*, then the Semaphore is locked for the PE, else MAPLE-B2 return to previous stage.
- Read access to Message Unit Status Register (MUSR). If the OMUB field is cleared, then the outbound message unit is not busy and the MAPLE-B2 can continue. If OMUB is set, MAPLE-B2 continues polling the field until it is cleared.

- Create the 32-byte outbound doorbell descriptor:
  - Set the descriptor type select field FTYPE=0b1010 to indicate doorbell type.
  - Set the retry field to indicate number or times the message unit should attempt to retry the doorbell before stopping.
  - Allocate buffer for the doorbell descriptor from one of the buffer pools available, preferable with the smallest buffer size (that is, 64-bytes).
  - For all other fields, see **Section 16.3.9.1,** *Type10 Outbound Doorbell Descriptor Format*
- Add the doorbell descriptor to a command descriptor and set FTYPE=0b1010 to indicate doorbell type.
- Enqueue the command descriptor onto the message queue.
- The message unit releases buffers used by the producer to hold the doorbell descriptor and data payload when the operation completes; if the buffer release bit (BR) is set in the message descriptor.

On doorbell operation completion, the doorbell controller interrupts the MAPLE-B2, which in turn, accesses the *MUSR* 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.
- The retry limit was exceeded: operation completed with error.

On doorbell completion interrupt assertion, the MAPLE-B2 performs the following:

1. Read access from the *MUSR*.
2. If *MUSR* status shows that the outbound message unit is idle (MUSR[OMUB]=0) and External HS is enabled (*HSPxBAP[HS_EN]* is set) then the MAPLE-B2 generates write access to External HS register with the *MDHSIDCP[‘pe’_HS_ID]* data to release the External HS.
3. If the MUIER[OTE] bit is set, the MAPLE-B2 asserts its general error interrupt and if the External HS is enabled (*HSPxBAP[HS_EN]* is set) then the MAPLE-B2 generates write access to External HS register with the *MDHSIDCP[‘pe’_HS_ID]* data to release the External HS.

Note: Generating Doorbell interrupts for every BD may cause a degradation in the MAPLE-B2 performance since it requires the MAPLE-B2 to configure a doorbell message Type10. If the controller is busy, the MAPLE-B2 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-B2, thus cannot be detected by it.

26.4.5 MAPLE-B2 Internal Task Control

The MAPLE-B2 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 MAPLE Timer Period Parameter (MP_TPP) and then starts a new search. There is no need for external intervention for the MAPLE-B2 to continuously process new BDs.

It is possible, though, to force the MAPLE-B2 to execute a certain routine. This is done using PSIF Command Register (PCR). By asserting the PCR[FLG] bit with the relevant PCR[opcode] field, the internal PSIF2 scheduler initiates a request to a RISC engine to execute one of the following routines:

- `MAPLE parse bd`. The MAPLE-B2 initiates the BD parse routines of all the PEs (eTVPE, FTPE_0, FTPE_1, FTPE_2, DEPE, EQPE, CONVPE and 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-B2 initiates the eTVPE BD parse routine, which scans the valid eTVPE BD Rings for a new valid BD to execute.
- `Maple_parse_eftpe_0_bd`. The MAPLE-B2 initiates the eFTPE_0 BD parse routine, which scans the valid eFTPE_0 BD Rings for a new valid BD to execute.
- `Maple_parse_eftpe_1_bd`. The MAPLE-B2 initiates the eFTPE_1 BD parse routine, which scans the valid eFTPE_1 BD Rings for a new valid BD to execute.
- `Maple_parse_eftpe_2_bd`. The MAPLE-B2 initiates the eFTPE_2 BD parse routine, which scans the valid eFTPE_2 BD Rings for a new valid BD to execute.
- `Maple_parse_depe_bd`. The MAPLE-B2 initiates the DEPE BD parse routine, which scans the valid DEPE BD Rings for a new valid BD to execute.
- `Maple_parse_crcpe_bd`. The MAPLE-B2 initiates the CRCPE BD parse routine, which scans the valid CRCPE BD Rings for a new valid BD to execute.
- `Maple_parse_eqpe_bd`. The MAPLE-B2 initiates the EQPE BD parse routine, which scans the valid EQPE BD Rings for a new valid BD to execute.
- `Maple_parse_convpe_bd`. The MAPLE-B2 initiates the CONVPE BD parse routine, which scans the valid CONVPE BD Rings for a new valid BD to execute.
- `Maple_parse_cgpe_bd`. The MAPLE-B2 initiates the CGPE BD parse routine, which scans the valid CGPE BD Rings for a new valid BD to execute.
- `Maple_reset_crpe`. The MAPLE-B2 initiates a reset sequence for all the CRPE related processing elements (CRPE-ULF, CRPE-DL and CRPE-ULB). For details see **Section 26.4.3.6.4**, *CRPE Reset*.

- `Maple_crpe_ulb_init`. The MAPLE-B2 initiates the CRPE-ULB to start with data processing after all related CRPE-ULB configuration is completed. For details see **Section 26.4.3.6.1.1.5**, *Maple_crpe_ulb_init Routine Activation*.
- `Maple_crpe_ulf_init`. The MAPLE-B2 initiates the first data fetch of the CRPE-ULF in case of CRPE-ULF Interpolation Bypass mode is enabled. For details see **Section 26.4.3.6.2.6.3**, *`Input Data interface—Interpolation Bypass Mode`*.
- `Maple_crpe_3x_eftpe_clock_gate_control`. The MAPLE-B2 disable/enable the main input clock of the CRPE or the 3 x eFTPE modules. For details see **Section 26.4.6.2**, *Per Processing Element Static Clock Gating Scheme*.
- `Maple_update_eftpe_pre_post_buffer`. The MAPLE-B2 update the eFTPEs pre/post multiplication buffers as per described in the relevant parameters of the eFTPEs. For details see **Section 26.4.3.3.5.3**, *'One-Shot' Initialization of the Pre/Post Multiplication buffers*.

26.4.6 MAPLE-B2 Power Gating Scheme

The MAPLE-B2 is designed for minimum power consumption while not in use. Each of the PEs in the MAPLE-B2 is designed to halt its internal clock toggling while in idle state, and to automatically re-toggle it when in use. The internal RISC engines in the PSIF2 are also programed to halt their activity whenever the 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 of the RISC engines once they are halted.

On top of the above automatic mechanisms, the following power control schemes exists:

26.4.6.1 Dynamic Power gating Scheme

Both the EQPE and the eTVPE include additional power saving scheme which allow the MAPLE-B2 to dynamically power off and power on the EQPE/eTVPE according to BD validation, that is, if no EQPE/eTVPE BD is available to execute, the MAPLE-B2 power off the EQPE/eTVPE thus saving much of its static power consumption. It is possible to control the EQPE/eTVPE power scheme using the [EQPE_PWR]/[ETVPE_PWR] fields of the MAPLE Mode Configuration 0 Parameter (MMC0P).

26.4.6.2 Per Processing Element Static Clock Gating Scheme

Both the CRPE and the 3 x eFTPE include additional static power saving scheme which allow gating the input clocks to these Processing elements. As opposed to the Dynamic power gating scheme, the clock gating scheme is not automatic and requires host activation to enable/disable the clock. Disabling the clock of the CRPE/3 x eFTPE should be done when the relevant PE is not required for long periods of time (for example, over night power-down) and is not targeted to be dynamically switched on/off between Buffer Descriptors processing.

Disabling/enabling of the clocks for the CRPE/3 x eFTPE is done using the following flow:

1. Stop activating the relevant PEs and assure it is idle before continuing to next steps.
2. Set the MCGCP field with the required operation (disable or enable the CRPE or 3 x eFTPE clock). See **Section 26.5.3.1.3**, *MAPLE Clock Gating Control Parameter (MCGCP)* for parameter description.
3. Initiate the `Maple_crpe_3x_eftpe_clock_gate_control` routine (see **Section 26.4.5**, *MAPLE-B2 Internal Task Control*), which performs the actual clock gating according to the configuration of the MCGCP.
4. On completion of the `Maple_crpe_3x_eftpe_clock_gate_control` routine, the MAPLE-B2 zeros the MCGCP parameter indicating the completion of the task.

Some notes regarding activating this flow:

- The 3 eFTPE units share the same clock source. Disabling this clock disables all 3 units.
- Activating the routine should be executed after the relevant PE has completed all operation. For 3 x eFTPEs it means all related BDs are completed. For CRPE it means disabling the CPRI and confirm no data transfer between CPRI and CRPE occur.
- Both, enabling and disabling a clock in the same host command results in clock disabled.

26.4.7 Reset

The MAPLE-B2 has two types of reset:

- *External Hard/Soft Reset*.On assertion of external reset, the MAPLE-B2 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 (*MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)) must be initiated to initialize the MAPLE-B2.
- *Internal Soft Reset*. It is possible to initiate an internal soft reset to the MAPLE-B2 using only the soft reset routine in the API. When called, it generates an internal soft reset to the whole MAPLE-B2 and initiate the MAPLE-B2 initialization sequence, which includes memories initialization and program upload to the MAPLE-B2 internal memory according to the chosen operation mode. For details on the API, see *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG).

26.5 Programming Model

The MAPLE-B2 includes the following functional memories/configurations areas:

- *PSIF2 DRAM*. Internal PSIF2 memory that includes the parameter RAM, the CRPE control parameters and the buffer descriptor rings RAM.
- *eTVPE memory space*. Includes internal eTVPE memory used by MAPLE-B2 to input data into the eTVPE to output data from the eTVPE, and registers configuration area used by the host and MAPLE-B2 for eTVPE configuration.
- *eFTPE memories space*. Include internal eFTPE memories used by MAPLE-B2 to input data into the 3 eFTPE modules and to output data from the eFTPE modules and registers configuration area used by the host and MAPLE-B2 for eFTPE modules configuration.
- *DEPE memory space*. Includes internal DEPE memory used by MAPLE-B2 to input data into the DEPE and to output data from the DEPE, and registers configuration area used by MAPLE-B2 for DEPE configuration.
- *EQPE memory space*. Includes internal EQPE memory used by MAPLE-B2 to input data into the EQPE and to output data from the EQPE, and registers configuration area for EQPE configuration.
- *CRPE memory space*. Includes internal CRPE memory used by MAPLE-B2 (or external host) to input data into the CRPE and to output data from the CRPE, and registers configuration area used by MAPLE-B2 and external host for CRPE control and configuration.

The MAPLE-B2 contains both registers and a parameter RAM. The registers and parameters are treated similarly. The only difference is that the default values for the registers are the “Reset” values whereas the default values for parameters are the “Initialization” values. The default values for parameters apply only after the MAPLE-B2 initialization sequence is completed using its API (see *MAPLE-B2 Application Programmer Interface (API) User’s Guide* (MAPLEAPIUG)).

All registers/parameters are described as REG/PARAM_NAME<letter> and offset value. 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 x∗0x8) means that the following registers are implemented:

- MTVBRHPA0P at offset 0x00000100
- MTVBRHPA1P at offset 0x00000108
- ...
- MBDRHPA7P at offset 0x00000138

Each of these memories/registers are accessed by the MBus Slave0 interface. MBus Slave1 interface is used for data transfer from Antenna I/F and into MAPLE-B2 memories only.

Figure 26-180 describes the internal MAPLE-B2 MBus slaves memory map partitioning.

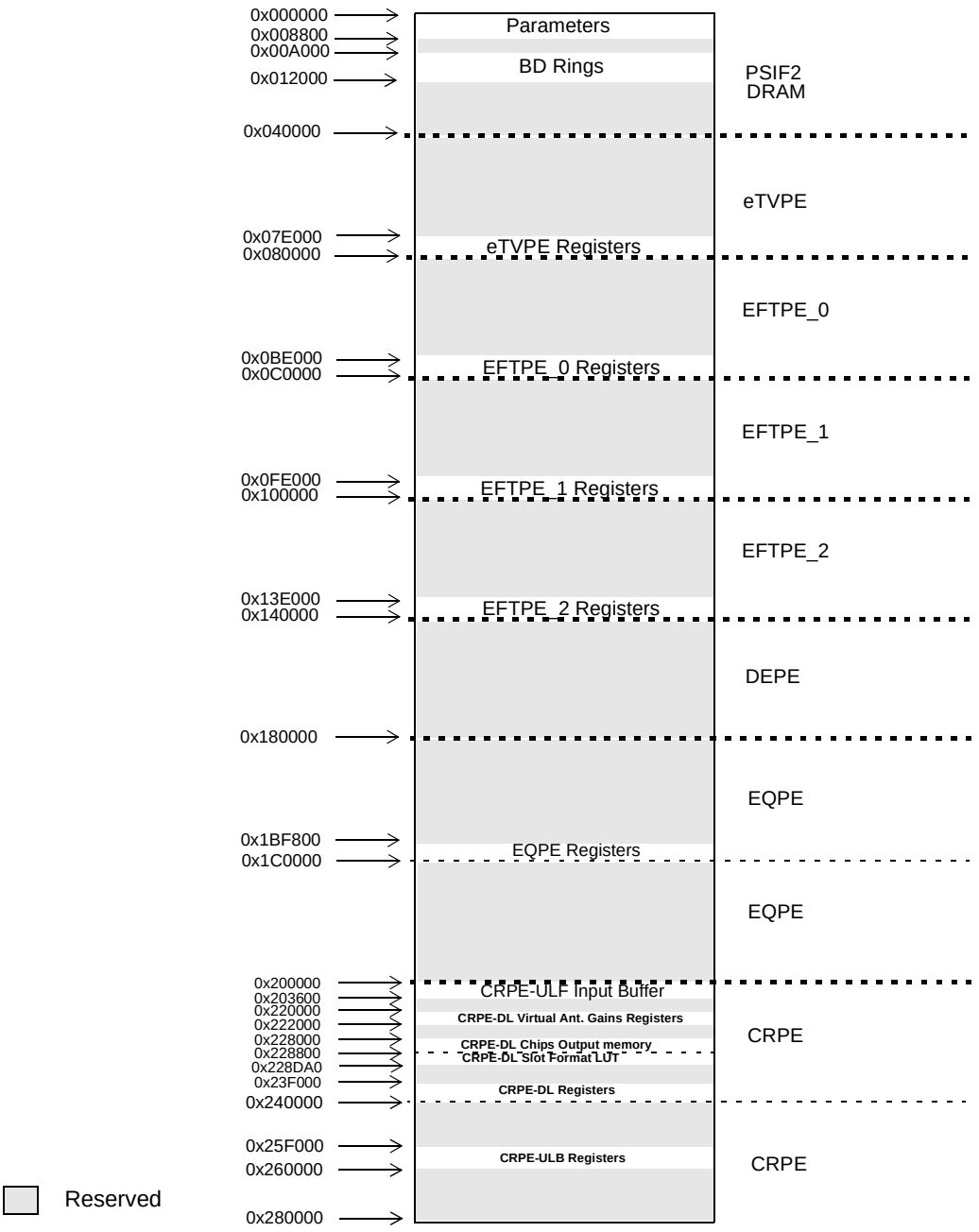


Figure 26-180. MBus Memory Map Partitioning

Note: Accessing Reserved areas may result in MAPLE-B2 entering an unknown state that may require a hard reset of the whole device!

Table 26-125 describes the MBus memory map in detail. Each of the memories/registers are accessed by the MBus Slave0 interface. MBus Slave1 interface is used for data transfer from Antenna interface and into MAPLE-B2 memories only.

Table 26-125. MAPLE-B2 MBus Memory Map

Address Range[1]	Description	Size (KB)
PSIF2 DRAM		
0x00000000–0x0000004FF	General Parameters RAM. For detailed map, see **Table 26-132**.	1.25
0x00000500–0x0000005FF	eTVPE parameters. For detailed map, see **Table 26-136**	0.25
0x00000600–0x0000006BF	eFTPE parameters. For detailed map, see **Table 26-142**	0.125
0x000006C0–0x0000006FF	EQPE parameters. For detailed map, see **Table 26-150**	0.125
0x00000700–0x0000007FF	sRIO Doorbell related parameters. For detailed map, see TBD	0.25
0x00000800–0x00002FFF	CRPE-ULB Parameters. For detailed map, see **Table 26-153**	12
0x00003800–0x00003BFF	CRPE-ULF Parameters. For detailed map, see **Table 26-166**	1
0x00003C00–0x00008FFF	CRPE-DL Parameters. For detailed map, see **Table 26-174**	21
0x00009000–0x00009FFF	Reserved	4
0x0000A000–0x0011FFF	Buffer Descriptor Rings	32
0x00012000–0x0003FFFF	Reserved.	184
eTVPE Memory Map		
0x00040000–0x0007DFFF	Reserved	248
0x0007E000–0x0007FFFF	eTVPE Registers. For details, see **Table 26-329**	8
eFTPE_0 Memory Map		
0x00080000–0x000BDFFF	Reserved.	248
0x000BE000–0x000BFFFF	eFTPE_0 registers. For detailed map, see **Table 26-332**.	8
eFTPE_1 Memory Map		
0x000C0000–0x000FDFFF	Reserved.	248
0x000FE000–0x000FFFFF	eFTPE_1 registers. For detailed map, see **Table 26-332**.	8
eFTPE_2 Memory Map		
0x00100000–0x0013DFFF	Reserved.	248
0x0013E000–0x0013FFFF	eFTPE_2 registers. For detailed map, see **Table 26-332**.	8
DEPE Memory Map		
0x00140000–0x0017FFFF	Reserved.	256
EQPE Memory Map		
0x00180000–0x001CFA07	Reserved.	254
0x001CFA08–0x001CFB0F	EQPE Registers. For details see **Table 26-338**	2
0x001CFB10–0x001FFFFF	Reserved	256
CRPE-ULF Memory Map		
0x00200000–0x002035FF	CRPE-ULF Input Buffer	13.5
0x00203600–0x00217FFF	Reserved	82.5
0x0023E000–0x0023EFFF	CRPE-ULF Registers. For details see **Table 26-349**	4
CRPE-DL Memory Map		
0x00218000–0x0021FFFF	Reserved	32
0x00220000–0x00221FFF	CRPE-DL Virtual Antenna Gains Control Registers. For details see **Table 26-360**.	8
0x00222000–0x00227FFF	Reserved	24
0x00228000–0x002287FF	CRPE-DL Output Chip Data Memory. For details see **Table 26-360**	2
0x00228800–0x00228D9F	CRPE-DL Slot Format LUT. For details see **Table 26-360**.	1.4
0x00228DA0–0x0023DFFF	Reserved	84.6
0x0023F000–0x0023F7FF	CRPE-DL Registers. For details see **Table 26-360**.	2
0x0023F800–0x0023F9FF	CRPE Scrambling Initialization LUT. For details see **Table 26-360**	0.5
0x0023FA00–0x0023FFFF	Reserved	1.5
CRPE-ULB Memory Map		
0x00240000–0x0025EFFF	Reserved.	124
0x0025F000–0x0025FFFF	CRPE-ULB Registers. For details see **Table 26-341**	4

1. The specified addresses are relative to MAPLE-B2 base_address, that is, for absolute address calculate MAPLE-B2_base + Address Range.

The SBus interface of the MAPLE-B2 is used to access the PSIF2 internal registers only. These registers use the MAPLE-B2 CCSR base address 0xFFF1C000.

Table 26-126 describes the SBus Memory Map.

Table 26-126. MAPLE-B2 SBus Memory Map

Address	Register	Access	Reset Value	Section
PSIF2 REGISTERS				
0x00000000–0x00000FFF	Reserved			
0x00001000	PCR - PSIF Command Register	R/W	0x00000000	**Section 26.5.5.1.1**
PSIC Programmable Interrupt Controller Registers				
0x00001600	PSPICER0. PSIF PIC Event Register 0	R/(W ‘1’ to clear)	0x00000000	**Section 26.5.5.1.2**
0x00001604	PSPICER1. PSIF PIC Event Register 1	R/(W ‘1’ to clear)	0x00000000	**Section 26.5.5.1.3**
0x00001608	PSPICER2. PSIF PIC Event Register 2	R/(W ‘1’ to clear)	0x00000000	**Section 26.5.5.1.4**
0x0000160C	PSPICELR. PSIF PIC Edge/Level Register	R/W	0x00000000	**Section 26.5.5.1.5**
0x00001610	PSPICMR0. PSIF PIC Mask Register 0	R/W	0x00000000	**Section 26.5.5.1.6**
0x00001614	PSPICMR1. PSIF PIC Mask Register 1	R/W	0x00000000	**Section 26.5.5.1.7**
0x00001618	PSPICMR2. PSIF PIC Mask Register 2	R/W	0x00000000	**Section 26.5.5.1.8**
0x0000161C	PSPICIACR. PSIF PIC Interrupt Assertion Clocks Registers	R/W	0x00000000	**Section 26.5.5.1.9**

26.5.1 MAPLE-B2 Programming

The MAPLE-B2 requires several levels of programming, including the following:

- Parameter initialization, which is done using the MAPLE-B2 application programmer interface (API). See **Section 26.5.2, *Initialization Parameters*** for details.
- Operational configuration of functional memory areas including the following:
  - Parameter RAM (includes control parameters for CRPE only)
  - RAM-based buffer descriptor rings, buffer descriptors, and task descriptors.
- PSIF2 and PE register configuration.

26.5.1.1 General Programming Guidelines

Following are few general guidelines which should be maintain while programming MAPLE-B2:

a. Reserved bits in parameters/registers must be written with the value of ‘0’ for future compatibility.

b. When a functional field is valid in a certain configuration/mode only, for all other configurations it should be written with the value of ‘0’ to prevent any miss-programming.

26.5.2 Initialization Parameters

There are five sets of configuration parameters (two system level and three PE-specific) that must be initialized by the MAPLE-B2 API prior to use. These parameters are the following:

- MAPLE Mode Configuration 0 and 1 Parameters (MMC[0–1]P)—General MAPLE-B2 initialization parameters. See **Section 26.5.2.1** and **Section 26.5.2.2**.
- eTVPE Mode Configuration Parameter (TVMCP)—General eTVPE related initialization parameters. See **Section 26.5.2.3**
- CRPE Uplink Batch Mode Configuration Parameter (CRUBMCP)—CRPE-ULB General configuration parameters. See **Section 26.5.2.5**.
- CRPE Down Link Output Mode Configuration Parameter (CDOMCP)—CRPE-DL Output mode related initialization parameters. See **Section 26.5.2.4**.

The following subsections include detailed descriptions of these parameters.

26.5.2.1 MAPLE Mode Configuration 0 Parameter (MMC0P)

The MMC0P parameter includes general parameters, to be initialized during the MAPLE-B2 API initialization. Following is the MMC0P fields description:

Offset	0x00000020 (MMC0P)											Access:		Read only		
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
				MB_ALC			MB_PR_SCH		MB3_DF_PR		MB2_DF_PR		MB1_DF_PR		MB0_DF_PR	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			EQPE_PWR				ETVPE_PWR					ECC_EN				STD

Figure 26-181. MMC0P parameter structure

Table 26-127. MMC0P Fields Description

Bits	Description
31 – 29	Reserved.
28	MB_ALC — MBus Allocation. Indicates whether the system MBus initiators interfaces allocation to each of the Processing Elements (PEs) is done internally by MAPLE-B2 or determined by the MMC1P parameter. For details, refer to **Section 26.5.2.2**, *MAPLE Mode Configuration 1 Parameter (MMC1P)*. 0 — PEs allocation to each of the MBus interfaces is done internally by MAPLE-B2 1 — PEs allocation to each of the MBus interfaces is done according to MMC1P configuration.
27 – 26	Reserved.
25 – 24	**MB_PR_SCH — MBus Priority Scheme**. Indicates the MAPLE-B2 which MBus initiators priority scheme is applied. For details see **Section 26.4.2.4**, *MBus Priority Scheme Configuration* 00— All MAPLE-B2 MBus master accesses are initiated with fixed priority level as described in the [MB0_DF_PR], [MB1_DF_PR], [MB2_DF_PR] and [MB1_DF_PR] fields for MBus0, MBus1, MBus2 and MBus3 respectively. 01—All MAPLE-B2 MBus master accesses related to a certain BD are initiated with priority as described in the [MB_PR] field of the BD. Previous DMA commands already launched into the relevant DMA queue are not upgraded with their priority accordingly. 10—All MAPLE-B2 MBus master accesses related to a certain BD are initiated with priority as described in the [MB_PR] field of the BD. All DMA commands already launched into the relevant DMA queue are upgraded with their priority accordingly. 11—Reserved.
23 – 22	**MB3_DF_PR — MBus3 Default Priority**. Valid only if the [MB_PR_SCH] is set to ‘00’. Describe the default priority of the MBus3 master. For details see **Section 26.4.2.4**, *MBus Priority Scheme Configuration* 00— The default MBus3 priority is ‘00’ (lowest priority) ... 11— The default MBus3 priority is ‘11’ (highest priority)
21 – 20	**MB2_DF_PR — MBus2 Default Priority**. Valid only if the [MB_PR_SCH] is set to ‘00’. Describe the default priority of the MBus2 master. For details see **Section 26.4.2.4**, *MBus Priority Scheme Configuration* 00— The default MBus2 priority is ‘00’ (lowest priority) ... 11— The default MBus2 priority is ‘11’ (highest priority)
19 – 18	**MB1_DF_PR — MBus1 Default Priority**. Valid only if the [MB_PR_SCH] is set to ‘00’. Describe the default priority of the MBus1 master. For details see **Section 26.4.2.4**, *MBus Priority Scheme Configuration* 00— The default MBus1 priority is ‘00’ (lowest priority) ... 11— The default MBus1 priority is ‘11’ (highest priority)
17 – 16	**MB0_DF_PR — MBus0 Default Priority**. Valid only if the [MB_PR_SCH] is set to ‘00’. Describe the default priority of the MBus0 master. For details see **Section 26.4.2.4**, *MBus Priority Scheme Configuration* 00— The default MBus0 priority is ‘00’ (lowest priority) ... 11— The default MBus0 priority is ‘11’ (highest priority)
15 – 14	Reserved.

Table 26-127. MMC0P Fields Description

Bits	Description
13 – 12	**EQPE_PWR — EQPE Power control Mode**. Determined the power scheme used for the EQPE module. For details, refer to **Section 26.4.6**, *MAPLE-B2 Power Gating Scheme*. 00—EQPE Automatic power switching mode. The MAPLE-B2 power off the EQPE when there is no available EQPE BD. 01— EQPE power on. The EQPE always powered on. No power switching 10— EQPE power off. The EQPE is powered off. 11— Reserved.
11 – 10	Reserved.
9 – 8	**ETVPE_PWR — eTVPE Power control Mode**. Determined the power scheme used for the eTVPE module. For details, refer to **Section 26.4.6**, *MAPLE-B2 Power Gating Scheme*. 00—eTVPE Automatic power switching mode. The MAPLE-B2 power off the eTVPE when there is no available eTVPE BD. 01— eTVPE power on. The eTVPE always powered on. No power switching 10— eTVPE power off. The eTVPE is powered off. 11— Reserved.
7 – 5	Reserved.
4	**ECC_EN**. ECC memory protection Enable indication. If set the MAPLE-B2 enables ECC protection to all its internal memories which support ECC protection. 0 — ECC is disabled. 1 — ECC is enabled.
3 – 1	Reserved.
0	**STD**. MAPLE Standard operation mode. Determines which technology is currently supported in MAPLE-B2 0 — 3G Operation mode. Supporting both UMTS and 3GLTE technologies 1 — WiMAX Operation mode. Supporting both WiMAX and 3GLTE technologies.

26.5.2.2 MAPLE Mode Configuration 1 Parameter (MMC1P)

The MMC1P parameter includes general MBus related parameters, to be initialized during the MAPLE-B2 API initialization. All fields in this parameter are valid only if the [MB_ALC] field of the MMC0P parameter is set.

Note: When assigning a certain MBus engine to a Read/Write operation per PE, it may happen that some minor Write/Read operations occurs on that MBus engine as well.

Following is the MMC1P fields description:

Offset	0x00000024 (MMC1P)											Access:		Read only		
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TV_W_B		TV_R_B		FT0_W_B		FT0_R_B		FT1_W_B		FT1_R_B		FT2_W_B		FT2_R_B	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DE_W_B		DE_R_B		EQ_W_B		EQ_R_B		CR_W_B		CR_R_B		CRCG_W_B		CRC_R_B	

Figure 26-182. MMC1P parameter structure

Table 26-128. MMC1P parameter Fields Description

Bits	Description
31 – 30	**TV_W_B — eTVPE Write Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the eTVPE write its results to system memory 00 — eTVPE results are written using MBus 0 01 — eTVPE results are written using MBus 1 10 — eTVPE results are written using MBus 2 11 — eTVPE results are written using MBus 3
29 – 28	**TV_R_B — eTVPE Read Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the eTVPE read its input data from system memory 00 — eTVPE input data is read using MBus 0 01 — eTVPE input data is read using MBus 1 10 — eTVPE input data is read using MBus 2 11 — eTVPE input data is read using MBus 3
27 – 26	**FT0_W_B — EFTPE_0 Write Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the EFTPE_0 write its results to system memory 00 — EFTPE_0 results are written using MBus 0 01 — EFTPE_0 results are written using MBus 1 10 — EFTPE_0 results are written using MBus 2 11 — EFTPE_0 results are written using MBus 3
25 – 24	**FT0_R_B — EFTPE_0 Read Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the EFTPE_0 read its input data from system memory 00 — EFTPE_0 input data is read using MBus 0 01 — EFTPE_0 input data is read using MBus 1 10 — EFTPE_0 input data is read using MBus 2 11 — EFTPE_0 input data is read using MBus 3
23 – 22	**FT1_W_B — EFTPE_1 Write Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the EFTPE_1 write its results to system memory 00 — EFTPE_1 results are written using MBus 0 01 — EFTPE_1 results are written using MBus 1 10 — EFTPE_1 results are written using MBus 2 11 — EFTPE_1 results are written using MBus 3
21 – 20	**FT1_R_B — EFTPE_1 Read Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the EFTPE_1 read its input data from system memory 00 — EFTPE_1 input data is read using MBus 0 01 — EFTPE_1 input data is read using MBus 1 10 — EFTPE_1 input data is read using MBus 2 11 — EFTPE_1 input data is read using MBus 3
19 – 18	**FT2_W_B — EFTPE_2 Write Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the EFTPE_2 write its results to system memory 00 — EFTPE_2 results are written using MBus 0 01 — EFTPE_2 results are written using MBus 1 10 — EFTPE_2 results are written using MBus 2 11 — EFTPE_2 results are written using MBus 3
17 – 16	**FT2_R_B — EFTPE_2 Read Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the EFTPE_2 read its input data from system memory 00 — EFTPE_2 input data is read using MBus 0 01 — EFTPE_2 input data is read using MBus 1 10 — EFTPE_2 input data is read using MBus 2 11 — EFTPE_2 input data is read using MBus 3

Table 26-128. MMC1P parameter Fields Description

Bits	Description
15 – 14	**DE_W_B — DEPE Write Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the DEPE write its results to system memory 00 — DEPE results are written using MBus 0 01 — DEPE results are written using MBus 1 10 — DEPE results are written using MBus 2 11 — DEPE results are written using MBus 3
13 – 12	**DE_R_B — DEPE Read Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the DEPE read its input data from system memory 00 — DEPE input data is read using MBus 0 01 — DEPE input data is read using MBus 1 10 — DEPE input data is read using MBus 2 11 — DEPE input data is read using MBus 3
11 – 10	**EQ_W_B — EQPE Write Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the EQPE write its results to system memory 00 — EQPE results are written using MBus 0 01 — EQPE results are written using MBus 1 10 — EQPE results are written using MBus 2 11 — EQPE results are written using MBus 3
9 – 8	**EQ_R_B — EQPE Read Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the EQPE read its input data from system memory 00 — EQPE input data is read using MBus 0 01 — EQPE input data is read using MBus 1 10 — EQPE input data is read using MBus 2 11 — EQPE input data is read using MBus 3
7 – 6	**CR_W_B — CRPE Write Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the CRPE write its results to system memory 00 — CRPE results are written using MBus 0 01 — CRPE results are written using MBus 1 10 — CRPE results are written using MBus 2 11 — CRPE results are written using MBus 3
5 – 4	**CR_R_B — CRPE Read Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the CRPE read its input data from system memory 00 — CRPE input data is read using MBus 0 01 — CRPE input data is read using MBus 1 10 — CRPE input data is read using MBus 2 11 — CRPE input data is read using MBus 3
3 – 2	**CRCG_W_B — CRCPE/CGPE[1] Write Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the CRCPE and the CGPE write their results to system memory 00 — CRCPE and CGPE results are written using MBus 0 01 — CRCPE and CGPE results are written using MBus 1 10 — CRCPE and CGPE results are written using MBus 2 11 — CRCPE and CGPE results are written using MBus 3
1 – 0	**CRC_R_B — CRCPE Read Bus.** If MMC0P[MB_ALC] bit is set, indicates on which of the 4 MBus initiators the CRCPE read its input data from system memory 00 — CRCPE input data is read using MBus 0 01 — CRCPE input data is read using MBus 1 10 — CRCPE input data is read using MBus 2 11 — CRCPE input data is read using MBus 3

1. The CGPE only output data and does not require input data. That is the reason it is sharing only the write bus with the CRCPE.

26.5.2.3 MAPLE eTVPE Configuration Parameter (MTVCP)

If the eTVPE is used by the application, then during the MAPLE-B2 initialization (see *MAPLE-B Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)), the eTVPE configuration parameter in the API must be initialized:

Offset	0x00000028 (MTVCP)											Access:		Read only		
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								SCRC	RANE		SEP_V	HE				
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			ANDRE						CRC_AUTOSTOP	AQC_AUTOSTOP		POL				

Figure 26-183. MAPLE eTVPE Configuration Parameter

Table 26-129. MAPLE eTVPE Configuration Fields Description

Bits	Description
24	**SCRC - Steady CRC**. Steady CRC stop criteria indication. The eTVPE stop the decoding automatically once CRC result is equal to CRC results from previous iteration. Must not be set if [CRC_AUTOSTOP] is set. For details, see **Section 26.4.3.2.6.4.4**, *Turbo Stopping Criteria Configurations*, on page 26-55 0—Steady CRC stop criteria is disabled. 1—Steady CRC stop criteria is enabled.
23	**RANE - De-Randomizer Enable**. Valid only for MAPLE-B2 WiMAX operation mode. If enabled the eTVPE performs de-randomization on the hard outputs. 0—De-randomizer is disabled. 1—De-randomizer is enabled.
21	**SEP_V**. Separate Vectors indication. Valid only for MAPLE-B2 3G operation mode. Indicated which rate de-matching scheme is used for UMTS jobs 0—Single Mixed Vector rate de-matching scheme is applied. 1—Separate Vectors rate de-matching scheme is applied.
20	**HE - HARQ Enable**. Indicated whether HARQ operation in eTVPE is enabled, hence which input data structures for the eTVPE are allowed. For details, refer to **Section 26.4.3.2.5**, *eTVPE Input Data Structures*, on page 26-30. 0—HARQ processing is disabled. 1—HARQ processing is enabled.
13	**ANDRE - Automatic NDRE**. If enabled the eTVPE internally sets the number of Turbo engines (DRE) to achieve maximum performance. 0—Number of Turbo engines is defined by host using NDRE field of eTVPE BD 1—Number of Turbo engines is set internally by eTVPE.
7	**CRC_AUTOSTOP**. CRC check stop criteria indication. The eTVPE stop the decoding automatically once the Hard Output bits pass CRC check. Must not be set if [SCRC] is set. For details, refer to **Section 26.4.3.2.6.4.4**, *Turbo Stopping Criteria Configurations*, on page 26-55. 0—CRC check stop criteria is disabled. 1—CRC check stop criteria is enabled.

Table 26-129. MAPLE eTVPE Configuration Fields Description

Bits	Description
6	**AQC_AUTOSTOP**. Aposteriori Quality stop criteria indication. the eTVPE stop the decoding automatically once the all the soft-decoded bits pass the [AQTH] threshold field. For details, refer to **Section 26.4.3.2.6.4.4**, *Turbo Stopping Criteria Configurations*, on page 26-55. 0—AQC stop criteria is disabled. 1—AQC stop criteria is enabled.
4	**POL - Polarity**. Describe the eTVPE input data polarity: 0—Logic 0 is mapped as (- 1); Logic 1 is mapped as (+1); 1—Logic 0 is mapped as (+1); Logic 1 is mapped as (- 1);

26.5.2.4 CRPE-DL Output Mode Configuration Parameter (CDOMCP)

If the application uses chip rate download processing, then during the MAPLE-B2 initialization (see *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)), the following CRPE-DL related parameter in the API must be initialized:

Offset	0x0000002C (CDOMCP)											Access:		Read only		
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	A15EN	A14EN	A13EN	A12EN	A11EN	A10EN	A9EN	A8EN	A7EN	A6EN	A5EN	A4EN	A3EN	A2EN	A1EN	A0EN
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														RCS	RSEL	MOM

Figure 26-184. CRPE-DL Output Mode Configuration Parameter

Table 26-130. CRPE-DL Output Mode Configuration Fields Description

Bits	Description
31	**A15EN - Output Antenna 15 Enable** 0 - Output antenna 15 is disabled. 1 - Output antenna 15 is enabled.
30	**A14EN - Output Antenna 14 Enable** 0 - Output antenna 14 is disabled. 1 - Output antenna 14 is enabled.
29	**A13EN - Output Antenna 13 Enable** 0 - Output antenna 13 is disabled. 1 - Output antenna 13 is enabled.
28	**A12EN - Output Antenna 12 Enable** 0 - Output antenna 12 is disabled. 1 - Output antenna 12 is enabled.
27	**A11EN - Output Antenna 11 Enable** 0 - Output antenna 11 is disabled. 1 - Output antenna 11 is enabled.
26	**A10EN - Output Antenna 10 Enable** 0 - Output antenna 10 is disabled. 1 - Output antenna 10 is enabled.

Table 26-130. CRPE-DL Output Mode Configuration Fields Description

Bits	Description
25	**A9EN - Output Antenna 9 Enable** 0 - Output antenna 9 is disabled. 1 - Output antenna 9 is enabled.
24	**A8EN - Output Antenna 8 Enable** 0 - Output antenna 8 is disabled. 1 - Output antenna 8 is enabled.
23	**A7EN - Output Antenna 7 Enable** 0 - Output antenna 7 is disabled. 1 - Output antenna 7 is enabled.
22	**A6EN - Output Antenna 6 Enable** 0 - Output antenna 6 is disabled. 1 - Output antenna 6 is enabled.
21	**A5EN - Output Antenna 5 Enable** 0 - Output antenna 5 is disabled. 1 - Output antenna 5 is enabled.
20	**A4EN - Output Antenna 4 Enable** 0 - Output antenna 4 is disabled. 1 - Output antenna 4 is enabled.
19	**A3EN - Output Antenna 3 Enable** 0 - Output antenna 3 is disabled. 1 - Output antenna 3 is enabled.
18	**A2EN - Output Antenna 2 Enable** 0 - Output antenna 2 is disabled. 1 - Output antenna 2 is enabled.
17	**A1EN - Output Antenna 1 Enable** 0 - Output antenna 1 is disabled. 1 - Output antenna 1 is enabled.
16	**A0EN - Output Antenna 0 Enable** 0 - Output antenna 0 is disabled. 1 - Output antenna 0 is enabled.
2	**RCS - Rate Control Source** This field selects the source of CRPE-DL rate control event to be either interrupt or write access to CRPE Downlink Rate Control Register. 0 - The source of CRPE-DL rate control event is an interrupt. 1 - The source of CRPE-DL rate control event is a write accesses to CRPE Downlink Rate Control Register.
1	**RSEL - Rate Selector** This field selects CRPE-DL rate control event to be either new 16 chips chunk event or new 256 chips sub-slot event. 0 - CRPE-DL rate control event is new 16 chips chunk event. 1 - CRPE-DL rate control event is new sub-slot event.
0	**MOM - MAPLE Output Mode** 0 - CRPE-DL is operating at CPRI output mode. 1 - CRPE-DL is operating at MAPLE output mode.

26.5.2.5 CRPE-ULB Mode Configuration Parameter (CRUBMCP)

If the application uses the CRPE Uplink Batch operation, then during the MAPLE-B2 initialization (see *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)), the following CRPE-ULB related parameter in the API must be initialized:

Offset	0x00000030 (CRUBMCP)											Access:		Read only		
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								DBL_ANT_EN					ISHA			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					DDS				DLY_SPRD	OOB		FC_MODE				INT_MODE

Figure 26-185. CRPE-ULB Mode Configuration Parameter

Table 26-131. CRPE-ULB Mode Configuration Fields Description

Bits	Description
31–25	Reserved
24	DBL_ANT_EN - Double Antenna Enable 0—Up to 48 Antennas are allowed; Maximum Interpolation of x16 1—Up to 96 Antennas are allowed; Maximum Interpolation of x8
23–20	Reserved
19–16	ISHA - Interpolation shift amount Contains the amount by which the sum of the interpolated (weight x chips) is shifted right before the result is truncated from 19bit to 8bit. For details see **Section 26.4.3.6.1.8.6**, *Interpolation Saturation During Internal Interpolation*. Valid values: 0 to15
15–12	Reserved
11	DDS - DD Saturation Valid if *FC_MODE* = 0 only; Selects whether to shift the result left by '1' or not 0—Shift left is disabled (output lower 16bits). Optional saturation. 1—Shift left is enabled (output higher 16bits). No saturation occurs.
10–8	Reserved
7	DLY_SPRD - Delay spread mode 0—Delay spread of up to 256 (*FOFFSET_H* of the Finger command allowed values of 0 to 511) 1—Delay spread of up to 512 (*FOFFSET_H* of the Finger command allowed values of 0 to 767)
6	OOB - One Output buffer Indicates if the CRPE-ULB functions with single Output Buffer or double Output buffer. for details see **Section 26.4.3.6.1.13**, *Output Buffer Capacity*. 0—Disabled; Output buffers 1 and 2 are double buffer. 1—Enabled; Output buffers 1 and 2 are concatenated into a single buffer.
5	Reserved
4–3	FC_MODE - Fingers Combine mode 0—FC bypass; fingers are output directly to output memory 1—FC with 16bit fixed point output per symbol, fixed scale, per the PCH command *EXP* field. 2—Reserved. 3—FC with 16bit floating point + 16bit exponent.

Table 26-131. CRPE-ULB Mode Configuration Fields Description

Bits	Description
2–1	Reserved
0	INT_MODE - Interpolation mode 0—Interpolation bypass 1—Internal interpolation enabled

26.5.3 Parameter RAM

The following sections describe the sets of parameter RAM.

26.5.3.1 General Parameter RAM Description

Table 26-132 describes the MAPLE-B2 general Parameter RAM in detail.

Table 26-132. Parameter RAM Detailed Description

Address	Parameters	Access	Initialization Value[1]	Section
0x00000000	Reserved			
0x00000004	MBDRCP0. MAPLE BD Rings Configuration Parameter 0	R/W	0x00000000	**Section 26.5.4.1.1**
0x00000008	MBDRCP1. MAPLE BD Rings Configuration Parameter 1	R/W	0x00000000	**Section 26.5.4.1.2**
0x0000000C	MBDRCP2. MAPLE BD Rings Configuration Parameter 2	R/W	0x00000000	**Section 26.5.4.1.3**
0x00000010	MUCVP. MAPLE UCode Version Parameter	R	ucode version	**Section 26.5.3.1.1**
0x00000014	Reserved.			
0x00000018	MP_TPP. MAPLE Timer Period Parameter	R/W	Set by API	**Section 26.5.3.1.2**
0x0000001C	MCGCP. MAPLE Clock Gating Control Parameter	R/W	0x00000000	**Section 26.5.3.1.3**
0x00000020	MMC0P. MAPLE Mode Configurations 0 Parameter.	R	Set by API	**Section 26.5.2.1**
0x00000024	MMC1P. MAPLE Mode Configurations 1 Parameter.	R	Set by API	**Section 26.5.2.2**
0x00000028	MTVCP. MAPLE eTVPE Configuration parameter	R	Set by API	**Section 26.5.2.3**
0x0000002C	CDOMCP. CRPE-DL Output Mode Configuration Parameter.	R	Set by API	**Section 26.5.2.4**
0x00000030	CUBMCP. CRPE-ULB Mode Configuration Parameter.	R	Set by API	**Section 26.5.2.5**
0x00000034–0x0000003F	Reserved			
0x00000040	MCRRCIP. MAPLE CRPE Reset Completion Indication Parameter	R	0x00000000	**Section 26.5.3.5.1.1**
0x00000044–0x0000007F	Reserved			

Table 26-132. Parameter RAM Detailed Description (Continued)

Address	Parameters	Access	Initialization Value[1]	Section
0x00000080–0x000000BF	MTVBRHPAxP. MAPLE eTVPE BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MTVBRHPBxP. MAPLE eTVPE BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x000000C0–0x000000FF	MTVBRLPAxP. MAPLE eTVPE BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MTVBRLPBxP. MAPLE eTVPE BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000100–0x0000013F	MF0BRHPAxP. MAPLE EFTPE_0 BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MF0BRHPBxP. MAPLE EFTPE_0 BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x00000140–0x0000017F	MF0BRLPAxP. MAPLE EFTPE_0 BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MF0BRLPBxP. MAPLE EFTPE_0 BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000180–0x000001BF	MF1BRHPAxP. MAPLE EFTPE_1 BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MF1BRHPBxP. MAPLE EFTPE_1 BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x000001C0–0x000001FF	MF1BRLPAxP. MAPLE EFTPE_1 BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MF1BRLPBxP. MAPLE EFTPE_1 BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000200–0x0000023F	MF2BRHPAxP. MAPLE EFTPE_2 BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MF2BRHPBxP. MAPLE EFTPE_2 BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x00000240–0x0000027F	MF2BRLPAxP. MAPLE EFTPE_2 BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MF2BRLPBxP. MAPLE EFTPE_2 BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000280–0x000002BF	MDEBRHPAxP. MAPLE DEPE BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MDEBRHPBxP. MAPLE DEPE BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x000002C0–0x000002FF	MDEBRLPAxP. MAPLE DEPE BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MDEBRLPBxP. MAPLE DEPE BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000300–0x0000033F	MCRCBRHPAxP. MAPLE CRCPE BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MCRCBRHPBxP. MAPLE CRCPE BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x00000340–0x0000037F	MCRCBRLPAxP. MAPLE CRCPE BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MCRCBRLPBxP. MAPLE CRCPE BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**

Table 26-132. Parameter RAM Detailed Description (Continued)

Address	Parameters	Access	Initialization Value[1]	Section
0x00000380–0x000003BF	MEQBRHPAxP. MAPLE EQPE BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MEQBRHPBxP. MAPLE EQPE BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x000003C0–0x000003FF	MEQBRLPAxP. MAPLE EQPE BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MEQBRLPBxP. MAPLE EQPE BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000400–0x0000043F	MCGBRHPAxP. MAPLE CGPE BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MCGBRHPBxP. MAPLE CGPE BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x00000440–0x0000047F	MCGBRLPAxP. MAPLE CGPE BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MCGBRLPBxP. MAPLE CGPE BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000480–0x000004BF	MCONVBRHPAxP. MAPLE CONVPE BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MCONVBRHPBxP. MAPLE CONVPE BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x000004C0–0x000004FF	MCONVBRLPAxP. MAPLE CONVPE BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MCONVBRLPBxP. MAPLE CONVPE BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**

1. All the parameters in the parameter RAM are initialized during the MAPLE-B2 API initialization sequence, and are not hardware reset values. See the *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)

26.5.3.1.1 MAPLE UCode Version Parameter (MUCVP)

This parameter includes the internal risc engines firmware version. It is written during the MAPLE-B2 activation using its API.

Offset 0x00000010 (MUCVP) Access: User read

	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					TDUC[3:0]				UCV[7:0]							
W																
Initial	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 26-186. MAPLE UCode Version Parameter

Table 26-133. MAPLE UCode Version Parameter Fields Description

Bits	Description
31–12	Reserved.
11–8	TDUC — Target Device firmware (uCode). Indicated to which device this uCode applies. 0000 The uCode is targeted for the MSBA8100 device 0001 The uCode is targeted for the MSC8156Rev1 device. 0010 The uCode is targeted for the MSC8156Rev2 device. 0011 The uCode is targeted for the MSC8157 device.
7–0	UCV — Ucode Version. This field describes the ucode and API version used to initiate the MAPLE-B2.

26.5.3.1.2 MAPLE Timer Period Parameter (MP_TPP)

This parameter sets the time periods between each time the internal RISC engines are being called to search for new Buffer Descriptor. For details, refer to **Section 26.4.5**, *MAPLE-B2 Internal Task Control*

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 26-187. MAPLE Timer Period Parameter

Table 26-134. MAPLE Timer Period Parameter Fields Description

Bits	Description
31–6	Reserved
5–0	TMR_PRD — Timer Period. Determines the periodical 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 details, refer to **Section 26.4.5, *MAPLE-B2 Internal Task Control***

26.5.3.1.3 MAPLE Clock Gating Control Parameter (MCGCP)

This parameter allows controlling (disabling/enabling) the input clocks for the CRPE and the 3 x eFTPE (shared clock gating for the 3 eFTPE modules), thus saving power when not in use for long periods of time (such as over-night shutdown). By setting this parameter and initiating the `Maple_crpe_3x_eftpe_clock_gate_control` routine (see **Section 26.4.5**, *MAPLE-B2 Internal Task Control*), the MAPLE-B2 disable/enable the clocks of the CRPE and 3 x eFTPE. For details see **Section 26.4.6.2**, *Per Processing Element Static Clock Gating Scheme*.

Offset 0x0000001C (MCGCP) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R							FT_CGE					CR_CGE				
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							FT_CGD					CR_CGD				
W																
Initial	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 26-188. MAPLE Clock Gating Control Parameter

Table 26-135. MAPLE Clock Gating Control Parameter Fields Description

Bits	Description
31–26	Reserved
25	FT_CGE — eFTPEs Clock Gating Enable. If set, MAPLE-B2 enable the input clock of the 3 x eFTPE modules. The MAPLE-B2 reset this bit once clock is enabled. 0 Do not enable the 3 x eFTPEs input clock. 1 Enable the input clock of the 3 x eFTPEs.
24–21	Reserved
20	CR_CGE — CRPE Clock Gating Enable. If set, MAPLE-B2 enable the input clock of the CRPE modules. The MAPLE-B2 reset this bit once clock is enabled. 0 Do not enable the CRPE input clock. 1 Enable the input clock of the CRPE.
19–10	Reserved
9	FT_CGD — eFTPEs Clock Gating Disable. If set, MAPLE-B2 disable the input clock of the 3 x eFTPE modules. The MAPLE-B2 reset this bit once clock is disabled. 0 Do not disable the 3 x eFTPEs input clock. 1 Disable the input clock of the 3 x eFTPEs.
8–5	Reserved

Table 26-135. MAPLE Clock Gating Control Parameter Fields Description

Bits	Description
4	CR_CGD — CRPE Clock Gating Disable. CRPE Clock Gating Disable. If set, MAPLE-B2 disable the input clock of the CRPE modules. The MAPLE-B2 reset this bit once clock is disabled. 0 Do not disable the CRPE input clock. 1 Disable the input clock of the CRPE.
3–0	Reserved

26.5.3.2 eTVPE Parameter RAM Description

Table 26-136 describes the MAPLE-B2 eTVPE Parameter RAM in detail.

Table 26-136. eTVPE RAM Detailed Description

Address	Parameters	Access	Initialization Value[1]	Section
0x00000500–0x00000548	MTVPVxHCP. MAPLE Turbo Viterbi Puncturing Vector <x> High Configuration Parameter (x = 0...9)	R/W	0x00000000	**Section 26.5.3.2.1**
0x00000504–0x0000054C	MTVPVxLCP. MAPLE Turbo Viterbi Puncturing Vector <x> Low Configuration Parameter. (x = 0...9)	R/W	0x00000000	**Section 26.5.3.2.2**
0x00000550–0x00000558	MTVPPCyP. MAPLE Turbo Viterbi Puncturing Period Configuration <y> Parameter (y = 0...2)	R/W	0x00000000	**Section 26.5.3.2.3**
0x0000055C	Reserved			
0x00000560–0x00000570	MTVPVSxC0P. MAPLE Turbo Viterbi Polynomial Vector Set <x> Configuration 0 parameter (x = 1...3)	R/W	0x00000000	**Section 26.5.3.2.4**
0x00000564–0x00000574	MTVPVSxC1P. MAPLE Turbo Viterbi Polynomial Vector Set <x> Configuration 1 parameter (x = 1...3)	R/W	0x00000000	**Section 26.5.3.2.5**
0x00000578–0x000005FF	Reserved			

1. All the parameters in the parameter RAM are initialized during the MAPLE-B2 API initialization sequence, and are not hardware reset values. See the *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)

The following subsections describe the MAPLE-B2 eTVPE configuration parameters which should be initialize by the host after the MAPLE-B2 initialization.

26.5.3.2.1 MAPLE Turbo Viterbi Puncturing Vector x High Configuration Parameter (MTVPVxHCP)

These parameters, along with the **MTVPVxLCP** parameters, control the Viterbi Puncturing Vectors configuration values related to the eTVPE.

Offset 0x00000500 (MTVPVxHCP) Access: User read/write
offset x∗0x8
range x = 0...9

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		PVEC[62:48]														
W																
Initial	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PVEC[47:32]															
W																
Initial	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 26-189. MAPLE Turbo Viterbi Puncturing Vector x High Configuration Parameter

Table 26-137. MAPLE Turbo Viterbi Puncturing Vector x High Configuration Parameter Fields Description

Bits	Description
31	Reserved.
31–0	PVEC[62:32]—Puncturing Vector. Last 31 bits of the puncturing vector. For each bit in the vector: 0 Punctured symbol 1 Valid symbol

26.5.3.2.2 MAPLE Turbo Viterbi Puncturing Vector x Low Configuration Parameter (MTVPVxLCP)

These parameters, together with the **MTVPVxHCP** parameters, control the Viterbi Puncturing Vectors configuration values related to the eTVPE.

Offset 0x00000504 (MTVPVxLCP) Access: User read/write
offset x∗0x8
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]															
Initial	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PVEC[15:0]															
Initial	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 26-190. MAPLE Turbo Viterbi Puncturing Vector x Low Configuration Parameter

Table 26-138. MAPLE Turbo Viterbi Puncturing Vector x Low Configuration Parameter Fields Description

Bits	Description
31–0	PVEC[31:0]—Puncturing Vector. First 32 bits of the puncturing vector. For each bit in the vector: 0 Punctured symbol 1 Valid symbol

26.5.3.2.3 MAPLE Turbo Viterbi Puncturing Period Configuration y Parameter (MTVPPCyP)

These parameters contain the period of the punctured input vectors for Viterbi processing.

Offset 0x00000550 (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																
Initial	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	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																
Initial	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 26-191. MAPLE Turbo Viterbi Puncturing Period Configuration y Parameter

Table 26-139. MAPLE Turbo Viterbi Puncturing Period Configuration y Parameter Fields Description

Bits	Description
31–30	Reserved
29–24	PPER<4y>—Puncturing Period. This field indicates the period of the puncturing if periodic puncturing is enabled. See **Section 26.4.3.2.5, *eTVPE Input Data Structures***. 1–63 possible values which stand for the possible periods
23–22	Reserved
21–16	PPER<4y+1>—Puncturing Period. This field indicates the period of the puncturing if periodic puncturing is enabled. See **Section 26.4.3.2.5, *eTVPE Input Data Structures***. 1–63 possible values which stand for the possible periods
15–14	Reserved
13–8	PPER<4y+2>—Puncturing Period. This field indicates the period of the puncturing if periodic puncturing is enabled. See **Section 26.4.3.2.5, *eTVPE Input Data Structures***. 1–63 possible values which stand for the possible periods
7–6	Reserved
5–0	PPER<4y+3>—Puncturing Period. This field indicates the period of the puncturing if periodic puncturing is enabled. See **Section 26.4.3.2.5, *eTVPE 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.

26.5.3.2.4 MAPLE Turbo Viterbi Polynomial Vector Set x Configuration 0 Parameter (MTVPVSxC0P)

These parameters, along with the MTVPVSxC1P parameters, define three sets of Viterbi polynomial configurations. The MAPLE-B2 determines with which set to configure the eTVPE according to the [VIT_SET] field of the eTVPE BD. For details, refer to **Section 26.5.4.2**, *eTVPE Buffer-Descriptor Structure*.

Offset 0x00000560 (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 26-192. MAPLE Turbo Viterbi Polynomial Vector Set x Configuration 0 Parameter

Table 26-140. MAPLE Turbo Viterbi Polynomial Vector Set x Configuration 0 Parameter Fields Description

Bits	Description
31–25	Reserved
24–16	M_POLYGEN1 - This field enables bits for the generation of the second parity bit. 0–(2^k - 1) possible values. K is the Viterbi Constraint Length.
15–9	Reserved
8–0	M_POLYGEN0 - This field enables bits for the generation of the first parity bit. 0–(2^k - 1) possible values. K is the Viterbi Constraint Length.

26.5.3.2.5 MAPLE Turbo Viterbi Polynomial Vector Set x Configuration 1 Parameter (MTVPVSxC1P)

These parameters, along with the MTVPVSxC0P parameters, define three sets of Viterbi polynomial configurations. The MAPLE-B2 determines with which set to configure the eTVPE according to the [VIT_SET] field of the eTVPE BD. For details, refer to **Section 26.5.4.2**, *eTVPE Buffer-Descriptor Structure*.

Offset 0x00000564 (MTVPVSxC1P) 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_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 26-193. MAPLE Turbo Viterbi Polynomial Vector Set x Configuration 1 Parameter

Table 26-141. MAPLE Turbo Viterbi Polynomial Vector Set x Configuration 1 Parameter Fields Description

Bits	Description
31–25	Reserved
24–16	M_POLYGEN3 - This field enables bits for the generation of the forth parity bit. Valid only for Viterbi Rate 1/4, must contain 0x0000 otherwise. 0–(2^k - 1) possible values. K is the Viterbi Constraint Length.
15–9	Reserved
8–0	M_POLYGEN2 - This field enables bits for the generation of the third parity bit. Valid only for Viterbi Rate 1/3 and 1/4, must contain 0x0000 otherwise. 0–(2^k - 1) possible values. K is the Viterbi Constraint Length.

26.5.3.3 eFTPE Parameter RAM Description

Table 26-142 describes the MAPLE-B2 eFTPE Parameter RAM in detail.

Table 26-142. eFTPE Parameter RAM Detailed Description

Address	Parameters	Access	Initialization Value[1]	Section
0x00000600–0x00000663	FTPEDSSxP0. eFTPE Data Size Set <x> Parameter 0 (x = 1...7)	R/W	0x00000000	**Section 26.5.3.3.1**
0x00000604–0x00000667	FTPEDSSxP1. eFTPE Data Size Set <x> Parameter 1 (x = 1...7)	R/W	0x00000000	**Section 26.5.3.3.2**
0x00000608–0x0000066B	FTPEDSSxP2. eFTPE Data Size Set <x> Parameter 2 (x = 1...7)	R/W	0x00000000	**Section 26.5.3.3.3**
0x0000066C	Reserved.			
0x00000670–0x00000693	FTPExUPRMBPP. eFTPE <x> Update Pre-Multiplication Buffer Pointer Parameter. (x = 0...2)	R/W	0x00000000	**Section 26.5.3.3.4**
0x00000674–0x00000697	FTPExUPSMBPP. eFTPE <x> Update Post-Multiplication Buffer Pointer Parameter. (x = 0...2)	R/W	0x00000000	**Section 26.5.3.3.5**
0x00000678–0x0000069B	FTPExUBSP. eFTPE <x> Update Buffers Size Parameter. (x = 0...2)	R/W	0x00000000	**Section 26.5.3.3.6**
0x0000069C	Reserved.			
0x000006A0	FTPECUBRP. eFTPE Complete Update Buffers Routine Parameter.	R/W	0x00000000	**Section 26.5.3.3.7**
0x000006A4–0x000006BF	Reserved.			

1. All the parameters in the parameter RAM are initialized during the MAPLE-B2 API initialization sequence, and are not hardware reset values. See the *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)

The following subsections describe the MAPLE-B2 eFTPE configuration parameters.

26.5.3.3.1 eFTPE Data Size Set x Parameter 0 (FTPEDSSxP0)

This parameter records half the data size to be input or output by MAPLE-B2 if Guard Insertion or Guard Removal are enabled. For details, see **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT* and **Section 26.4.3.3.4.2.1**, *Guard Band Removal for FFT*.

Offset 0x00000600 (FTPEDSSxP0) Access: User read/write
offset 0x10
x = 1...7

	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 26-194. eFTPE Data Size Set x Parameter 0

Table 26-143. eFTPE Data Size Set x Parameter 0 Field Descriptions

Bits	Description
31–26	Reserved
25–16	DS0—Data Size 0. Size of the Right/Left data of the iFFT/FFT Transform input/output with the length of 128. For details, see **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT* and **Section 26.4.3.3.4.2.1**, *Guard Band Removal for FFT*
15–10	Reserved
9–0	DS1—Data Size 1. Size of the Right/Left data of the iFFT/FFT Transform input/output with the length of 256. For details, see **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT* and **Section 26.4.3.3.4.2.1**, *Guard Band Removal for FFT*

26.5.3.3.2 eFTPE Data Size Set x Parameter 1 (FTPEDSSxP1)

This parameter records half the data size to be input or output by MAPLE-B2 if Guard Insertion or Guard Removal are enabled. For details, see **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT* and **Section 26.4.3.3.4.2.1**, *Guard Band Removal for FFT*.

Offset 0x00000604 (FTPEDSSxP1) Access: User read/write
offset 0x10
x = 1...7

	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 26-195. eFTPE Data Size Set x Parameter 1

Table 26-144. eFTPE Data Size Set x Parameter 1Field Descriptions

Bits	Description
31–26	Reserved
25–16	DS2—Data Size 2. Size of the Right/Left data of the iFFT/FFT Transform input/output with the length of 512. For details, see **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT* and **Section 26.4.3.3.4.2.1**, *Guard Band Removal for FFT*
15–10	Reserved
9–0	DS3—Data Size 3. Size of the Right/Left data of the iFFT/FFT Transform input/output with the length of 1024. For details, see **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT* and **Section 26.4.3.3.4.2.1**, *Guard Band Removal for FFT*

26.5.3.3.3 eFTPE Data Size Set x Parameter 2(FTPEDSSxP2)

This parameter records half the data size to be input or output by MAPLE-B2 if Guard Insertion or Guard Removal are enabled. For details, see **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT* and **Section 26.4.3.3.4.2.1**, *Guard Band Removal for FFT*.

Offset 0x00000608 (FTPEDSSxP2) Access: User read/write
offset 0x10
x = 1...7

	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	DS5									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-196. eFTPE Data Size Set x Parameter 2

Table 26-145. eFTPE Data Size Set x Parameter 2Field Descriptions

Bits	Description
31–26	Reserved
25–16	DS4—Data Size 4. Size of the Right/Left data of the iFFT/FFT Transform input/output with the length of 1536. For details, see **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT* and **Section 26.4.3.3.4.2.1**, *Guard Band Removal for FFT*
15–10	Reserved
9–0	DS5—Data Size 5. Size of the Right/Left data of the iFFT/FFT Transform input/output with the length of 2048. For details, see **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT* and **Section 26.4.3.3.4.2.1**, *Guard Band Removal for FFT*

26.5.3.3.4 eFTPE x Update Pre-Multiplication Buffer Pointer Parameter (FTPExUPRMBPP)

This parameter indicates the MAPLE-B2 during the ‘Maple_update_eftpe_pre_post_buffers’ routine assertion (see **Section 26.4.5**) if to update the pre-multiplication buffer of eFTPE x. For details see **Section 26.4.3.3.5.3**, *‘One-Shot’ Initialization of the Pre/Post Multiplication buffers*.

Offset 0x00000670 (FTPExUPRMBPP) Access: User read/write
offset 0x10
x = 0...2

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

Figure 26-197. eFTPE x Update Pre-Multiplication Buffer Pointer Parameter

Table 26-146. eFTPE x Update Pre-Multiplication Buffer Pointer Field Descriptions

Bits	Description
31–0	PRE_BUF_PTR—Pre Multiplication Buffer Pointer. Valid during ‘Maple_update_eftpe_pre_post_buffers’ activation. If set to a value different than zero, the MAPLE-B2 uses this field as a pointer to the system memory where the new pre-multiplication buffer of eFTPE x is expected. The size of the buffer is described in *FTPExUBSP[PRE_BUF_SIZE]*.

26.5.3.3.5 eFTPE x Update Post-Multiplication Buffer Pointer Parameter (FTPExUPSMBPP)

This parameter indicates the MAPLE-B2 during the ‘Maple_update_eftpe_pre_post_buffers’ routine assertion (see **Section 26.4.5**) if to update the post-multiplication buffer of eFTPE x. for details see **Section 26.4.3.3.5.3**, *‘One-Shot’ Initialization of the Pre/Post Multiplication buffers*.

Offset 0x00000674 (FTPExUPSMBPP) Access: User read/write
offset 0x10
x = 0...2

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

Figure 26-198. eFTPE x Update Post-Multiplication Buffer Pointer Parameter

Table 26-147. eFTPE x Update Post-Multiplication Buffer Pointer Field Descriptions

Bits	Description
31–0	POST_BUF_PTR—Post Multiplication Buffer Pointer. Valid during ‘Maple_update_eftpe_pre_post_buffers’ activation. If set to a value different than zero, the MAPLE-B2 uses this field as a pointer to the system memory where the new post-multiplication buffer of eFTPE x is expected. The size of the buffer is described in *FTPExUBSP[PST_BUF_SIZE]*.

26.5.3.3.6 eFTPE x Update Buffers Size Parameter (FTPExUBSP)

This parameter indicates the MAPLE-B2 during the ‘Maple_update_eftpe_pre_post_buffers’ routine assertion (see **Section 26.4.5**) the size of the pre and/or post multiplication buffers to be fetched. For details see **Section 26.4.3.3.5.3**, *‘One-Shot’ Initialization of the Pre/Post Multiplication buffers*.

Offset 0x00000678 (FTPExUBSP) Access: User read/write
offset 10
x = 0...2

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

Figure 26-199. eFTPE x Update Buffers Size Parameter

Table 26-148. eFTPE x Update Buffers Size Field Descriptions

Bits	Description
31–16	PRE_BUF_SIZE—Pre Multiplication Buffer Size. Valid during ‘Maple_update_eftpe_pre_post_buffers’ activation. If the relevant PRE_BUF_PTR is enabled (> 0), this field indicates the size (in bytes) of the new pre-multiplication buffer to be fetched by MAPLE-B2. Valid values: up to 8KB, which is the maximal size of the pre-multiplication buffer.
15–0	PST_BUF_SIZE—Post Multiplication Buffer Size. Valid during ‘Maple_update_eftpe_pre_post_buffers’ activation. If the relevant POST_BUF_PTR is enabled (> 0), this field indicates the size (in bytes) of the new post-multiplication buffer to be fetched by MAPLE-B2. Valid values: up to 8KB, which is the maximal size of the post-multiplication buffer.

26.5.3.3.7 eFTPE Complete Update Buffers Routine Parameter (FTPECUBRP)

This parameter must be cleared by the host before initiating the ‘Maple_update_eftpe_pre_post_buffers’ routine (see **Section 26.4.5**). Once the routine is completed the MAPLE-B set this parameter to ‘1’ indicating the host that the routine is completed and new Buffer Descriptors can be written into MAPLE-B2 internal memory. For details see **Section 26.4.3.3.5.3**, *‘One-Shot’ Initialization of the Pre/Post Multiplication buffers*

Offset 0x000006A0 (FTPECUBRP) 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																UBC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-200. eFTPE Complete Update Buffers Routine Parameter

Table 26-149. eFTPE Complete Update Buffers Routine Field Descriptions

Bits	Description
31–1	Reserved
0	UBC—Update Buffers Complete indication. Should be cleared by the host prior to initiating the ‘Maple_update_eftpe_pre_post_buffers_routine’. Set to ‘1’ by the MAPLE-B2 once the routine is completed. For details see **Section 26.4.3.3.5.3**, *‘One-Shot’ Initialization of the Pre/Post Multiplication buffers*.

26.5.3.4 EQPE Parameter RAM Description

Table 26-150 describes the MAPLE-B2 EQPE Parameter RAM in detail.

Table 26-150. EQPE Parameter RAM Detailed Description

Address	Parameters	Access	Initialization Value[1]	Section
0x000006C0	MEQYBMP. MAPLE EQPE Y Buffer Mode Parameter	R/W	0x00000000	**Section 26.5.3.4**
0x000006C4–0x000006FF	Reserved			

1. All the parameters in the parameter RAM are initialized during the MAPLE-B2 API initialization sequence, and are not hardware reset values. See the *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)

The following section describes the MAPLE-B2 EQPE Y Buffer Mode Parameter (MEQYBMP). This parameter describe the expected order of the input Y buffers in the system memory. For details see **Section 26.4.3.5.4.1.1**, *Y Samples Input Data Structure*.

Offset 0x000006C0 (MEQYBMP) 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										
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										Y_MOD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-201. MAPLE EQPE Y Buffer Mode Parameter

Table 26-151. MAPLE EQPE Y Buffer Mode Field Descriptions

Name	Description
31–1	Reserved
0	Y_MODE—Y buffer mode. Valid for LNEQ and MLEQ modes only. Indicates the order of the antenna buffers (after FFT processing) in the system memory. For details see **Section 26.4.3.5.4.1.1**, *Y Samples Input Data Structure*. 0 - The Y buffer in the system memory is ordered by successive columns from the same Rx antenna. 1 - The Y buffer in the system memory is ordered by same column from different Rx antenna.

26.5.3.5 CRPE Parameter RAM Descriptions

The CRPE uses different sets of parameter RAM and control parameters for each function, as follows:

- CRPE General parameters
- CRPE Uplink Batch Mode (CRPE-ULB)
- CRPE Uplink Fast Processing (CRPE-ULF)
- CRPE Downlink (CRPE-DL)

The following subsections define these parameter sets.

26.5.3.5.1 CRPE General Parameter Description

26.5.3.5.1.1 MAPLE CRPE Reset Completion Indication Parameter (MCRRCIP)

This parameter indicates the status of the CRPE reset routine. Once the routine is initiated, the status bit is set by the MAPLE-B2 indicating the reset routine is active. When reset assertion and all internal configuration is completed, the bit is reset.

Offset 0x00000040 (MCRRCIP) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
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																CR_RST
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-202. MAPLE CRPE Reset Completion Indication Parameter

Table 26-152. MAPLE CRPE Reset Completion Indication Fields Description

Bits	Description
31–1	Reserved
0	CR_RST — CRPE Reset. Indicates the execution of the CRPE reset routine. Sets by the MAPLE-B2 at the beginning of the CRPE reset routine. Cleared by the MAPLE-B2 on CRPE reset routine completion. For mode details see **Section 26.4.3.6.4**, *CRPE Reset*. 0 - CRPE reset routine is not being processed. 1 - CRPE reset routine is being processed.

26.5.3.5.2 CRPE-ULB Parameters Description

Table 26-153 lists the MAPLE-B2 CRPE-ULB Parameter RAM sections.

Table 26-153. CRPE-ULB Parameter RAM Detailed Description

Address	Parameters	Access	Initialization Value[1]	Section
CRPE-ULB Parameter RAM				
0x00000800–0x000027FF	MCUBPCHxBAP - MAPLE CRPE ULB Physical Channel <x> Base Address Parameter (x = 0...511)	R/W	0x00000000	**Section 26.5.3.5.2.1**
	MCUBPCHxSZP - MAPLE CRPE ULB Physical Channel <x> Size Parameter (x = 0...511)	R/W	0x00000000	**Section 26.5.3.5.2.2**
	MCUBPCHxWPP - MAPLE CRPE ULB Physical Channel <x> Write Pointer Parameter (x = 0...511)	R/W	0x00000000	**Section 26.5.3.5.2.3**
	MCUBPCHxOBICP - MAPLE CRPE ULB Physical Channel <x> Output Buffer Interrupt configuration Parameter (x = 0...511)	R/W	0x00000000	**Section 26.5.3.5.2.4**
0x00002800–0x0000297F	MCUBGxC1P - MAPLE CRPE ULB Group x Configuration 1 Parameter (x = 0...23)	R/W	0x00000000	**Section 26.5.3.5.2.5**
	MCUBGxC2P - MAPLE CRPE ULB Group x Configuration 2 Parameter (x = 0...23)	R/W	0x00000000	**Section 26.5.3.5.2.6**
	MCUBGxC3P - MAPLE CRPE ULB Group x Configuration 3 Parameter (x = 0...23)	R/W	0x00000000	**Section 26.5.3.5.2.7**
	MCUBGxC4P - MAPLE CRPE ULB Group x Configuration 4 Parameter (x = 0...23)	R/W	0x00000000	**Section 26.5.3.5.2.8**
0x00002980–0x00002AFF	MCUBANTxDP - MAPLE CRPE ULB Antenna x Descriptor Parameter (x = 0...95)	R/W	0x00000000	**Section 26.5.3.5.2.9**
0x00002B00	MCUBCP - MAPLE CRPE ULB Configuration Parameter	R/W	0x00000000	**Section 26.5.3.5.2.10**
0x00002B04	MCUBOBICP - MAPLE CRPE ULB Output Buffer Interrupt Configuration Parameter.	R/W	0x00000000	**Section 26.5.3.5.2.11**
0x00002B08–0x00002BFF	Reserved			
0x00002C00–0x00002FFF	MCUBFCBxP - MAPLE CRPE ULB Finished Channels Buffer x Parameter (x = 0...511)	R/W	0x00000000	**Section 26.5.3.5.2.12**
0x00003000–0x000037FF	Group Core Descriptors Memory.	R/W	0x00000000	

1. All the parameters in the parameter RAM are initialized during the MAPLE-B2 API initialization sequence, and are not hardware reset values. See the *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)

26.5.3.5.2.1 MAPLE CRPE-ULB Physical Channel <x> Base Address Parameter (MCUBPCHxBAP)

This parameter holds the base address of physical channel number x.

Offset 0x00000800 (MCUBPCHxBAP) Access: User read/write
offset 0x10
x = 0...511

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

Figure 26-203. MAPLE CRPE-ULB Physical Channel x Base Address

Table 26-154. MAPLE CRPE-ULB Physical Channel x Base Address Fields Description

Bits	Description
31–5	PCH_xBA—Physical Channel $_x$ Base Address. Holds the base address in the system memory of the output buffer of Physical Channel x. The address must be 32 bytes aligned (bits [4:0] are reserved)
4–0	Reserved

26.5.3.5.2.2 MAPLE CRPE-ULB Physical Channel <x> Size Parameter (MCUBPCHxSZP)

This parameter holds the size of physical channel number x.

Offset 0x00000804 (MCUBPCHxSZP) Access: User read/write
offset 0x10
x = 0...511

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

Figure 26-204. MAPLE CRPE-ULB Physical Channel x Size

Table 26-155. MAPLE CRPE-ULB Physical Channel x Size Fields Description

Bits	Description
31–20	Reserved
19–5	PCH_xSZ—Physical Channel $_x$ Size. Holds the size of the output buffer of Physical Channel x. The size (in bytes) must be a multiplication of 32 (bits [4:0] are reserved). 0x0001 - Output buffers size is 32 bytes 0x0002 - output buffer size is 64 bytes. ...
4–0	Reserved

26.5.3.5.2.3 MAPLE CRPE-ULB Physical Channel <x> Write Pointer Parameter (MCUBPCHxWPP)

This parameter holds the current write pointer in the output buffer of physical channel number x.

Offset 0x00000808 (MCUBPCHxWPP) Access: User read/write
offset 0x10
x = 0...511

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

Figure 26-205. MAPLE CRPE-ULB Physical Channel x Write Pointer

Table 26-156. MAPLE CRPE-ULB Physical Channel x Write Pointer Fields Description

Bits	Description
31–1	PCH_xWP—Physical Channel $_x$ Write Pointer. Holds the write pointer of the output buffer of Physical Channel x. The write pointer must be initialized to the value between PCH_xBA and (PCH_xBA + PCH_xSZ) before the channel activation. The initialization value must be an even number.

26.5.3.5.2.4 MAPLE CRPE-ULB Physical Channel <x> Output Buffer Interrupt Configuration Parameter (MCUBPCHxOBICP)

This parameter holds the interrupt configuration of physical channel number x.

Offset 0x0000080C (MCUBPCHxOBICP) Access: User read/write
offset 0x10
x = 0...511

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

Figure 26-206. MAPLE CRPE-ULB Physical Channel x Output Buffer Interrupt Configuration

Table 26-157. MAPLE CRPE-ULB Physical Channel x Output Buffer Interrupt Configuration Fields Description

Bits	Description
31–29	Reserved
28	MASK— Enable/Disable indication for the check if channel should be added to finished channels buffer. Should be initialized to 1 on channel activation. Maintained by MAPLE-B2 and used to prevent false channel finish indications may caused due to counters wrap events. For details see **Section 26.4.3.6.1.15**, *Output Interrupt and Finished Channels Buffer Maintenance.* 0—Checks are disabled. 1—Checks are enabled.
27	Reserved
26–16	MARK— Indicated the SBS number after which the related channel needs to be added to the finished channels buffer. Self increment by MAPLE-B2 by *STEP* value during group check points when the *MARK* value is smaller than the current internal output SBS counter. This field should be initialized with the required number of offset SBS by the HOST during the channel activation. 0 to 1200 valid values
15–11	Reserved
10–0	STEP— Indicates the number of SBS between two MAPLE ULB interrupt assertions related to this channel. After each MAPLE-B2 CRPE-ULB interrupt assertion the value in *STEP* is being added by MAPLE-B2 to the MARK value which becomes the new threshold for the next interrupt assertion. For details see **Section 26.4.3.6.1.15**, *Output Interrupt and Finished Channels Buffer Maintenance.* Value of zero for the *STEP* field indicated the MAPLE-B2 that no interrupt should be asserted for this channel. 0 to 1200 valid values.

26.5.3.5.2.5 MAPLE CRPE-ULB Group <x> Configuration 1 Parameter (MCUBGxC1P)

The MCUBGxC1P to MCUBGxC4P parameters holds the uplink processing group configurations.

Offset 0x00002800 (MCUBGxC1P) Access: User read/write
offset 0x10
x = 0...23

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

Figure 26-207. MAPLE CRPE-ULB Group x Configuration 1 Parameter

Table 26-158. MAPLE CRPE-ULB Group x Configuration 1 Fields Description

Bits	Description
31	Reserved
30–28	G_xNOC— Group Number of Cores. Indicates the number of cores assigned to this group. 0 - Single core is assigned to this group. 1 - two cores are assigned to this group. ... 7- eight cores are assigned to this group.
27–25	Reserved
24–16	G_xNOS— Group Number of Sub Slots. Holds the number of SBS commands list descriptors for each core in the command list dada-base. Valid values are 1 to 300.
15–2	G_xCDLBA — Group Cores Descriptors List Base Address. Pointer to list of cores descriptors related to this group. The base address is an offset in the Core Descriptors Memory and it must be 4 byte aligned.
1–0	Reserved.

26.5.3.5.2.6 MAPLE CRPE-ULB Group <x> Configuration 2 Parameter (MCUBGxC2P)

The MCUBGxC1P to MCUBGxC4P parameters holds the uplink processing group configurations.

Offset 0x00002804 (MCUBGxC2P) Access: User read/write
offset 0x10
x = 0...23

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

Figure 26-208. MAPLE CRPE-ULB Group x Configuration 2 Parameter

Table 26-159. MAPLE CRPE-ULB Group x Configuration 2 Fields Description

Bits	Description
31–25	Reserved
24–16	G_xSCRP — Group x Sub slot Commands Read Pointer. This field counts cyclically (modulo *GNOS*) the Sub-Slot indicates the index in the (command) Sub-Slot descriptors list that the VCOP is going to process. It is used for accessing the Sub-Slot descriptor lists of all the cores in the group. Initialized by the user and then increment by MAPLE-B2. For details see **Section 26.4.3.6.1.5**, *CRPE-ULB Command Processing Flow* Values: 0...*GNOS*-1
15–9	Reserved
8–0	G_xSRP— Group x Sub-Slot read Pointer. This field indicates the sub slot index that the VCOP is going to read next from all antennas associated with the current group. Initialized by the user and then increment by MAPLE-B2. Values: 0...299

26.5.3.5.2.7 MAPLE CRPE-ULB Group <x> Configuration 3 Parameter (MCUBGxC3P)

The MCUBGxC1P to MCUBGxC4P parameters holds the uplink processing group configurations.

Offset 0x00002808 (MCUBGxC3P) Access: User read/write
offset 0x10
x = 0...23

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

Figure 26-209. MAPLE CRPE-ULB Group x Configuration 3 Parameter

Table 26-160. MAPLE CRPE-ULB Group x Configuration 3 Fields Description

Bits	Description
31–28	Reserved
27–1]	G_xANTSZ — Antenna OVS Input Buffer size in 512 bytes multiplications (256 chips)
15–12	Reserved
11–0	G_xARP— Read pointer indication to all antennas in the group. Should be initialized by the host, and maintained by MAPLE-B2 during data read operations. This pointer indicates the offset in SBS (512 bytes / 256 chips) from the beginning of each antenna OVS buffer of the group. Calculating the group/antenna/Sub-Slot/OVS buffer address is done as follows: (*MCUBANTxDP[ABA] + OVS*ANTSZ + GxARP)*512*, where OVS = 0 to #(num of oversamples-1).

26.5.3.5.2.8 MAPLE CRPE-ULB Group <x> Configuration 4 Parameter (MCUBGxC4P)

The MCUBGxC1P to MCUBGxC4P parameters holds the uplink processing group configurations.

Offset 0x0000280C (MCUBGxC4P)
offset 0x10
x = 0...23

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								G_xOBSWP								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-210. MAPLE CRPE-ULB Group x Configuration 4 Parameter

Table 26-161. MAPLE CRPE-ULB Group x Configuration 4 Fields Description

Bits	Description
31–9	Reserved
8–0	G_xOBSWP — Group Output Buffer Sub Slot Write Pointer. Indicates the next sub slot index to be written to the Group output buffers. Initialized to *GSRP* and maintained by MAPLE-B2. Valid values:0 to 299

26.5.3.5.2.9 MAPLE CRPE-ULB Antenna <x> Descriptor Parameter (MCUBANTxDP)

Following is the data structure of the Antenna Descriptors of the CRPE-ULB:

Offset 0x00002980 (MCUBANTxDP) Access: User read/write
offset 4
x = 0...95

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

Figure 26-211. MAPLE CRPE-ULB Antenna x Descriptor Parameter

Table 26-162. MAPLE CRPE-ULB Antenna x Descriptor Fields Description

Bits	Description
31–9	ABA — Antenna Base Address in system memory. The address must be 512 bytes aligned (256 chips).
8–0	Reserved

26.5.3.5.2.10 MAPLE CRPE-ULB Configuration Parameter (MCUBCP)

This parameter holds the number of groups and the uplink processing group configuration.

Offset 0x00002B00 (MCUBCP) Access: User read/write

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

Figure 26-212. MAPLE CRPE-ULB Configuration Parameter

Table 26-163. MAPLE CRPE-ULB Configuration Fields Description

Bits	Description
31–18	Reserved
17–16	MB_PR — Valid only if the [MB_PR_SCH] of the MMC0P parameter (see **Section 26.5.2.1**, *MAPLE Mode Configuration 0 Parameter (MMC0P)*) is set to ‘01’ or ‘10’. Indicated the MBus accesses priority related to the CRPE-ULB. For details, see **Section 26.4.2.4**, *MBus Priority Scheme Configuration* 00— The MBus accesses related to the CRPE-ULB are initiated with priority ‘00’ ... 11—The MBus accesses related to the CRPE-ULB are initiated with priority ‘11’
15–-7	Reserved.
6–5	IIL — Input Interpolation Level. Indicates the input interpolation level in case of external interpolation mode (*INT_MODE* = 0). 00 - Input interpolation is 2. 01 - Input interpolation is 4. 10 - Input interpolation is 8. 11 - Input interpolation is 16.
4–0	NOG — Number of Groups. Indicates the total number of groups. Valid values: 1 to 24

26.5.3.5.2.11 MAPLE CRPE-ULB Output Buffer Interrupt Configuration Parameter (MCUBOBICP)

This parameter describes the general output buffer interrupt configuration parameters.

Offset 0x00002B04 (MCUBOBICP) Access: User read/write

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

Figure 26-213. MAPLE CRPE-ULB Output Buffer Interrupt Configuration Parameter

Table 26-164. MAPLE CRPE-ULB Output Buffer Interrupt Configuration Fields Description

Bits	Description
31–26	Reserved
25–16	OB_INT_CHK_STEP — Output Buffer Interrupt Check Step. Indicates the time distance in number of SBS, where MAPLE-B2 is to check the finished physical channels buffer. If set to zero no checks are performed and no output buffer interrupt is issued. For details see **Section 26.4.3.6.1.15**, *Output Interrupt and Finished Channels Buffer Maintenance.*
15–10	Reserved
9–1	FCWP — Finished Channels Write Pointer. Write pointer to the finished channels buffer in 2 bytes unit. It is the offset from the beginning of the buffer. Increment modulo 512 when adding a channel to the buffer. Host should maintain a read pointer to the buffer to be able to read newly added finished channels up to the FCWP see **Section 26.4.3.6.1.15**, *Output Interrupt and Finished Channels Buffer Maintenance.*
0	Reserved

26.5.3.5.2.12 MAPLE CRPE-ULB Finished Channels Buffer <x> Parameter (MCUBFCBxP)

This set of parameters is a cyclic list maintained by MAPLE-B2 and includes the finished physical channels.

Offset 0x00002C00 (MCUBFCBxP)
offset 2
x = 0... 511

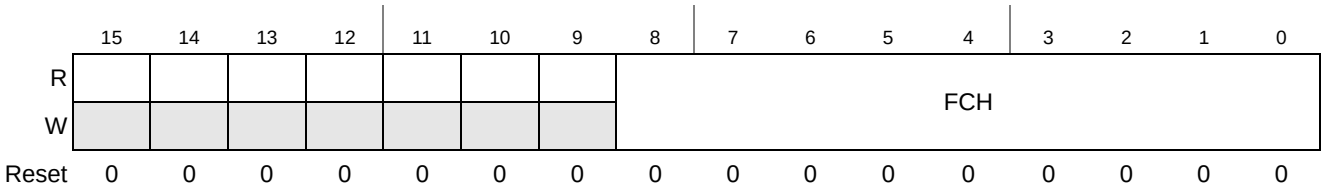


Figure 26-214. MAPLE CRPE-ULB Finished Channel Buffer Parameter

Table 26-165. MAPLE CRPE-ULB Finished Channel Buffer Fields Description

Bits	Description
15–9	Reserved
8–0	FCH — Finished Channel ID. A physical channel id of a finished channel. This array constitutes the finished channels buffer. MAPLE-B2 adds any new finished buffer to this buffer at the *MCUBOBICP[FCWP]* location and advances *MCUBOBICP[FCWP]*. For details see **Section 26.4.3.6.1.15**, *Output Interrupt and Finished Channels Buffer Maintenance.*

26.5.3.5.3 CRPE-ULF Parameters Description

The following subsections describe the CRPE-ULF description and control parameters. **Table 26-166** lists the MAPLE-B2 CRPE-ULF Parameter RAM sections.

Table 26-166. CRPE-ULF Parameter RAM Detailed Description

Address	Parameters	Access	Initialization Value[1]	Section
CRPE-ULF Parameter RAM				
0x00003800–0x0000381F	MCUFCIBAxP - MAPLE CRPE-ULF Commands Input Buffer Address x Parameter, (x = 0...7)	R/W	0x00000000	**Section 26.5.3.5.3.1**
0x00003820–0x0000383F	MCUFCIBWPxP - MAPLE CRPE-ULF Commands Input Buffer Write Pointer x Parameter, (x = 0...7)	R/W	0x00000000	**Section 26.5.3.5.3.2**
0x00003840–0x0000385F	MCUFCIBRPxP - MAPLE CRPE-ULF Commands Input Buffer Read Pointer x Parameter, (x = 0...7)	R/W	0x00000000	**Section 26.5.3.5.3.3**
0x00003860–0x0000386F	Reserved			
0x00003870	MCUFIBGCP - MAPLE CRPE-ULF Interpolation Bypass General Configuration Parameter.	R/W	0x00000000	**Section 26.5.3.5.3.4**
0x00003874	MCUFGCP - MAPLE-CRPE-ULF General Configuration Parameter	R/W	0x00000000	**Section 26.5.3.5.3.5**
0x00003878–0x0000387F	Reserved			
0x00003880–0x0000389F	MCUFIBGAxP - MAPLE CRPE-ULF Interpolation Bypass Group Attributes x parameter, (x = 0...7)	R/W	0x00000000	**Section 26.5.3.5.3.6**
0x000038A0–0x000038FF	MCUFIBAAxP - MAPLE CRPE-ULF Interpolation Bypass Ant Address x Parameter (x = 0...23)	R/W	0x00000000	**Section 26.5.3.5.3.7**
0x00003900–0x00003BFF	Reserved			

1. All the parameters in the parameter RAM are initialized during the MAPLE-B2 API initialization sequence, and are not hardware reset values. See the *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)

26.5.3.5.3.1 MAPLE CRPE-ULF Command Input Buffer Address <x> Parameter (MCUFCIBAxP)

This parameter holds the base address and size of the command input buffers.

Offset 0x00003800 (MCUFCIBAxP) Access: User read/write
offset 4
x = 0... 7

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

Figure 26-215. MAPLE CRPE-ULF Command Input Buffer Address <x>

Table 26-167. MAPLE CRPE-ULF Command Input Buffer Address <x> Fields Description

Bits	Description
31–8	BASEx — Points to the base address in the system memory of the cyclic command buffer #x. The address must be 256 bytes aligned.
7–0	SIZEx — Indicates the size of the cyclic command buffer #x in 256 bytes jumps. 0 - the size of the command buffer is 256 bytes. 1 - the size of the command buffer is 512 bytes. ... 255 - the size of the command buffer is 64Kbytes.

26.5.3.5.3.2 MAPLE CRPE-ULF Command Input Buffer Write Pointer <x> Parameter (MCUFCIBWPxP)

This parameter holds the write pointer of the command input buffers.

Offset 0x00003820 (MCUFCIBWPxP) Access: User read/write
offset 4
x = 0... 7

	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	WPx															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-216. MAPLE CRPE-ULF Command Input Buffer Write Pointer <x>

Table 26-168. MAPLE CRPE-ULF Command Input Buffer Write Pointer <x> Fields Description

Bits	Description
31–16	Reserved.
15–3	WPx — Command Buffer #x Write Pointer indication. Points to the next address which is ready. The pointer is 8 bytes aligned and is with respect to the base address of the Command Buffer base address as described in MCUFCIBAxP[BASEx]. The WP indication should be increment by the host when new commands are launched into the system command buffer.
2–0	Reserved.

26.5.3.5.3.3 MAPLE CRPE-ULF Command Input Buffer Read Pointer <x> Parameter (MCUFCIBRPxP)

This parameter holds the read pointer of the command input buffers.

Offset 0x00003840 (MCUFCIBRPxP) Access: User read/write
offset 4
x = 0... 7

	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	RPx															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-217. MAPLE CRPE-ULF Command Input Buffer Read Pointer <x>

Table 26-169. MAPLE CRPE-ULF Command Input Buffer Read Pointer <x> Fields Description

Bits	Description
31–16	Reserved.
15–3	RPx — Command Buffer #x Read Pointer indication. Points to the last address which has been read by MAPLE-B2. The pointer is 8 bytes aligned and is with respect to the base address of the Command Buffer base address as described in MCUFCIBAxP[BASEx]. The pointer is maintained by MAPLE-B2.
2–0	Reserved.

26.5.3.5.3.4 MAPLE CRPE-ULF Interpolation Bypass General Configuration Parameter (MCUFIBGCP)

This parameter holds the general configuration parameters for interpolation bypass mode.

Offset 0x00003870 (MCUFIBGCP) Access: User read/write

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

Figure 26-218. MAPLE CRPE-ULF Interpolation Bypass General Configuration

Table 26-170. MAPLE CRPE-ULF Interpolation Bypass General Configuration Fields Description

Bits	Description
31–27	TMR — Time (in clock cycles) to wait between last buffer check and next buffer check in 32 chips processing time window. The number of wait cycles are calculates as follows: (TMR + 4) x 130. 0—wait period is 520 cycles. 1—wait period is 650 cycles. ...
26–24	NOG — Number Of Groups. 0 - Single group 1 - Two groups. ... 7 - Eight groups.
23–16	SIZE — Size of OVS buffer in 512 byte resolution. Each antenna's buffer size is calculated as follows: SIZE x 512 x OVS. Size is common to all antenna buffers. 0—Reserved. 1—Size of OVS buffer is 512 Bytes 2—Size of OVS buffer is 1024 Bytes ...
15–6	RP — Joined Read pointer common to all groups antennas. The Read Pointer is 64 bytes aligned. The read pointer is increment by MAPLE-B2 when fetching new input data from all groups.
5–0	Reserved.

26.5.3.5.3.5 MAPLE CRPE-ULF General Configuration Parameter (MCUFGCP)

This parameter holds the general configuration parameters of the CRPE-ULF.

Offset 0x00003874 (MCUFGCP) 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															MB_PR	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-219. MAPLE CRPE-ULF General Configuration Parameter

Table 26-171. MAPLE CRPE-ULF General Configuration Fields Description

Bits	Description
31–2	Reserved.
1–0	MB_PR — Valid only if the [MB_PR_SCH] of the MMC0P parameter (see **Section 26.5.2.1**, *MAPLE Mode Configuration 0 Parameter (MMC0P)*) is set to ‘01’ or ‘10’. Indicated the MBus accesses priority related to the CRPE-ULF Command fetching and input data fetching (in case of interpolation bypass). For details, see **Section 26.4.2.4**, *MBus Priority Scheme Configuration*. The CRPE-ULF output results accesses priority is set in the *ULFGCR[MP].* 00— The MBus accesses related to command fetching and data fetching for interpolation bypass are initiated with priority ‘00’ (lowest priority) ... 11—The MBus accesses related to command fetching and data fetching for interpolation bypass are initiated with priority ‘11’ (highest priority)

26.5.3.5.3.6 MAPLE CRPE-ULF Interpolation Bypass Group Attributes <x> Parameter (MCUFIBGAxP)

These parameters holds the groups attributes for interpolation bypass mode.

Offset 0x00003880 (MCUFIBGAxP) Access: User read/write
offset 4
x = 0... 7

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

Figure 26-220. MAPLE CRPE-ULF Interpolation Bypass Group Attributes <x>

Table 26-172. MAPLE CRPE-ULF Interpolation Bypass Group Attributes Fields Description

Bits	Description
31–29	Reserved
28–24	FIAx — First Antenna index of the group indication.
23–21	Reserved
20–16	GNOAx — Number of Antennas associated with the Group. 0—Reserved. 1—One antenna is associated with the Group 2—Two antennas are associated with the Group ...
15–6	WPx — Joined Write Pointer of host for all its group antenna buffers. This value represent 64 Bytes chunks. Should be updated by host once input data of all antennas related to that group is placed in system memory.
5–0	Reserved.

26.5.3.5.3.7 MAPLE CRPE-ULF Interpolation Bypass Antenna Address <x> Parameter (MCUFIBAAxP)

This set of parameters holds the base address of the antenna input buffers in system memory.

Offset 0x000038A0 (MCUFIBAAxP) Access: User read/write
offset 4
x = 0... 23

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

Figure 26-221. MAPLE CRPE-ULF Interpolation Bypass Antenna Address <x>

Table 26-173. MAPLE CRPE-ULF Interpolation Bypass Antenna Address Fields Description

Bits	Description
31–8	BASEx — Points to the base address in system memory of data input buffer of antenna x.
7–0	Reserved

26.5.3.5.4 CRPE-DL Parameters Description

The following subsections describe the MAPLE-B2 CRPE-DL channels parameters and status fields. Most of the fields are updated by the host, some status fields are updated by MAPLE-B2. The parameters are implemented in MAPLE-B2's DRAM and used for channels' activation, termination and compression status update. The parameters contain all required information for performing symbols fetch from system buffer and chips transfers to system output buffers. **Table 26-174** lists the MAPLE-B2 CRPE-DL Parameter RAM sections.

Table 26-174. CRPE-DL Parameter RAM Detailed Description

Address	Parameters	Access	Initialization Value[1]	Section
CRPE-DL Parameter RAM				
0x00003C00–0x00007BFF	MCDLSCxP0 - MAPLE CRPE DownLink Slot Channel <x> Parameter 0 (x = 0...1023)	R/W	0x00000000	**Section 26.5.3.5.4.1**
	MCDLSCxP1 - MAPLE CRPE DownLink Slot Channel <x> Parameter 1 (x = 0...1023)	R/W	0x00000000	**Section 26.5.3.5.4.2**
	MCDLSCxP2 - MAPLE CRPE DownLink Slot Channel <x> Parameter 2 (x = 0...1023)	R/W	0x00000000	**Section 26.5.3.5.4.3**
	MCDLSCxRPP - MAPLE CRPE DownLink Slot Channel <x> Read Pointer Parameter (x = 0...1023)	R/W	0x00000000	**Section 26.5.3.5.4.4**
0x00007C00–0x0000887F	MCDLFCxP0 - MAPLE CRPE DownLink Fast Channel <x> Parameter 0 (x = 0...199)	R/W	0x00000000	**Section 26.5.3.5.4.5**
	MCDLFCxP1 - MAPLE CRPE DownLink Fast Channel <x> Parameter 1 (x = 0...199)	R/W	0x00000000	**Section 26.5.3.5.4.6**
	MCDLFCxP2 - MAPLE CRPE DownLink Fast Channel <x> Parameter 2 (x = 0...199)	R/W	0x00000000	**Section 26.5.3.5.4.7**
	MCDLFCxRPP - MAPLE CRPE DownLink Fast Channel<x> Read Pointer Parameter (x = 0...199)	R/W	0x00000000	**Section 26.5.3.5.4.8**
0x00008880–0x0000897F	MCDLOBxBAP - MAPLE CRPE-DL Output Buffer <x> Base Address Parameter (x = 0...15)	R/W	0x00000000	**Section 26.5.3.5.4.9**
	MCDLOBxSP - MAPLE CRPE-DL Output Buffer <x> Size Parameter (x = 0...15)	R/W	0x00000000	**Section 26.5.3.5.4.10**
	MCDLOBxWPP - MAPLE CRPE-DL Output Buffer <x> Write Pointer Parameter (x = 0... 15)	R/W	0x00000000	**Section 26.5.3.5.4.11**
0x00008980	MCDLNOCLP - MAPLE CRPE Downlink Number Of Channels Limit Parameter.	R/W	0x00000000	**Section 26.5.3.5.4.12**
0x00008984	MCDLGCP - MAPLE CRPE-DL General Configuration Parameter	R/W	0x00000000	**Section 26.5.3.5.4.13**
0x00008988–0x00008FFF	Reserved			

1. All the parameters in the parameter RAM are initialized during the MAPLE-B2 API initialization sequence, and are not hardware reset values. See the *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)

26.5.3.5.4.1 MAPLE CRPE-DL Slot Channel <x> Parameter 0 (MCDLSCxP0)

These parameters and status fields, along with *MCDLSCxP1*, *MCDLSCxP2* and *MCDLSCxRPP* parameters, control the channel #x activation, termination, compressed status update and symbols' fetch from system buffer. MAPLE-B2 checks these parameters and status fields once every slot.

Offset 0x00003C00 (MCDLSCxP0) Access: User read/write
offset 0x10
x = 0...1023

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MSBSZ															
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	MSBSZ			MVALID				MSTARTED	MESL				MSSL			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-222. MAPLE CRPE-DL Slot Channel <x> Parameter 0

Table 26-175. MAPLE CRPE-DL Slot Channel <x> Parameter 0 Fields Description

Bits	Description
31–14	MSBSZ — MAPLE-B2 System Buffer Size. Size of the channel's system buffer at 2 bytes unit. Range: 8 to (256K-1).
13	Reserved
12	MVALID — Valid indication for channel's parameters. Cleared by MAPLE-B2 at initialization after reset negation. Set by host upon parameters' update. Cleared by MAPLE-B2 when channel is terminated. 0 - Channel parameters are not valid. 1 - Channel parameters are valid.
11–9	Reserved
8	MSTARTED — Indication that channel has already been started. Cleared by MAPLE-B2 at initialization after reset negation. Cleared by host upon channel activation. Set by MAPLE-B2 at first channel's slot. Cleared by MAPLE-B2 when channel is terminated. 0 - Channel has not been started. 1 - Channel has been started.
7–4	MESL — MAPLE-B2 End Slot. Slot to stop the channel. Valid only when *MCONT*=0. Range: 0 to14.
3–0	MSSL — MAPLE-B2 Start Slot. When *STARTED*=0 and *MVALID*=1, *MSSL* defines the slot in which the channel starts. *MSSL* also defines channel's transition slot from non-compressed mode to compressed mode or vice-versa. Valid only when *MSTARTED*=0. Range: 0-14.

26.5.3.5.4.2 MAPLE CRPE-DL Slot Channel <x> Parameter 1 (MCDLSCxP1)

These parameters and status fields, along with *MCDLSCxP0*, *MCDLSCxP2* and *MCDLSCxRPP* parameters, control the channel #x activation, termination, compressed status update and symbols' fetch from system buffer. MAPLE-B2 checks these parameters and status fields once every slot.

Offset 0x00003C04 (MCDLSCxP1) Access: User read/write
offset 0x10
x = 0...1023

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		MCMPN	MCMPC	MSYNC												
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				MCONT	MESBS				MSSBS				MSFR	MSF		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-223. MAPLE CRPE-DL Slot Channel <x> Parameter 1

Table 26-176. MAPLE CRPE-DL Slot Channel <x> Parameter 1 Fields Description

Bits	Description
31	Reserved
30	MCMPN — MAPLE-B2 Compressed Mode Next. This field indicates compressed mode status of channel starting from *MSSL* and *MSSBS* (including). 0 - Non Compressed Mode. 1 - Compressed Mode.
29	MCMPC — MAPLE-B2 Compressed Mode Current. This field indicates compressed mode status of channel up until *MSSL* and *MSSBS* (not including). MAPLE-B2 overrides *MCMPC* by *MCMPN* when input stream associated with *MSSL* and *MSSBS* is handled. 0 - Non Compressed Mode. 1 - Compressed Mode.
28	MSYNC — MAPLE-B2 Synchronization Channel. Synchronization channel indication. For synchronization channel, 512 bytes are fetched at each slot, starting with *MSSL*=*MSSBS*=0. 0 - Non Synchronization Channel. 1 - Synchronization Channel.
27–13	Reserved
12	MCONT — MAPLE-B2 Continuing Channel. Indication that following *MSSL* and *MSSBS*, channel processing should continue, ignoring *MESL* and *MESBS*. 0 - non continuing channel, *MESL* and *MESBS* represent valid termination point of the channel. 1 - continuing channel, *MESL* and *MESBS* are non-valid and ignored.
11–8	MESBS — MAPLE-B2 End Sub Slot. Channel's processing ends at this sub-slot position at *ESL* slot. Valid only when *MCONT*=0. Range: 0-9.

Table 26-176. MAPLE CRPE-DL Slot Channel <x> Parameter 1 Fields Description

Bits	Description
7–4	MSSBS — MAPLE-B2 Start Sub Slot. Channel's processing starts from this sub-slot position at *MSSL* slot. Valid only when *MSTARTED*=0 and *MVALID*=1. Range: 0-9.
3	MSFR — MAPLE-B2 Spreading Factor Reduction. This field indicates whether the channel is reducing spreading factor by 2 when operating at compressed mode. 0 - Spreading factor is not reduced when operating at compressed mode. 1 - Spreading factor is reduced by 2 when operating at compressed mode.
2–0	MSF — MAPLE-B2 Spreading Factor This field indicates the spreading factor at non-compressed mode. 0 - spreading factor = 4 1 - spreading factor = 8 2 - spreading factor = 16 3 - spreading factor = 32 4 - spreading factor = 64 5 - spreading factor = 128 6 - spreading factor = 256 7 - spreading factor = 1 (for synchronization channels)

26.5.3.5.4.3 MAPLE CRPE-DL Slot Channel <x> Parameter 2 (MCDLSCxP2)

These parameters and status fields, along with *MCDLSCxP0*, *MCDLSCxP1* and *MCDLSCxRPP* parameters, control the channel #x activation, termination, compressed status update and symbols' fetch from system buffer. MAPLE-B2 checks these parameters and status fields once every slot.

Offset 0x00003C08 (MCDLSCxP2) Access: User read/write
offset 0x10
x = 0...1023

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

Figure 26-224. MAPLE CRPE-DL Slot Channel <x> Parameter 2

Table 26-177. MAPLE CRPE-DL Slot Channel <x> Parameter 2 Fields Description

Bits	Description
31–8	MSBBA — MAPLE-B2 System Buffer Base Address. Base address of system buffer in 256 Bytes unit.
7–0	Reserved

26.5.3.5.4.4 MAPLE CRPE-DL Slot Channel <x> Read Pointer Parameter (MCDLSCxRPP)

This is the read pointer for channel #x system buffer at 2 bytes unit, updated by MAPLE-B2.

Offset 0x00003C0C (MCDLSCxRPP) Access: User read/write
offset 0x10
x = 0...1023

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

Figure 26-225. MAPLE CRPE-DL Slot Channel <x> Read Pointer Parameter

Table 26-178. MAPLE CRPE-DL Slot Channel <x> Read Pointer Parameter Fields Description

Bits	Description
31–19	Reserved
18–1	MRD_PTR — MAPLE-B2 Read Pointer. System buffer read pointer at 2 Bytes unit, updated by MAPLE-B2.
0	Reserved

26.5.3.5.4.5 MAPLE CRPE-D1L Fast Channel <x> Parameter 0 (MCDLFCxP0)

These parameters and status fields, along with *MCDLFCxP1*, *MCDLFCxP2* and *MCDLFCxRPP* parameters, relate to low latency (sub-slot) activated channels and control the channel *#MFCID* activation, termination, compressed status update and symbols' fetch from system buffer. MAPLE-B2 checks these parameters and status fields once every SBS.

Offset 0x00007C00 (MCDLFCxP0) Access: User read/write
offset 0x10
x = 0...199

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MFSBSZ															
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	MFSBSZ			MFVALID				MFSTARTED	MFESL				MFSSL			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-226. MAPLE CRPE-DL Fast Channel <x> Parameter 0

Table 26-179. MAPLE CRPE-DL Fast Channel <x> Parameter 0 Fields Description

Bits	Description
31–14	MFSBSZ — MAPLE-B2 Fast System Buffer Size. Size of the channel's system buffer at 2 bytes unit. Range: 8 to (256K-1).
13	Reserved
12	MFVALID — Valid indication for fast channel's parameters. Cleared by MAPLE-B2 at initialization after reset negation. Set by host upon parameters' update. Cleared by MAPLE-B2 when channel is terminated. 0 - Channel parameters are not valid. 1 - Channel parameters are valid.
11–9	Reserved
8	MFSTARTED — Indication that fast channel has already been started. Cleared by MAPLE-B2 at initialization after reset negation. Cleared by host upon channel activation. Set by MAPLE-B2 at first channel's slot. Cleared by MAPLE-B2 when channel is terminated. 0 - Channel has not been started. 1 - Channel has been started.
7–4	MFESL — MAPLE-B2 End Slot. Slot to stop the fast channel. Valid only when *MCONT*=0. Range: 0 to14.
3–0	MFSSL — MAPLE-B2 Start Slot. When *SMFTARTED*=0 and *MFVALID*=1, *MSSL* defines the slot in which the fast channel starts. *MFSSL* also defines channel's transition slot from non-compressed mode to compressed mode or vice-versa. Valid only when *MFSTARTED*=0. Range: 0-14.

26.5.3.5.4.6 MAPLE CRPE-DL Fast Channel <x> Parameter 1 (MCDLFCxP1)

These parameters and status fields, along with *MCDLFCxP0*, *MCDLFCxP2* and *MCDLFCxRPP* parameters, relate to low latency (sub-slot) activated channels and control the channel $\#MFCID_x$ activation, termination, compressed status update and symbols' fetch from system buffer. MAPLE-B2 checks these parameters and status fields once every Sub Slot.

Offset 0x00007C04 (MCDLSCxP1) Access: User read/write
offset 0x10
x = 0...199

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		MFC MPN	MFC MPC	MFSY NC			MFCID									
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				MFC ONT	MFESBS				MFSSBS				MFSF R	MFSF		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-227. MAPLE CRPE-DL Fast Channel <x> Parameter 1

Table 26-180. MAPLE CRPE-DL Fast Channel <x> Parameter 1 Fields Description

Bits	Description
31	Reserved
30	MCMPN — MAPLE-B2 Fast channel Compressed Mode Next. This field indicates compressed mode status of channel starting from *MFSSL* and *MFSSBS* (including). 0 - Non Compressed Mode. 1 - Compressed Mode.
29	MFCMPC — MAPLE-B2 Fast channel Compressed Mode Current. This field indicates compressed mode status of channel up until *MFSSL* and *MFSSBS* (not including). MAPLE-B2 overrides *MFCMPC* by *MFCMPN* when input stream associated with *MFSSL* and *MFSSBS* is handled. 0 - Non Compressed Mode. 1 - Compressed Mode.
28	MFSYNC — MAPLE-B2 Synchronization of fast Channel. Synchronization channel indication. For synchronization channel, 512 bytes are fetched at each slot, starting with *MFSSL*=*MFSSBS*=0. 0 - Non Synchronization Channel. 1 - Synchronization Channel.
27–26	Reserved
25–16	MFCID - MAPLE-B2 Fast Channel ID. This field indicates the ID of the low latency activated channel whose parameters are being updated.
15–13	Reserved
12	MFCONT — MAPLE-B2 Continuing Fast Channel. Indication that following *MFSSL* and *MFSSBS*, channel processing should continue, ignoring *MFESL* and *MFESBS*. 0 - non continuing channel, *MFESL* and *MFESBS* represent valid termination point of the channel. 1 - continuing channel, *MFESL* and *MFESBS* are non-valid and ignored.

Table 26-180. MAPLE CRPE-DL Fast Channel <x> Parameter 1 Fields Description

Bits	Description
11–8	MFESBS — MAPLE-B2 Fast channel End Sub Slot. Channel's processing ends at this sub-slot position at ESL slot. Valid only when *MFCONT*=0. Range: 0-9.
7–4	MFSSBS — MAPLE-B2 Fast channel's Start Sub Slot. Channel's processing starts from this sub-slot position at MFSSL slot. Valid only when *MFSTARTED*=0 and *MFVALID*=1. Range: 0-9.
3	MFSFR — MAPLE-B2 fast channel's Spreading Factor Reduction. This field indicates whether the channel is reducing spreading factor by 2 when operating at compressed mode. 0 - Spreading factor is not reduced when operating at compressed mode. 1 - Spreading factor is reduced by 2 when operating at compressed mode.
2–0	MFSF — MAPLE-B2 Fast channel's Spreading Factor This field indicates the spreading factor at non-compressed mode. 0 - spreading factor = 4 1 - spreading factor = 8 2 - spreading factor = 16 3 - spreading factor = 32 4 - spreading factor = 64 5 - spreading factor = 128 6 - spreading factor = 256 7 - spreading factor = 1 (for synchronization channels)

26.5.3.5.4.7 MAPLE CRPE-DL Slot Channel <x> Parameter 2 (MCDLSCxP2)

These parameters and status fields, along with MCDLFCxP0, MCDLFCxP1 and MCDLFCxRPP parameters, relate to low latency (sub-slot) activated channels and control the channel #$MFCID_x$ activation, termination, compressed status update and symbols' fetch from system buffer. MAPLE-B2 checks these parameters and status fields once every SBS.

Offset 0x00007C08 (MCDLFCxP2) Access: User read/write
offset 0x10
x = 0...199

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

Figure 26-228. MAPLE CRPE-DL Fast Channel <x> Parameter 2

Table 26-181. MAPLE CRPE-DL Fast Channel <x> Parameter 2 Fields Description

Bits	Description
31–8	MFSBBA — MAPLE-B2 Fast channel's System Buffer Base Address. Base address of system buffer in 256 Bytes unit.
7–0	Reserved

26.5.3.5.4.8 MAPLE CRPE-DL Fast Channel <x> Read Pointer Parameter (MCDLFCxRPP)

This is the read pointer for low latency (SBS) activated channel #*MFCID* system buffer at 2 bytes unit, updated by MAPLE-B2.

Offset 0x00007C0C (MCDLFCxRPP) Access: User read/write
offset 0x10
x = 0...199

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

Figure 26-229. MAPLE CRPE-DL Fast Channel <x> Read Pointer Parameter

Table 26-182. MAPLE CRPE-DL Fast Channel <x> Read Pointer Parameter Fields Description

Bits	Description
31–19	Reserved
18–1	MFRD_PTR — MAPLE-B2 Fast channel Read Pointer. System buffer read pointer at 2 Bytes unit, updated by MAPLE-B2.
0	Reserved

26.5.3.5.4.9 MAPLE CRPE-DL Output Buffer <x> Base Address Parameter (MCDLOBxBAP)

This parameter along with *MCDLOBxSP* and *MCDLOBxWPP* control the antenna #x output chips transfer to system buffer. On internal CRPE-DL event, MAPLE-B2 is triggered to transfer 16 chips per antenna once every processing chunk period.

Offset 0x00008880 (MCDLOBxBAP) Access: User read/write
offset 0x10
x = 0...15

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

Figure 26-230. MAPLE CRPE-DL Output Buffer <x> Base Address Parameter

Table 26-183. MAPLE CRPE-DL Output Buffer <x> Base Address Parameter Fields Description

Bits	Description
31–7	MOBBA — MAPLE-B2 Output Buffer Base Address. The base address in the system memory must be 128 bytes aligned.
6–0	Reserved

26.5.3.5.4.10 MAPLE CRPE-DL Output Buffer <x> Size Parameter (MCDLOBxSP)

This parameter along with MCDLOBxBAP and MCDLOBxWPP control the antenna #x output chips transfer to system buffer. On internal CRPE-DL event, MAPLE-B2 is triggered to transfer 16 chips per antenna once every processing chunk period.

Offset 0x00008884 (MCDLOBxSP) Access: User read/write
offset 0x10
x = 0...15

	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		MOBSZ														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-231. MAPLE CRPE-DL Output Buffer <x> Size Parameter

Table 26-184. MAPLE CRPE-DL Output Buffer <x> Size Parameter Fields Description

Bits	Description
31–15	Reserved
14–7	MOBSZ — MAPLE-B2 Output Buffer Size. Size of the antenna's output buffer. The size must be 128 bytes aligned. Range: 8 - (256 – 1).
6–0	Reserved

26.5.3.5.4.11 MAPLE CRPE-DL Output Buffer <x> Write Pointer Parameter (MCDLOBxWPP)

This parameter along with *MCDLOBxBAP* and *MCDLOBxSP* control the antenna #x output chips transfer to system buffer. On internal CRPE-DL event, MAPLE-B2 is triggered to transfer 16 chips per antenna once every processing chunk period.

Offset 0x00008888 (MCDLOBxWPP) Access: User read/write
offset 0x10
x = 0...15

	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		MOWR_PTR														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-232. MAPLE CRPE-DL Output Buffer <x> Write Pointer Parameter

Table 26-185. MAPLE CRPE-DL Output Buffer <x> Write Pointer Parameter Fields Description

Bits	Description
31–15	Reserved
14–6	MOWR_PTR — MAPLE-B2 Output Write Pointer. Output buffer write pointer at 64 Bytes unit, maintained by MAPLE-B2.
5–0	Reserved

26.5.3.5.4.12 MAPLE CRPE-DL Number Of Channels Limit Parameter (MCDLNOCLP)

This parameter defines a limit to the number of channels used for downlink.

Offset 0x00008980 (MCDLNOCLP) Access: User read/write

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

Figure 26-233. MAPLE CRPE-DL Number Of Channels Limit Parameter

Table 26-186. MAPLE CRPE-DL Number Of Channels Limit Parameter Fields Description

Bits	Description
31–24	Reserved
23–16	MNOFCL — MAPLE Number Of Fast Channels Limit. This field defines a limit to the number of supported fast channels. This limit includes the external-sttd channels. For a limit of k channels, first k sets fast channels parameters are operational. Range: 0-200. **Note**: Though the number of supported sets of parameters for fast channels may exceed 100, actual transmission of only up to 100 fast channels should be active at the same time.
15–10	Reserved
9–0	MNORCL — MAPLE Number Of Regular Channels Limit. This field defines a limit to the number of supported channels. This limit does not include the external-sttd channels. For a limit of k channels, channels #0-#(k-1) are supported in addition to their optional coupled channels #512–#(k+511). Range: 0-512.

26.5.3.5.4.13 MAPLE CRPE-DL General Configuration Parameter (MCDLGCP)

This parameter defines the general configuration parameters of the CRPE-DL.

Offset 0x00008984 (MCDLGCP) 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															MB_PR	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-234. MAPLE CRPE-DL General Configuration Parameter

Table 26-187. MAPLE CRPE-DL General Configuration Parameter Fields Description

Bits	Description
31–2	Reserved
1–0	MB_PR — Valid only if the *MB_PR_SCH* of the *MMC0P* parameter (see **Section 26.5.2.1**, *MAPLE Mode Configuration 0 Parameter (MMC0P)*) is set to ‘01’ or ‘10’. Indicated the MBus accesses priority related to CRPE-DL. For details, see **Section 26.4.2.4**, *MBus Priority Scheme Configuration* 00— The MBus accesses related to the CRPE-DL are initiated with priority ‘00’ ... 11—The MBus accesses related to the CRPE-DL are initiated with priority ‘11’

26.5.3.6 Serial RapidIO Doorbell Support Attributes Parameters

For the functional description of the Serial RapidIO parameters, see **Section 26.4.4, *External Masters Support Using Serial RapidIO Doorbell*."**

26.5.3.6.1 Serial RapidIO Outbound RapidIO Doorbell Port 1 Base Address Parameter (SORDP0BAP)

This parameter defines the base address of the Serial RapidIO Outbound Doorbell controller of port 1 for MAPLE-B2.

Offset 0x00000480 (SORDP0BAP) Access: User read/write

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

Figure 26-235. Serial RapidIO Outbound RapidIO Doorbell Port 1 Base Address Parameter

Table 26-188. Serial RapidIO Outbound RapidIO Doorbell Port 1 Base Address Parameter Fields Description

Bits	Description
31–0	BA—Base Address of the Outbound Doorbell controller for port 1. Using this field MAPLE-B2 assumes the following addresses for the Outbound Doorbell controller of port 1: OD0MR BA + 0x0 OD0SR BA + 0x4 OD0DPR BA + 0x18 OD0DATR BA + 0x1C OD0RETCR BA + 0x2C This parameter is valid only if MBDR(H/L)PBxP[EXT_MST] equals '01'.

26.5.3.6.2 Serial RapidIO Outbound RapidIO Doorbell Port 2 Base Address Parameter (SORDP1BAP)

This parameter defines the base address of the Serial RapidIO Outbound Doorbell controller of port 2 for MAPLE-B2.

Offset 0x00000484 (SORDP1BAP) Access: User read/write

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

Figure 26-236. Serial RapidIO Outbound RapidIO Doorbell Port 2 Base Address Parameter

Table 26-189. Serial RapidIO Outbound RapidIO Doorbell Port 2 Base Address Parameter Field Description

Bits	Description
31–0	BA—Base Address of the Outbound Doorbell controller for port 2. Using this field MAPLE-B2 assumes the following addresses for the Outbound Doorbell controller of port 2: OD0MR[1] BA + 0x0 OD0SR BA + 0x4 OD0DPR BA + 0x18 OD0DATR BA + 0x1C OD0RETCR BA + 0x2C This parameter is valid only if MBDR(H/L)PBxP[EXT_MST] equals ’10’.

1. For details on the Doorbell controller registers see RMU_block_guide_Rev0.pdf section 4.4.1.1

26.5.3.6.3 Hardware Semaphore Port 1 Base Address Parameter (HSP0BAP)

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 of port 0 for the MAPLE-B2. If the enable bit is set, the Serial RapidIO Doorbell can only be accessed if the semaphore is locked by the MAPLE-B2.

Offset 0x00000488 (HSP0BAP) Access: User read/write

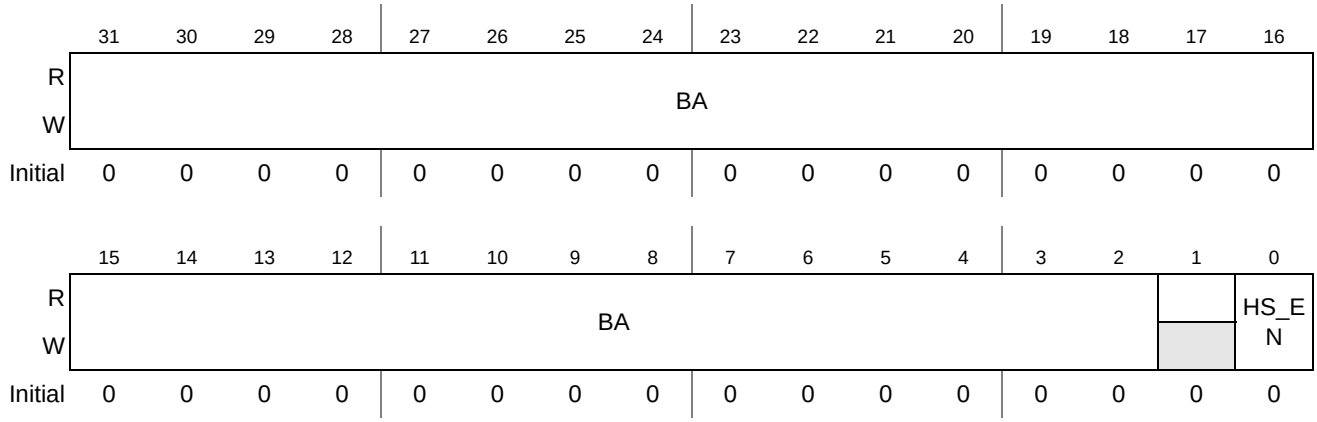


Figure 26-237. Hardware Semaphore Port 1 Base Address Parameter

Table 26-190. Hardware Semaphore Port 1 Base Address Parameter Field Description

Bits	Description
31–2	BA—Base Address of the hardware Semaphore register. If the HSP0BAP[HS_EN] is enabled and the MBDR(H/L)PBxP[EXT_MST] equals'01', then the MAPLE-B2 conditions its access to 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.
0	HS_EN—Hardware Semaphore Enable. If set, the MAPLE-B2 must lock the external semaphore located in address HSP0BAP[BA] with the PE ID (MDHSIDCP['pe'_HS_ID]) before it can access the doorbell controller of port 1. 0 - The MAPLE-B2 uses internal semaphore implementation to arbitrate between the PEs trying to access the doorbell controller port 1 1 - The MAPLE-B2 uses the external semaphore located at address HSP0BAP[BA] to arbitrate between the PEs and other external potential doorbell controller masters of port 1.

26.5.3.6.4 Hardware Semaphore Port 2 Base Address Parameter (HSP1BAP)

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 of port 1 for the MAPLE-B2. If the enable bit is set, the Serial RapidIO Doorbell can only be accessed if the semaphore is locked by the MAPLE-B2.

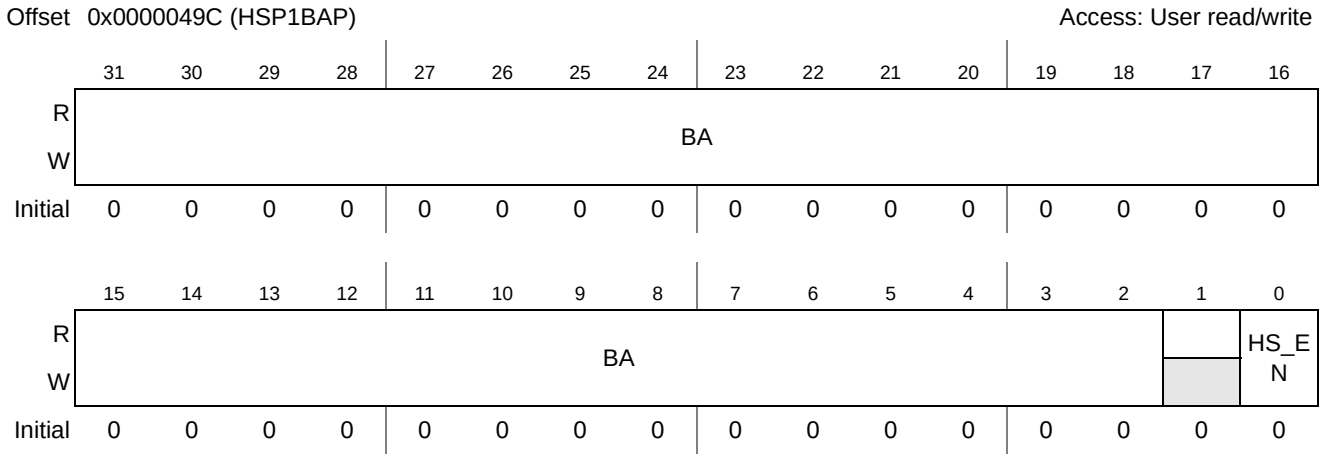


Figure 26-238. Hardware Semaphore Port 2 Base Address Parameter

Table 26-191. Hardware Semaphore Port 2 Base Address Parameter Field Description

Bits	Description
31–2	BA—Base Address of the hardware Semaphore register. If the HSP1BAP[HS_EN] is enabled and the MBDR(H/L)PBxP[EXT_MST] equals'10', then the MAPLE-B2 conditions its access to 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.
0	HS_EN—Hardware Semaphore Enable. If set, the MAPLE-B2 must lock the external semaphore located in address HSP1BAP[BA] with the PE ID (MDHSIDCP['pe'_HS_ID]) before it can access the doorbell controller of port 2. 0 - The MAPLE-B2 uses internal semaphore implementation to arbitrate between the PEs trying to access the doorbell controller port 2 1 - The MAPLE-B2 uses the external semaphore located at address HSP1BAP[BA] to arbitrate between the PEs and other external potential doorbell controller masters of port 2.

26.5.3.6.5 MAPLE-B Doorbell Hardware Semaphore ID Configuration Parameter (MDHSIDCP)

This parameter defines the ID number by which it locks the Hardware Semaphore register for MAPLE-B2.

Offset 0x00000490 (MDHSIDCP) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved 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 26-239. MAPLE-B Hardware Semaphore ID

Table 26-192. MAPLE-B Hardware Semaphore ID Field Description

Bits	Description
31 – 24	CRC_HS_ID—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-B2 must lock the External Hardware Semaphore (HS) before accessing the sRIO Doorbell controller. Accessing the HS for the CRCPE rings is done using the ID described in this field
23 –16	TV_HS_ID—eTVPE 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-B2 must lock the External Hardware Semaphore (HS) before accessing the sRIO Doorbell controller. Accessing the HS for the eTVPE rings is done using the ID described in this field.
15–8	DF_HS_ID—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-B2 must lock the External Hardware Semaphore (HS) before accessing the sRIO Doorbell controller. Accessing the HS for the DFTPE rings is done using the ID described in this field.
7–0	FF_HS_ID—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-B2 must lock the External Hardware Semaphore (HS) before accessing the sRIO Doorbell controller. Accessing the HS for the FFTPE rings is done using the ID described in this field.

26.5.3.6.6 MAPLE-B Doorbell General Configuration Parameter (MDGCP)

This parameter defines general parameters to be used while accessing the sRIO doorbell controller.

Offset 0x00000494 (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 26-240. MAPLE-B Hardware Semaphore ID

Table 26-193. MAPLE-B Hardware Semaphore ID Field Description

Bits	Description
31–10	Reserved.
9–8	BD_PR—sRIO Doorbell ‘Transaction flow level’. Defines the Priority of the doorbell transaction. This field is written by the MAPLE-B2 to the *OM0DATR[DTFLOWLVL]* register of the Doorbell controller.
7–0	RET—sRIO Doorbell ‘Retry error threshold’. Defines the number of times the Doorbell unit attempts to transmit the Doorbell. This field is written by the MAPLE-B2 to the *OD0RETCR[RET]* register of the Doorbell controller.

26.5.4 Descriptors

Descriptors include buffer descriptor (BD) rings, BDs, and task descriptors. The BD rings and BDs include general MAPLE-B2 BD rings and BDs and sets specific for individual PEs.

26.5.4.1 General BD Ring and BD Parameters

Table 26-194 lists the MAPLE-B2 general BD Ring and BD parameters.

Table 26-194. General BD Ring and BD Parameters

Address	Parameters	Access	Initialization Value[1]	Section
0x00000004	MBDRCP0. MAPLE BD Rings Configuration Parameter 0	R/W	0x00000000	**Section 26.5.4.1.1**
0x00000008	MBDRCP1. MAPLE BD Rings Configuration Parameter 1	R/W	0x00000000	**Section 26.5.4.1.2**
0x0000000C	MBDRCP2. MAPLE BD Rings Configuration Parameter 2	R/W	0x00000000	**Section 26.5.4.1.3**
0x00000080–0x000000BF	MTVBRHPAxP. MAPLE eTVPE BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MTVBRHPBxP. MAPLE eTVPE BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x000000C0–0x000000FF	MTVBRLPAxP. MAPLE eTVPE BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MTVBRLPBxP. MAPLE eTVPE BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000100–0x0000013F	MF0BRHPAxP. MAPLE EFTPE_0 BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MF0BRHPBxP. MAPLE EFTPE_0 BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x00000140–0x0000017F	MF0BRLPAxP. MAPLE EFTPE_0 BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MF0BRLPBxP. MAPLE EFTPE_0 BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000180–0x000001BF	MF1BRHPAxP. MAPLE EFTPE_1 BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MF1BRHPBxP. MAPLE EFTPE_1 BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x000001C0–0x000001FF	MF1BRLPAxP. MAPLE EFTPE_1 BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MF1BRLPBxP. MAPLE EFTPE_1 BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000200–0x0000023F	MF2BRHPAxP. MAPLE EFTPE_2 BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MF2BRHPBxP. MAPLE EFTPE_2 BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x00000240–0x0000027F	MF2BRLPAxP. MAPLE EFTPE_2 BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MF2BRLPBxP. MAPLE EFTPE_2 BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000280–0x000002BF	MDEBRHPAxP. MAPLE DEPE BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MDEBRHPBxP. MAPLE DEPE BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x000002C0–0x000002FF	MDEBRLPAxP. MAPLE DEPE BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MDEBRLPBxP. MAPLE DEPE BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000300–0x0000033F	MCRCBRHPAxP. MAPLE CRCPE BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MCRCBRHPBxP. MAPLE CRCPE BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**

Table 26-194. General BD Ring and BD Parameters (Continued)

Address	Parameters	Access	Initialization Value[1]	Section
0x00000340–0x0000037F	MCRCBRLPAxP. MAPLE CRCPE BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MCRCBRLPBxP. MAPLE CRCPE BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000380–0x000003BF	MEQBRHPAxP. MAPLE EQPE BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MEQBRHPBxP. MAPLE EQPE BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x000003C0–0x000003FF	MEQBRLPAxP. MAPLE EQPE BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MEQBRLPBxP. MAPLE EQPE BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000400–0x0000043F	MCGBRHPAxP. MAPLE CGPE BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MCGBRHPBxP. MAPLE CGPE BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x00000440–0x0000047F	MCGBRLPAxP. MAPLE CGPE BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MCGBRLPBxP. MAPLE CGPE BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**
0x00000480–0x000004BF	MCONVBRHPAxP. MAPLE CONVPE BD Ring High Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.4**
	MCONVBRHPBxP. MAPLE CONVPE BD Ring High Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.5**
0x000004C0–0x000004FF	MCONVBRLPAxP. MAPLE CONVPE BD Ring Low Priority A <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.6**
	MCONVBRLPBxP. MAPLE CONVPE BD Ring Low Priority B <x> Parameter (x = 0...7)	R/W	0x00000000	**Section 26.5.4.1.7**

1. All the parameters in the parameter RAM are initialized during the MAPLE-B2 API initialization sequence, and are not hardware reset values. See the *MAPLE-B2 Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)

The following subsections configure the BD rings and BDs.

Note: The MAPLE-B2 general configuration parameters should be initialized by the host after the MAPLE-B2 initialization.

26.5.4.1.1 MAPLE BD Rings Configuration Parameter 0 (MBDRCP0)

This parameter, along with the MBDRCP1 and MBDRCP2 parameters, determines the number of high/low BD rings used and the arbitration scheme between the high/low BD rings for each of the PEs.

Offset 0x00000004 (MBDRCP0) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		F0_R_LOOP				F0_HL_ARB				F1_R_LOOP				F1_HL_ARB		
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		F2_R_LOOP				F2_HL_ARB				TV_R_LOOP				TV_HL_ARB		
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-241. MAPLE BD Rings Configuration Parameter 0

Table 26-195. MAPLE BD Rings Configuration Parameter 0 Fields Description

Bits	Description
31	Reserved
30–28	F0_R_LOOP.—eFTPE_0 Ring Loop. The MAPLE-B2 assumes the potential valid high/low rings of the eFTPE_0 are located only between ring #0 and ring #(F0_R_LOOP). For details, see **Section 26.4.2.3.2, *BD Arbitration*.**" 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
26–24	F0_HL_ARB—eFTPE_0 High Low arbitration sequence. Determines the order for the eFTPE_0 to scan for its next job in the eFTPE_0 High BD rings and Low BD rings. 000 MAPLE-B2 scans the High BD rings of the eFTPE_0 only. 001 MAPLE-B2 scans for next valid BD of the eFTPE_0 as follows: H-L-H-L... that is, after executing one job from any of the High priority rings of the eFTPE_0, it scans for the next job in the Low priority rings of the eFTPE_0. 010 MAPLE-B2 scans for next valid BD of the eFTPE_0 as follows: H-H-L-H-H-L... that is, after executing two jobs from any of the High priority rings of the eFTPE_0, it scans the next job in the Low priority rings of the eFTPE_0. 011 MAPLE-B2 scans for next valid BD of the eFTPE_0 as follows: H-H-H-L-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the eFTPE_0, it scans the next job in the Low priority rings of the eFTPE_0. 100 MAPLE-B2 scans for next valid BD of the eFTPE_0 as follows: H-H-H-H-L-H-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the eFTPE_0, it scans the next job in the Low priority rings of the eFTPE_0. 101 to 111 Reserved
23	Reserved
22–20	F1_R_LOOP.—eFTPE_1 Ring Loop. The MAPLE-B2 assumes the potential valid high/low rings of the eFTPE_1 are located only between ring #0 and ring #(F1_R_LOOP). For details, see **Section 26.4.2.3.2, *BD Arbitration*** 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

Table 26-195. MAPLE BD Rings Configuration Parameter 0 Fields Description (Continued)

Bits	Description
19	Reserved
18–16	F1_HL_ARB—eFTPE_1 High Low arbitration sequence. Determines the order for the eFTPE_1 to scan for its next job in the eFTPE_1 High BD rings and Low BD rings. 000 MAPLE-B2 scans the High BD rings of the eFTPE_1 only. 001 MAPLE-B2 scans for next valid BD of the eFTPE_1 as follows: H-L-H-L... that is, after executing one job from any of the High priority rings of the eFTPE_1, it scans for the next job in the Low priority rings of the eFTPE_1. 010 MAPLE-B2 scans for next valid BD of the eFTPE_1 as follows: H-H-L-H-H-L... that is, after executing two jobs from any of the High priority rings of the eFTPE_1, it scans the next job in the Low priority rings of the eFTPE_1. 011 MAPLE-B2 scans for next valid BD of the eFTPE_1 as follows: H-H-H-L-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the eFTPE_1, it scans the next job in the Low priority rings of the eFTPE_1. 100 MAPLE-B2 scans for next valid BD of the eFTPE_1 as follows: H-H-H-H-L-H-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the eFTPE_1, it scans the next job in the Low priority rings of the eFTPE_1. 101 to 111 Reserved
15	Reserved
14–12	F2_R_LOOP.—eFTPE_2 Ring Loop. The MAPLE-B2 assumes the potential valid high/low rings of the eFTPE_2 are located only between ring #0 and ring #(F2_R_LOOP). For details, see **Section 26.4.2.3.2, *BD Arbitration*** 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
10–8	F2_HL_ARB—eFTPE_2 High Low arbitration sequence. Determines the order for the eFTPE_2 to scan for its next job in the eFTPE_2 High BD rings and Low BD rings. 000 MAPLE-B2 scans the High BD rings of the eFTPE_2 only. 001 MAPLE-B2 scans for next valid BD of the eFTPE_2 as follows: H-L-H-L... that is, after executing one job from any of the High priority rings of the FTPE_2, it scans for the next job in the Low priority rings of the eFTPE_2. 010 MAPLE-B2 scans for next valid BD of the eFTPE_2 as follows: H-H-L-H-H-L... that is, after executing two jobs from any of the High priority rings of the eFTPE_2, it scans the next job in the Low priority rings of the eFTPE_2. 011 MAPLE-B2 scans for next valid BD of the eFTPE_2 as follows: H-H-H-L-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the eFTPE_2, it scans the next job in the Low priority rings of the eFTPE_2. 100 MAPLE-B2 scans for next valid BD of the eFTPE_2 as follows: H-H-H-H-L-H-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the eFTPE_2, it scans the next job in the Low priority rings of the eFTPE_2. 101 to 111 Reserved

Table 26-195. MAPLE BD Rings Configuration Parameter 0 Fields Description (Continued)

Bits	Description
7	Reserved
6–4	TV_R_LOOP—eTVPE Ring Loop. The MAPLE-B2 assumes the potential valid high/low rings of the eTVPE are located only between ring #0 and ring #(TV_R_LOOP). For details, see **Section 26.4.2.3.2, *BD Arbitration***." 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
2–0	TV_HL_ARB—eTVPE High Low arbitration sequence. Determines the order for the eTVPE to scan for its next job in the eTVPE High BD rings and Low BD rings. 000 MAPLE-B2 scans the High BD rings of the eTVPE only. 001 MAPLE-B2 scans for next valid BD of the eTVPE as follows: H-L-H-L... that is, after executing one job from any of the High priority rings of the eTVPE, it scans for the next job in the Low priority rings of the eTVPE. 010 MAPLE-B2 scans for next valid BD of the eTVPE as follows: H-H-L-H-H-L... that is, after executing two jobs from any of the High priority rings of the eTVPE, it scans the next job in the Low priority rings of the eTVPE. 011 MAPLE-B2 scans for next valid BD of the eTVPE as follows: H-H-H-L-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the eTVPE, it scans the next job in the Low priority rings of the eTVPE. 100 MAPLE-B2 scans for next valid BD of the eTVPE as follows: H-H-H-H-L-H-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the eTVPE, it scans the next job in the Low priority rings of the eTVPE. 101 to 111 Reserved

26.5.4.1.2 MAPLE BD Rings Configuration Parameter 1 (MBDRCP1)

This parameter, along with the MBDRCP0 and the MBDRCP2 parameters, determines the number of high/low BD rings used and the arbitration scheme between the high/low BD rings for each of the PEs.

Offset 0x00000008 (MBDRCP1) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		CONV_R_LOOP				CONV_HL_ARB				EQ_R_LOOP				EQ_HL_ARB		
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		DE_R_LOOP				DE_HL_ARB				CRC_R_LOOP				CRC_HL_ARB		
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-242. MAPLE BD Rings Configuration Parameter 1

Table 26-196. MAPLE BD Rings Configuration Parameter 1 Fields Description

Bits	Description
31	Reserved
30–28	CONV_R_LOOP — CONVPE Ring Loop. The MAPLE-B2 assumes the potential valid high/low rings of the CONVPE are located only between ring #0 and ring #(CONV_R_LOOP). For details, see **Section 26.4.2.3.2, *BD Arbitration*** 000 Only ring #0 is potentially valid 001 Ring #0 to ring #1 are potentially valid ... 111Ring #0 to ring #7 are potentially valid
27	Reserved
26–24	CONV_HL_ARB — CONVPE High Low arbitration sequence. Determines the order for the CONVPE to scan for its next job in the CONVPE High BD rings and Low BD rings. 000 MAPLE-B2 scans the High BD rings of the CONVPE only. 001 MAPLE-B2 scans for next valid BD of the CONVPE as follows: H-L-H-L... that is, after executing one job from any of the High priority rings of the CONVPE, it scans for the next job in the Low priority rings of the CONVPE. 010 MAPLE-B2 scans for next valid BD of the CONVPE as follows: H-H-L-H-H-L... that is, after executing two jobs from any of the High priority rings of the CONVPE, it scans the next job in the Low priority rings of the CONVPE. 011 MAPLE-B2 scans for next valid BD of the CONVPE as follows: H-H-H-L-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the CONVPE, it scans the next job in the Low priority rings of the CONVPE. 100 MAPLE-B2 scans for next valid BD of the CONVPE as follows: H-H-H-H-L-H-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the CONVPE, it scans the next job in the Low priority rings of the CONVPE. 101 to 111 Reserved
23	Reserved
22–20	EQ_R_LOOP — EQPE Ring Loop. The MAPLE-B2 assumes the potential valid high/low rings of the EQPE are located only between ring #0 and ring #(EQ_R_LOOP). For details, see **Section 26.4.2.3.2, *BD Arbitration*** 000 Only ring #0 is potentially valid 001 Ring #0 to ring #1 are potentially valid ... 111Ring #0 to ring #7 are potentially valid
19	Reserved

Table 26-196. MAPLE BD Rings Configuration Parameter 1 Fields Description (Continued)

Bits	Description
18–16	EQ_HL_ARB — EQPE High Low arbitration sequence. Determines the order for the EQPE to scan for its next job in the EQPE High BD rings and Low BD rings. 000 MAPLE-B2 scans the High BD rings of the EQPE only. 001 MAPLE-B2 scans for next valid BD of the EQPE as follows: H-L-H-L... that is, after executing one job from any of the High priority rings of the EQPE, it scans for the next job in the Low priority rings of the EQPE. 010 MAPLE-B2 scans for next valid BD of the EQPE as follows: H-H-L-H-H-L... that is, after executing two jobs from any of the High priority rings of the EQPE, it scans the next job in the Low priority rings of the EQPE. 011 MAPLE-B2 scans for next valid BD of the EQPE as follows: H-H-H-L-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the EQPE, it scans the next job in the Low priority rings of the EQPE. 100 MAPLE-B2 scans for next valid BD of the EQPE as follows: H-H-H-H-L-H-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the EQPE, it scans the next job in the Low priority rings of the EQPE. 101 to 111 Reserved
15	Reserved
14–12	DE_R_LOOP — DEPE Ring Loop. The MAPLE-B2 assumes the potential valid high/low rings of the DEPE are located only between ring #0 and ring #(DE_R_LOOP). For details, see **Section 26.4.2.3.2, *BD Arbitration*** 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
10–8	DE_HL_ARB—DEPE High Low arbitration sequence. Determines the order for the DEPE to scan for its next job in the DEPE High BD rings and Low BD rings. 000 MAPLE-B2 scans the High BD rings of the DEPE only. 001 MAPLE-B2 scans for next valid BD of the DEPE as follows: H-L-H-L... that is, after executing one job from any of the High priority rings of the DEPE, it scans for the next job in the Low priority rings of the DEPE. 010 MAPLE-B2 scans for next valid BD of the DEPE as follows: H-H-L-H-H-L... that is, after executing two jobs from any of the High priority rings of the DEPE, it scans the next job in the Low priority rings of the DEPE. 011 MAPLE-B2 scans for next valid BD of the DEPE as follows: H-H-H-L-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the DEPE, it scans the next job in the Low priority rings of the DEPE. 100 MAPLE-B2 scans for next valid BD of the DEPE as follows: H-H-H-H-L-H-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the DEPE, it scans the next job in the Low priority rings of the DEPE. 101 to 111 Reserved
7	Reserved
6–4	CRC_R_LOOP.—CRCPE Ring Loop. The MAPLE-B2 assumes the potential valid high/low rings of the CRCPE are located only between ring #0 and ring #(CRC_R_LOOP). For details, see **Section 26.4.2.3.2, *BD Arbitration*** 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
2–0	CRC_HL_ARB—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-B2 scans the High BD rings of the CRCPE only. 001 MAPLE-B2 scans for next valid BD of the CRCPE as follows: H-L-H-L... that is, 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-B2 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 MAPLE-B2 scans for next valid BD of the CRCPE as follows: H-H-H-L-H-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. 100 MAPLE-B2 scans for next valid BD of the CRCPE as follows: H-H-H-H-L-H-H-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. 101 to 111 Reserved

26.5.4.1.3 MAPLE BD Rings Configuration Parameter 2 (MBDRCP2)

This parameter, along with the MBDRCP0 and MBDRCP1 parameters, determines the number of high/low BD rings used and the arbitration scheme between the high/low BD rings for each of the PEs.

Offset 0x00000008 (MBDRCP2) 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										CG_R_LOOP				CG_HL_ARB		
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-243. MAPLE BD Rings Configuration Parameter 2

Table 26-197. MAPLE BD Rings Configuration Parameter 2 Fields Description

Bits	Description
31–7	Reserved
6–4	CG_R_LOOP — CGPE Ring Loop. The MAPLE-B2 assumes the potential valid high/low rings of the CGPE are located only between ring #0 and ring #(CG_R_LOOP). For details, see **Section 26.4.2.3.2, *BD Arbitration*** 000 Only ring #0 is potentially valid 001 Ring #0 to ring #1 are potentially valid ... 111Ring #0 to ring #7 are potentially valid
3	Reserved
2–0	CG_HL_ARB — CGPE High Low arbitration sequence. Determines the order for the CGPE to scan for its next job in the CGPE High BD rings and Low BD rings. 000 MAPLE-B2 scans the High BD rings of the CGPE only. 001 MAPLE-B2 scans for next valid BD of the CGPE as follows: H-L-H-L... that is, after executing one job from any of the High priority rings of the CGPE, it scans for the next job in the Low priority rings of the CGPE. 010 MAPLE-B2 scans for next valid BD of the CGPE as follows: H-H-L-H-H-L... that is, after executing two jobs from any of the High priority rings of the CGPE, it scans the next job in the Low priority rings of the CGPE. 011 MAPLE-B2 scans for next valid BD of the CGPE as follows: H-H-H-L-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the CGPE, it scans the next job in the Low priority rings of the CGPE. 100 MAPLE-B2 scans for next valid BD of the CGPE as follows: H-H-H-H-L-H-H-H-H-L... that is, after executing two jobs from any of the High priority rings of the CGPE, it scans the next job in the Low priority rings of the CGPE. 101 to 111 Reserved

26.5.4.1.4 MAPLE <yy>PE BD Ring High Priority A <x> Parameter (M<yy>BRHPAxP)

These parameters, along with **M<yy>BRHBxP,** describe the attributes of a High-Priority yyPE BD ring. The <yy> stands for: TV (eTVPE), F0 (eFTPE_0), F1 (eFTPE_1), F2 (eFTPE_2), DE (DEPE), CRC (CRCPE), EQ (EQPE), CONV (CONVPE). There are 8 such parameters for each PE, one for each possible High-Priority BD ring related to the PE. The following table describes all the instantiations of this parameter:

Table 26-198. M<yy>BRHPAxP Instantiation

Offset	Instantiation	Access	Reset Value
0x00000080	MTVBRHPA0P - MAPLE eTVPE BD Ring High Priority A 0 Parameter	R/W	0x0000_0000
0x00000088	MTVBRHPA1P - MAPLE eTVPE BD Ring High Priority A 1 Parameter		
0x00000090	MTVBRHPA2P - MAPLE eTVPE BD Ring High Priority A 2 Parameter		
0x00000098	MTVBRHPA3P - MAPLE eTVPE BD Ring High Priority A 3 Parameter		
0x000000A0	MTVBRHPA4P - MAPLE eTVPE BD Ring High Priority A 4 Parameter		
0x000000A8	MTVBRHPA5P - MAPLE eTVPE BD Ring High Priority A 5 Parameter		
0x000000B0	MTVBRHPA6P - MAPLE eTVPE BD Ring High Priority A 6 Parameter		
0x000000B8	MTVBRHPA7P - MAPLE eTVPE BD Ring High Priority A 7 Parameter		
0x00000100	MF0BRHPA0P - MAPLE EFTPE_0 BD Ring High Priority A 0 Parameter	R/W	0x0000_0000
0x00000108	MF0BRHPA1P - MAPLE EFTPE_0 BD Ring High Priority A 1 Parameter		
0x00000110	MF0BRHPA2P - MAPLE EFTPE_0 BD Ring High Priority A 2 Parameter		
0x00000118	MF0BRHPA3P - MAPLE EFTPE_0 BD Ring High Priority A 3 Parameter		
0x00000120	MF0BRHPA4P - MAPLE EFTPE_0 BD Ring High Priority A 4 Parameter		
0x00000128	MF0BRHPA5P - MAPLE EFTPE_0 BD Ring High Priority A 5 Parameter		
0x00000130	MF0BRHPA6P - MAPLE EFTPE_0 BD Ring High Priority A 6 Parameter		
0x00000138	MF0BRHPA7P - MAPLE EFTPE_0 BD Ring High Priority A 7 Parameter		
0x00000180	MF1BRHPA0P - MAPLE EFTPE_1 BD Ring High Priority A 0 Parameter	R/W	0x0000_0000
0x00000188	MF1BRHPA1P - MAPLE EFTPE_1 BD Ring High Priority A 1 Parameter		
0x00000190	MF1BRHPA2P - MAPLE EFTPE_1 BD Ring High Priority A 2 Parameter		
0x00000198	MF1BRHPA3P - MAPLE EFTPE_1 BD Ring High Priority A 3 Parameter		
0x000001A0	MF1BRHPA4P - MAPLE EFTPE_1 BD Ring High Priority A 4 Parameter		
0x000001A8	MF1BRHPA5P - MAPLE EFTPE_1 BD Ring High Priority A 5 Parameter		
0x000001B0	MF1BRHPA6P - MAPLE EFTPE_1 BD Ring High Priority A 6 Parameter		
0x000001B8	MF1BRHPA7P - MAPLE EFTPE_1 BD Ring High Priority A 7 Parameter		
0x00000200	MF2BRHPA0P - MAPLE EFTPE_2 BD Ring High Priority A 0 Parameter	R/W	0x0000_0000
0x00000208	MF2BRHPA1P - MAPLE EFTPE_2 BD Ring High Priority A 1 Parameter		
0x00000210	MF2BRHPA2P - MAPLE EFTPE_2 BD Ring High Priority A 2 Parameter		
0x00000218	MF2BRHPA3P - MAPLE EFTPE_2 BD Ring High Priority A 3 Parameter		
0x00000220	MF2BRHPA4P - MAPLE EFTPE_2 BD Ring High Priority A 4 Parameter		
0x00000228	MF2BRHPA5P - MAPLE EFTPE_2 BD Ring High Priority A 5 Parameter		
0x00000230	MF2BRHPA6P - MAPLE EFTPE_2 BD Ring High Priority A 6 Parameter		
0x00000238	MF2BRHPA7P - MAPLE EFTPE_2 BD Ring High Priority A 7 Parameter		

Table 26-198. M<yy>BRHPAxP Instantiation

Offset	Instantiation	Access	Reset Value
0x00000280	MDEBRHPA0P - MAPLE DEPE BD Ring High Priority A 0 Parameter	R/W	0x0000_0000
0x00000288	MDEBRHPA1P - MAPLE DEPE BD Ring High Priority A 1 Parameter		
0x00000290	MDEBRHPA2P - MAPLE DEPE BD Ring High Priority A 2 Parameter		
0x00000298	MDEBRHPA3P - MAPLE DEPE BD Ring High Priority A 3 Parameter		
0x000002A0	MDEBRHPA4P - MAPLE DEPE BD Ring High Priority A 4 Parameter		
0x000002A8	MDEBRHPA5P - MAPLE DEPE BD Ring High Priority A 5 Parameter		
0x000002B0	MDEBRHPA6P - MAPLE DEPE BD Ring High Priority A 6 Parameter		
0x000002B8	MDEBRHPA7P - MAPLE DEPE BD Ring High Priority A 7 Parameter		
0x00000300	MCRCBRHPA0P - MAPLE CRCPE BD Ring High Priority A 0 Parameter	R/W	0x0000_0000
0x00000308	MCRCBRHPA1P - MAPLE CRCPE BD Ring High Priority A 1 Parameter		
0x00000310	MCRCBRHPA2P - MAPLE CRCPE BD Ring High Priority A 2 Parameter		
0x00000318	MCRCBRHPA3P - MAPLE CRCPE BD Ring High Priority A 3 Parameter		
0x00000320	MCRCBRHPA4P - MAPLE CRCPE BD Ring High Priority A 4 Parameter		
0x00000328	MCRCBRHPA5P - MAPLE CRCPE BD Ring High Priority A 5 Parameter		
0x00000330	MCRCBRHPA6P - MAPLE CRCPE BD Ring High Priority A 6 Parameter		
0x00000338	MCRCBRHPA7P - MAPLE CRCPE BD Ring High Priority A 7 Parameter		
0x00000380	MEQBRHPA0P - MAPLE EQPE BD Ring High Priority A 0 Parameter	R/W	0x0000_0000
0x00000388	MEQBRHPA1P - MAPLE EQPE BD Ring High Priority A 1 Parameter		
0x00000390	MEQBRHPA2P - MAPLE EQPE BD Ring High Priority A 2 Parameter		
0x00000398	MEQBRHPA3P - MAPLE EQPE BD Ring High Priority A 3 Parameter		
0x000003A0	MEQBRHPA4P - MAPLE EQPE BD Ring High Priority A 4 Parameter		
0x000003A8	MEQBRHPA5P - MAPLE EQPE BD Ring High Priority A 5 Parameter		
0x000003B0	MEQBRHPA6P - MAPLE EQPE BD Ring High Priority A 6 Parameter		
0x000003B8	MEQBRHPA7P - MAPLE EQPE BD Ring High Priority A 7 Parameter		
0x00000400	MCGBRHPA0P - MAPLE CGPE BD Ring High Priority A 0 Parameter	R/W	0x0000_0000
0x00000408	MCGBRHPA1P - MAPLE CGPE BD Ring High Priority A 1 Parameter		
0x00000410	MCGBRHPA2P - MAPLE CGPE BD Ring High Priority A 2 Parameter		
0x00000418	MCGBRHPA3P - MAPLE CGPE BD Ring High Priority A 3 Parameter		
0x00000420	MCGBRHPA4P - MAPLE CGPE BD Ring High Priority A 4 Parameter		
0x00000428	MCGBRHPA5P - MAPLE CGPE BD Ring High Priority A 5 Parameter		
0x00000430	MCGBRHPA6P - MAPLE CGPE BD Ring High Priority A 6 Parameter		
0x00000438	MCGBRHPA7P - MAPLE CGPE BD Ring High Priority A 7 Parameter		
0x00000480	MCONVBRHPA0P - MAPLE CONVPE BD Ring High Priority A 0 Parameter	R/W	0x0000_0000
0x00000488	MCONVBRHPA1P - MAPLE CONVPE BD Ring High Priority A 1 Parameter		
0x00000490	MCONVBRHPA2P - MAPLE CONVPE BD Ring High Priority A 2 Parameter		
0x00000498	MCONVBRHPA3P - MAPLE CONVPE BD Ring High Priority A 3 Parameter		
0x000004A0	MCONVBRHPA4P - MAPLE CONVPE BD Ring High Priority A 4 Parameter		
0x000004A8	MCONVBRHPA5P - MAPLE CONVPE BD Ring High Priority A 5 Parameter		
0x000004B0	MCONVBRHPA6P - MAPLE CONVPE BD Ring High Priority A 6 Parameter		
0x000004B8	MCONVBRHPA7P - MAPLE CONVPE BD Ring High Priority A 7 Parameter		

Figure 26-244 describe the M<yy>BRHPAxP structure:

Offset see **Table 26-198** Access: User read/write

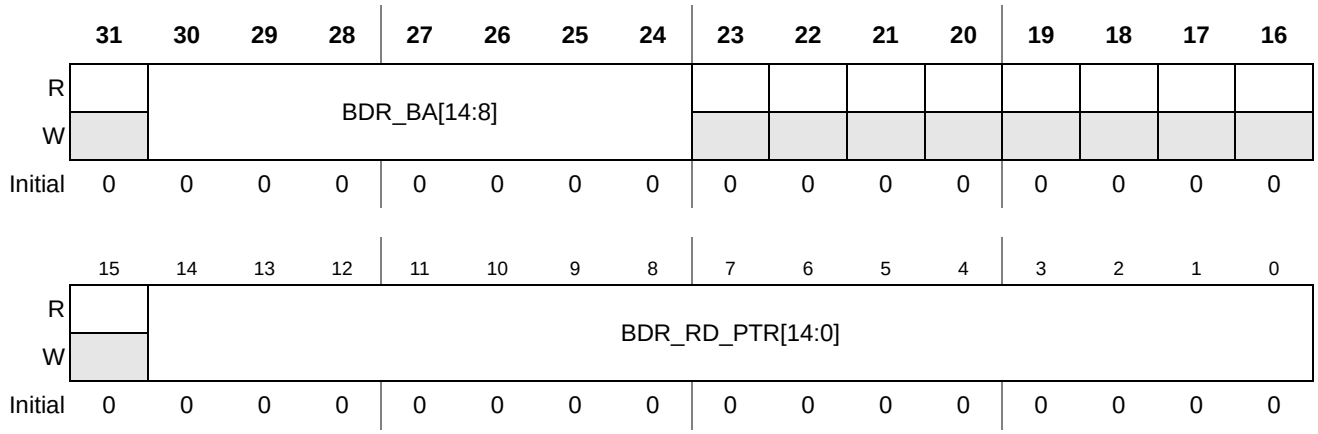


Figure 26-244. MAPLE <yy>PE BD Ring High Priority A <x> Parameter

Table 26-199. MAPLE <yy>PE BD Ring High Priority A <x> Parameter Fields Description

Bits	Description
31	Reserved
30–24	BDR_BA[14:8]—BD Ring Base Address. Points to the base address of the BD ring in the PSIF2 DRAM. The address must be aligned to 256 bytes. Therefore the field range is 14:8. The B.A. should be within the 32KBytes of BD memory in the DRAM. The given address is relative to the start address of the BD memory in the DRAM. 0x00 to 0x7F BD Ring Base Address for <yy>PE High Priority Ring x. (stands for base address 0x0000 to 0x7F00)
23–15	Reserved
14–0	BDR_RD_PTR[14:0]—BD Ring Read Pointer. This pointer points to the current BD in the ring the MAPLE-B2 is currently processing. It is increment by the MAPLE-B2 after BD has started processing. The host can use this pointer to track the MAPLE-B2's 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 (BDR_RD_PTR[14:0] = {BDR_BA[14:8],0...0}), and must not be overwritten once the operation starts. Altering the contents of this pointer by the host results in 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: 0x0000 to 0x7FFF Read Pointer for <yy>PE High Priority Ring x

26.5.4.1.5 MAPLE <yy>PE BD Ring High Priority B <x> Parameter (M<yy>BRHPBxP)

These parameters, along with **M<yy>BRHAxP,** describe the attributes of a High-Priority yyPE BD ring. The <yy> stands for: TV (eTVPE), F0 (eFTPE_0), F1 (eFTPE_1), F2 (eFTPE_2), DE (DEPE), CRC (CRCPE), EQ (EQPE), CONV (CONVPE). There are 8 such parameters for each PE, one for each possible High-Priority BD ring related to the PE. The following table describes all the instantiations of this parameter:

Table 26-200. M<yy>BRHPBxP Instantiation

Offset	Instantiation	Access	Reset Value
0x00000084	MTVBRHPB0P - MAPLE eTVPE BD Ring High Priority B 0 Parameter	R/W	0x0000_0000
0x0000008C	MTVBRHPB1P - MAPLE eTVPE BD Ring High Priority B 1 Parameter		
0x00000094	MTVBRHPB2P - MAPLE eTVPE BD Ring High Priority B 2 Parameter		
0x0000009C	MTVBRHPB3P - MAPLE eTVPE BD Ring High Priority B 3 Parameter		
0x000000A4	MTVBRHPB4P - MAPLE eTVPE BD Ring High Priority B 4 Parameter		
0x000000AC	MTVBRHPB5P - MAPLE eTVPE BD Ring High Priority B 5 Parameter		
0x000000B4	MTVBRHPB6P - MAPLE eTVPE BD Ring High Priority B 6 Parameter		
0x000000BC	MTVBRHPB7P - MAPLE eTVPE BD Ring High Priority B 7 Parameter		
0x00000104	MF0BRHPB0P - MAPLE EFTPE_0 BD Ring High Priority B 0 Parameter	R/W	0x0000_0000
0x0000010C	MF0BRHPB1P - MAPLE EFTPE_0 BD Ring High Priority B 1 Parameter		
0x00000114	MF0BRHPB2P - MAPLE EFTPE_0 BD Ring High Priority B 2 Parameter		
0x0000011C	MF0BRHPB3P - MAPLE EFTPE_0 BD Ring High Priority B 3 Parameter		
0x00000124	MF0BRHPB4P - MAPLE EFTPE_0 BD Ring High Priority B 4 Parameter		
0x0000012C	MF0BRHPB5P - MAPLE EFTPE_0 BD Ring High Priority B 5 Parameter		
0x00000134	MF0BRHPB6P - MAPLE EFTPE_0 BD Ring High Priority B 6 Parameter		
0x0000013C	MF0BRHPB7P - MAPLE EFTPE_0 BD Ring High Priority B 7 Parameter		
0x00000184	MF1BRHPB0P - MAPLE EFTPE_1 BD Ring High Priority B 0 Parameter	R/W	0x0000_0000
0x0000018C	MF1BRHPB1P - MAPLE EFTPE_1 BD Ring High Priority B 1 Parameter		
0x00000194	MF1BRHPB2P - MAPLE EFTPE_1 BD Ring High Priority B 2 Parameter		
0x0000019C	MF1BRHPB3P - MAPLE EFTPE_1 BD Ring High Priority B 3 Parameter		
0x000001A4	MF1BRHPB4P - MAPLE EFTPE_1 BD Ring High Priority B 4 Parameter		
0x000001AC	MF1BRHPB5P - MAPLE EFTPE_1 BD Ring High Priority B 5 Parameter		
0x000001B4	MF1BRHPB6P - MAPLE EFTPE_1 BD Ring High Priority B 6 Parameter		
0x000001BC	MF1BRHPB7P - MAPLE EFTPE_1 BD Ring High Priority B 7 Parameter		
0x00000204	MF2BRHPB0P - MAPLE EFTPE_2 BD Ring High Priority B 0 Parameter	R/W	0x0000_0000
0x0000020C	MF2BRHPB1P - MAPLE EFTPE_2 BD Ring High Priority B 1 Parameter		
0x00000214	MF2BRHPB2P - MAPLE EFTPE_2 BD Ring High Priority B 2 Parameter		
0x0000021C	MF2BRHPB3P - MAPLE EFTPE_2 BD Ring High Priority B 3 Parameter		
0x00000224	MF2BRHPB4P - MAPLE EFTPE_2 BD Ring High Priority B 4 Parameter		
0x0000022C	MF2BRHPB5P - MAPLE EFTPE_2 BD Ring High Priority B 5 Parameter		
0x00000234	MF2BRHPB6P - MAPLE EFTPE_2 BD Ring High Priority B 6 Parameter		
0x0000023C	MF2BRHPB7P - MAPLE EFTPE_2 BD Ring High Priority B 7 Parameter		

Table 26-200. M<yy>BRHPBxP Instantiation

Offset	Instantiation	Access	Reset Value
0x00000284	MDEBRHPB0P - MAPLE DEPE BD Ring High Priority B 0 Parameter	R/W	0x0000_0000
0x0000028C	MDEBRHPB1P - MAPLE DEPE BD Ring High Priority B 1 Parameter		
0x00000294	MDEBRHPB2P - MAPLE DEPE BD Ring High Priority B 2 Parameter		
0x0000029C	MDEBRHPB3P - MAPLE DEPE BD Ring High Priority B 3 Parameter		
0x000002A4	MDEBRHPB4P - MAPLE DEPE BD Ring High Priority B 4 Parameter		
0x000002AC	MDEBRHPB5P - MAPLE DEPE BD Ring High Priority B 5 Parameter		
0x000002B4	MDEBRHPB6P - MAPLE DEPE BD Ring High Priority B 6 Parameter		
0x000002BC	MDEBRHPB7P - MAPLE DEPE BD Ring High Priority B 7 Parameter		
0x00000304	MCRCBRHPB0P - MAPLE CRCPE BD Ring High Priority B 0 Parameter	R/W	0x0000_0000
0x0000030C	MCRCBRHPB1P - MAPLE CRCPE BD Ring High Priority B 1 Parameter		
0x00000314	MCRCBRHPB2P - MAPLE CRCPE BD Ring High Priority B 2 Parameter		
0x0000031C	MCRCBRHPB3P - MAPLE CRCPE BD Ring High Priority B 3 Parameter		
0x00000324	MCRCBRHPB4P - MAPLE CRCPE BD Ring High Priority B 4 Parameter		
0x0000032C	MCRCBRHPB5P - MAPLE CRCPE BD Ring High Priority B 5 Parameter		
0x00000334	MCRCBRHPB6P - MAPLE CRCPE BD Ring High Priority B 6 Parameter		
0x0000033C	MCRCBRHPB7P - MAPLE CRCPE BD Ring High Priority B 7 Parameter		
0x00000384	MEQBRHPB0P - MAPLE EQPE BD Ring High Priority B 0 Parameter	R/W	0x0000_0000
0x0000038C	MEQBRHPB1P - MAPLE EQPE BD Ring High Priority B 1 Parameter		
0x00000394	MEQBRHPB2P - MAPLE EQPE BD Ring High Priority B 2 Parameter		
0x0000039C	MEQBRHPB3P - MAPLE EQPE BD Ring High Priority B 3 Parameter		
0x000003A4	MEQBRHPB4P - MAPLE EQPE BD Ring High Priority B 4 Parameter		
0x000003AC	MEQBRHPB5P - MAPLE EQPE BD Ring High Priority B 5 Parameter		
0x000003B4	MEQBRHPB6P - MAPLE EQPE BD Ring High Priority B 6 Parameter		
0x000003BC	MEQBRHPB7P - MAPLE EQPE BD Ring High Priority B 7 Parameter		
0x00000404	MCGBRHPB0P - MAPLE CGPE BD Ring High Priority B 0 Parameter	R/W	0x0000_0000
0x0000040C	MCGBRHPB1P - MAPLE CGPE BD Ring High Priority B 1 Parameter		
0x00000414	MCGBRHPB2P - MAPLE CGPE BD Ring High Priority B 2 Parameter		
0x0000041C	MCGBRHPB3P - MAPLE CGPE BD Ring High Priority B 3 Parameter		
0x00000424	MCGBRHPB4P - MAPLE CGPE BD Ring High Priority B 4 Parameter		
0x0000042C	MCGBRHPB5P - MAPLE CGPE BD Ring High Priority B 5 Parameter		
0x00000434	MCGBRHPB6P - MAPLE CGPE BD Ring High Priority B 6 Parameter		
0x0000043C	MCGBRHPB7P - MAPLE CGPE BD Ring High Priority B 7 Parameter		
0x00000484	MCONVBRHPB0P - MAPLE CONVPE BD Ring High Priority B 0 Parameter	R/W	0x0000_0000
0x0000048C	MCONVBRHPB1P - MAPLE CONVPE BD Ring High Priority B 1 Parameter		
0x00000494	MCONVBRHPB2P - MAPLE CONVPE BD Ring High Priority B 2 Parameter		
0x0000049C	MCONVBRHPB3P - MAPLE CONVPE BD Ring High Priority B 3 Parameter		
0x000004A4	MCONVBRHPB4P - MAPLE CONVPE BD Ring High Priority B 4 Parameter		
0x000004AC	MCONVBRHPB5P - MAPLE CONVPE BD Ring High Priority B 5 Parameter		
0x000004B4	MCONVBRHPB6P - MAPLE CONVPE BD Ring High Priority B 6 Parameter		
0x000004BC	MCONVBRHPB7P - MAPLE CONVPE BD Ring High Priority B 7 Parameter		

Figure 26-245 describe the M<yy>BRHPBxP structure:

Offset see **Table 26-200** Access: User read/write

	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 26-245. MAPLE <yy>PE BD Ring High Priority B <x> Parameter

Table 26-201. MAPLE <yy>PE BD Ring High Priority B <x> Parameter Fields Description

Bits	Description
31–16	HOST_ID—16 bits of Host ID for Serial RapidIO door-bell support for an External (to DSP device) master connected to the DSP device by Serial RapidIO port and is the owner of this ring. This field is valid only if MTVBRHPBxP[EXT_MST] equals ‘01’ or’10’.
15–12	P_TR_IF—Port Target Interface. Determines which RapidIO port to use for this doorbell. This field is used by the MAPLE-B2 for the Word4[TINT] field in the Type10 Outbell Doorbell descriptor.
11–10	EXT_MST—External Master. Describes if the Ring owner can receive any of the MAPLE-B2 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 DSP device via Serial RapidIO port and the Serial RapidIO door-bell interrupt is generated if the [INT_EN] bit in the BD is set. For details see **Section 26.4.4, *External Masters Support Using Serial RapidIO Doorbell*.”** ‘00’ The master of this ring can receive any of the MAPLE-Bs regular interrupts, therefore a regular interrupt is generated according to MTVBRHPBxP[INT_TRGT]. ’01’ The master of this ring is External (connected via Serial RapidIO port), therefore a Serial RapidIO door-bell interrupt is generated. The door-bell should be initiated via Serial RapidIO port 1 (**Section 26.4.4, *External Masters Support Using Serial RapidIO Doorbell***”). ‘10’ The master of this ring is External (connected via Serial RapidIO port), therefore a Serial RapidIO door-bell interrupt is generated. The door-bell should be initiated via Serial RapidIO port 2 (**Section 26.4.4, *External Masters Support Using Serial RapidIO Doorbell***”). ‘11’ Reserved
9	Reserved.
8–4	INT_TRGT—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 ’00’. 00000 irq0 00001 irq1 ... 11111 irq31
3–1	Reserved
0	VLD—Valid Bit. Describes if the current BD ring is valid. The MAPLE-B2 uses this bit to determine the BD Ring’s validation, therefore any change in any of the BD Ring’s parameters must be done while the VLD bit is cleared. 0 BD ring is not valid. 1 BD ring is valid.

26.5.4.1.6 MAPLE <yy>PE BD Ring Low Priority A <x> Parameter (M<yy>BRLPAxP)

These parameters, along with **M<yy>BRLBxP,** describe the attributes of a Low-Priority <yy>PE BD ring. The <yy> stands for: TV (eTVPE), F0 (eFTPE_0), F1 (eFTPE_1), F2 (eFTPE_2), DE (DEPE), CRC (CRCPE), EQ (EQPE), CONV (CONVPE). There are 8 such parameters for each PE, one for each possible Low-Priority BD ring related to the PE. The following table details all the instantiations of this parameter:

Table 26-202. M<yy>BRLPAxP Instantiation

Offset	Instantiation	Access	Reset Value
0x000000C0	MTVBRLPA0P - MAPLE eTVPE BD Ring Low Priority A 0 Parameter	R/W	0x0000_0000
0x000000C8	MTVBRLPA1P - MAPLE eTVPE BD Ring Low Priority A 1 Parameter		
0x000000D0	MTVBRLPA2P - MAPLE eTVPE BD Ring Low Priority A 2 Parameter		
0x000000D8	MTVBRLPA3P - MAPLE eTVPE BD Ring Low Priority A 3 Parameter		
0x000000E0	MTVBRLPA4P - MAPLE eTVPE BD Ring Low Priority A 4 Parameter		
0x000000E8	MTVBRLPA5P - MAPLE eTVPE BD Ring Low Priority A 5 Parameter		
0x000000F0	MTVBRLPA6P - MAPLE eTVPE BD Ring Low Priority A 6 Parameter		
0x000000F8	MTVBRLPA7P - MAPLE eTVPE BD Ring Low Priority A 7 Parameter		
0x00000140	MF0BRLPA0P - MAPLE EFTPE_0 BD Ring Low Priority A 0 Parameter	R/W	0x0000_0000
0x00000148	MF0BRLPA1P - MAPLE EFTPE_0 BD Ring Low Priority A 1 Parameter		
0x00000150	MF0BRLPA2P - MAPLE EFTPE_0 BD Ring Low Priority A 2 Parameter		
0x00000158	MF0BRLPA3P - MAPLE EFTPE_0 BD Ring Low Priority A 3 Parameter		
0x00000160	MF0BRLPA4P - MAPLE EFTPE_0 BD Ring Low Priority A 4 Parameter		
0x00000168	MF0BRLPA5P - MAPLE EFTPE_0 BD Ring Low Priority A 5 Parameter		
0x00000170	MF0BRLPA6P - MAPLE EFTPE_0 BD Ring Low Priority A 6 Parameter		
0x00000178	MF0BRLPA7P - MAPLE EFTPE_0 BD Ring Low Priority A 7 Parameter		
0x000001C0	MF1BRLPA0P - MAPLE EFTPE_1 BD Ring Low Priority A 0 Parameter	R/W	0x0000_0000
0x000001C8	MF1BRLPA1P - MAPLE EFTPE_1 BD Ring Low Priority A 1 Parameter		
0x000001D0	MF1BRLPA2P - MAPLE EFTPE_1 BD Ring Low Priority A 2 Parameter		
0x000001D8	MF1BRLPA3P - MAPLE EFTPE_1 BD Ring Low Priority A 3 Parameter		
0x000001E0	MF1BRLPA4P - MAPLE EFTPE_1 BD Ring Low Priority A 4 Parameter		
0x000001E8	MF1BRLPA5P - MAPLE EFTPE_1 BD Ring Low Priority A 5 Parameter		
0x000001F0	MF1BRLPA6P - MAPLE EFTPE_1 BD Ring Low Priority A 6 Parameter		
0x000001F8	MF1BRLPA7P - MAPLE EFTPE_1 BD Ring Low Priority A 7 Parameter		
0x00000240	MF2BRLPA0P - MAPLE EFTPE_2 BD Ring Low Priority A 0 Parameter	R/W	0x0000_0000
0x00000248	MF2BRLPA1P - MAPLE EFTPE_2 BD Ring Low Priority A 1 Parameter		
0x00000250	MF2BRLPA2P - MAPLE EFTPE_2 BD Ring Low Priority A 2 Parameter		
0x00000258	MF2BRLPA3P - MAPLE EFTPE_2 BD Ring Low Priority A 3 Parameter		
0x00000260	MF2BRLPA4P - MAPLE EFTPE_2 BD Ring Low Priority A 4 Parameter		
0x00000268	MF2BRLPA5P - MAPLE EFTPE_2 BD Ring Low Priority A 5 Parameter		
0x00000270	MF2BRLPA6P - MAPLE EFTPE_2 BD Ring Low Priority A 6 Parameter		
0x00000278	MF2BRLPA7P - MAPLE EFTPE_2 BD Ring Low Priority A 7 Parameter		

Table 26-202. M<yy>BRLPAxP Instantiation

Offset	Instantiation	Access	Reset Value
0x000002C0	MDEBRLPA0P - MAPLE DEPE BD Ring Low Priority A 0 Parameter	R/W	0x0000_0000
0x000002C8	MDEBRLPA1P - MAPLE DEPE BD Ring Low Priority A 1 Parameter		
0x000002D0	MDEBRLPA2P - MAPLE DEPE BD Ring Low Priority A 2 Parameter		
0x000002D8	MDEBRLPA3P - MAPLE DEPE BD Ring Low Priority A 3 Parameter		
0x000002E0	MDEBRLPA4P - MAPLE DEPE BD Ring Low Priority A 4 Parameter		
0x000002E8	MDEBRLPA5P - MAPLE DEPE BD Ring Low Priority A 5 Parameter		
0x000002F0	MDEBRLPA6P - MAPLE DEPE BD Ring Low Priority A 6 Parameter		
0x000002F8	MDEBRLPA7P - MAPLE DEPE BD Ring Low Priority A 7 Parameter		
0x00000340	MCRCBRLPA0P - MAPLE CRCPE BD Ring Low Priority A 0 Parameter	R/W	0x0000_0000
0x00000348	MCRCBRLPA1P - MAPLE CRCPE BD Ring Low Priority A 1 Parameter		
0x00000350	MCRCBRLPA2P - MAPLE CRCPE BD Ring Low Priority A 2 Parameter		
0x00000358	MCRCBRLPA3P - MAPLE CRCPE BD Ring Low Priority A 3 Parameter		
0x00000360	MCRCBRLPA4P - MAPLE CRCPE BD Ring Low Priority A 4 Parameter		
0x00000368	MCRCBRLPA5P - MAPLE CRCPE BD Ring Low Priority A 5 Parameter		
0x00000370	MCRCBRLPA6P - MAPLE CRCPE BD Ring Low Priority A 6 Parameter		
0x00000378	MCRCBRLPA7P - MAPLE CRCPE BD Ring Low Priority A 7 Parameter		
0x000003C0	MEQBRLPA0P - MAPLE EQPE BD Ring Low Priority A 0 Parameter	R/W	0x0000_0000
0x000003C8	MEQBRLPA1P - MAPLE EQPE BD Ring Low Priority A 1 Parameter		
0x000003D0	MEQBRLPA2P - MAPLE EQPE BD Ring Low Priority A 2 Parameter		
0x000003D8	MEQBRLPA3P - MAPLE EQPE BD Ring Low Priority A 3 Parameter		
0x000003E0	MEQBRLPA4P - MAPLE EQPE BD Ring Low Priority A 4 Parameter		
0x000003E8	MEQBRLPA5P - MAPLE EQPE BD Ring Low Priority A 5 Parameter		
0x000003F0	MEQBRLPA6P - MAPLE EQPE BD Ring Low Priority A 6 Parameter		
0x000003F8	MEQBRLPA7P - MAPLE EQPE BD Ring Low Priority A 7 Parameter		
0x00000440	MCGBRLPA0P - MAPLE CGPE BD Ring Low Priority A 0 Parameter	R/W	0x0000_0000
0x00000448	MCGBRLPA1P - MAPLE CGPE BD Ring Low Priority A 1 Parameter		
0x00000450	MCGBRLPA2P - MAPLE CGPE BD Ring Low Priority A 2 Parameter		
0x00000458	MCGBRLPA3P - MAPLE CGPE BD Ring Low Priority A 3 Parameter		
0x00000460	MCGBRLPA4P - MAPLE CGPE BD Ring Low Priority A 4 Parameter		
0x00000468	MCGBRLPA5P - MAPLE CGPE BD Ring Low Priority A 5 Parameter		
0x00000470	MCGBRLPA6P - MAPLE CGPE BD Ring Low Priority A 6 Parameter		
0x00000478	MCGBRLPA7P - MAPLE CGPE BD Ring Low Priority A 7 Parameter		
0x000004C0	MCONVBRLPA0P - MAPLE CONVPE BD Ring Low Priority A 0 Parameter	R/W	0x0000_0000
0x000004C8	MCONVBRLPA1P - MAPLE CONVPE BD Ring Low Priority A 1 Parameter		
0x000004D0	MCONVBRLPA2P - MAPLE CONVPE BD Ring Low Priority A 2 Parameter		
0x000004D8	MCONVBRLPA3P - MAPLE CONVPE BD Ring Low Priority A 3 Parameter		
0x000004E0	MCONVBRLPA4P - MAPLE CONVPE BD Ring Low Priority A 4 Parameter		
0x000004E8	MCONVBRLPA5P - MAPLE CONVPE BD Ring Low Priority A 5 Parameter		
0x000004F0	MCONVBRLPA6P - MAPLE CONVPE BD Ring Low Priority A 6 Parameter		
0x000004F8	MCONVBRLPA7P - MAPLE CONVPE BD Ring Low Priority A 7 Parameter		

Figure 26-246 describe the M<yy>BRLPAxP structure:

Offset see **Table 26-202** Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		BDR_BA[14:8]														
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[14:0]														
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-246. MAPLE <yy>PE BD Ring Low Priority A <x> Parameter

Table 26-203. MAPLE <YY>PE BD Ring Low Priority A <x> Parameter Fields Description

Bits	Description
31	Reserved
30–24	BDR_BA[14:8]—BD Ring Base Address. Points to the base address of the BD ring in the PSIF2 DRAM. The address must be aligned to 256 bytes, and therefore, the field range is 14:8. The B.A should be within the 32Kbytes of BD memory in the DRAM. The given address is relative to the start address of the BD memory in the DRAM. 0x00 to 0x7F BD Ring Base Address for <yy>PE Low Priority Ring x. (stands for base address 0x0000 to 0x7FFF)
23–15	Reserved
14–3	BDR_RD_PTR[14:0]—BD Ring Read Pointer. This pointer points to the current BD in the ring the MAPLE-B2 is currently processing. It is post-increment by the MAPLE-B2, after BD has started processing. The host can use this pointer to track the MAPLE-B2's 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[14:0] = {BDR_BA[14:8],0...0}), and must not be overwritten once the operation starts. Altering the contents of this pointer by the host results in 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: 0x0000 to 0x7F00 Read Pointer for <yy>PE Low Priority Ring x
2–0	Reserved

26.5.4.1.7 MAPLE <yy>PE BD Ring Low Priority B <x> Parameter (M<yy>BRLPBxP)

These parameters, along with **M<yy>BRLAxP,** describe the attributes of a Low-Priority <yy>PE BD ring. The <yy> stands for: TV (eTVPE), F0 (eFTPE_0), F1 (eFTPE_1), F2 (eFTPE_2), DE (DEPE), CRC (CRCPE), EQ (EQPE), CONV (CONVPE). There are 8 such parameters for each PE, one for each possible Low-Priority BD ring related to the PE. The following table details all the instantiations of this parameter:

Table 26-204. M<yy>BRLPBxP Instantiation

Offset	Instantiation	Access	Reset Value
0x000000C4	MTVBRLPB0P - MAPLE eTVPE BD Ring Low Priority B 0 Parameter	R/W	0x0000_0000
0x000000CC	MTVBRLPB1P - MAPLE eTVPE BD Ring Low Priority B 1 Parameter		
0x000000D4	MTVBRLPB2P - MAPLE eTVPE BD Ring Low Priority B 2 Parameter		
0x000000DC	MTVBRLPB3P - MAPLE eTVPE BD Ring Low Priority B 3 Parameter		
0x000000E4	MTVBRLPB4P - MAPLE eTVPE BD Ring Low Priority B 4 Parameter		
0x000000EC	MTVBRLPB5P - MAPLE eTVPE BD Ring Low Priority B 5 Parameter		
0x000000F4	MTVBRLPB6P - MAPLE eTVPE BD Ring Low Priority B 6 Parameter		
0x000000FC	MTVBRLPB7P - MAPLE eTVPE BD Ring Low Priority B 7 Parameter		
0x00000144	MF0BRLPB0P - MAPLE EFTPE_0 BD Ring Low Priority B 0 Parameter	R/W	0x0000_0000
0x0000014C	MF0BRLPB1P - MAPLE EFTPE_0 BD Ring Low Priority B 1 Parameter		
0x00000154	MF0BRLPB2P - MAPLE EFTPE_0 BD Ring Low Priority B 2 Parameter		
0x0000015C	MF0BRLPB3P - MAPLE EFTPE_0 BD Ring Low Priority B 3 Parameter		
0x00000164	MF0BRLPB4P - MAPLE EFTPE_0 BD Ring Low Priority B 4 Parameter		
0x0000016C	MF0BRLPB5P - MAPLE EFTPE_0 BD Ring Low Priority B 5 Parameter		
0x00000174	MF0BRLPB6P - MAPLE EFTPE_0 BD Ring Low Priority B 6 Parameter		
0x0000017C	MF0BRLPB7P - MAPLE EFTPE_0 BD Ring Low Priority B 7 Parameter		
0x000001C4	MF1BRLPB0P - MAPLE EFTPE_1 BD Ring Low Priority B 0 Parameter	R/W	0x0000_0000
0x000001CC	MF1BRLPB1P - MAPLE EFTPE_1 BD Ring Low Priority B 1 Parameter		
0x000001D4	MF1BRLPB2P - MAPLE EFTPE_1 BD Ring Low Priority B 2 Parameter		
0x000001DC	MF1BRLPB3P - MAPLE EFTPE_1 BD Ring Low Priority B 3 Parameter		
0x000001E4	MF1BRLPB4P - MAPLE EFTPE_1 BD Ring Low Priority B 4 Parameter		
0x000001EC	MF1BRLPB5P - MAPLE EFTPE_1 BD Ring Low Priority B 5 Parameter		
0x000001F4	MF1BRLPB6P - MAPLE EFTPE_1 BD Ring Low Priority B 6 Parameter		
0x000001FC	MF1BRLPB7P - MAPLE EFTPE_1 BD Ring Low Priority B 7 Parameter		
0x00000244	MF2BRLPB0P - MAPLE EFTPE_2 BD Ring Low Priority B 0 Parameter	R/W	0x0000_0000
0x0000024C	MF2BRLPB1P - MAPLE EFTPE_2 BD Ring Low Priority B 1 Parameter		
0x00000254	MF2BRLPB2P - MAPLE EFTPE_2 BD Ring Low Priority B 2 Parameter		
0x0000025C	MF2BRLPB3P - MAPLE EFTPE_2 BD Ring Low Priority B 3 Parameter		
0x00000264	MF2BRLPB4P - MAPLE EFTPE_2 BD Ring Low Priority B 4 Parameter		
0x0000026C	MF2BRLPB5P - MAPLE EFTPE_2 BD Ring Low Priority B 5 Parameter		
0x00000274	MF2BRLPB6P - MAPLE EFTPE_2 BD Ring Low Priority B 6 Parameter		
0x0000027C	MF2BRLPB7P - MAPLE EFTPE_2 BD Ring Low Priority B 7 Parameter		

Table 26-204. M<yy>BRLPBxP Instantiation

Offset	Instantiation	Access	Reset Value
0x000002C4	MDEBRLPB0P - MAPLE DEPE BD Ring Low Priority B 0 Parameter	R/W	0x0000_0000
0x000002CC	MDEBRLPB1P - MAPLE DEPE BD Ring Low Priority B 1 Parameter		
0x000002D4	MDEBRLPB2P - MAPLE DEPE BD Ring Low Priority B 2 Parameter		
0x000002DC	MDEBRLPB3P - MAPLE DEPE BD Ring Low Priority B 3 Parameter		
0x000002E4	MDEBRLPB4P - MAPLE DEPE BD Ring Low Priority B 4 Parameter		
0x000002EC	MDEBRLPB5P - MAPLE DEPE BD Ring Low Priority B 5 Parameter		
0x000002F4	MDEBRLPB6P - MAPLE DEPE BD Ring Low Priority B 6 Parameter		
0x000002FC	MDEBRLPB7P - MAPLE DEPE BD Ring Low Priority B 7 Parameter		
0x00000344	MCRCBRLPB0P - MAPLE CRCPE BD Ring Low Priority B 0 Parameter	R/W	0x0000_0000
0x0000034C	MCRCBRLPB1P - MAPLE CRCPE BD Ring Low Priority B 1 Parameter		
0x00000354	MCRCBRLPB2P - MAPLE CRCPE BD Ring Low Priority B 2 Parameter		
0x0000035C	MCRCBRLPB3P - MAPLE CRCPE BD Ring Low Priority B 3 Parameter		
0x00000364	MCRCBRLPB4P - MAPLE CRCPE BD Ring Low Priority B 4 Parameter		
0x0000036C	MCRCBRLPB5P - MAPLE CRCPE BD Ring Low Priority B 5 Parameter		
0x00000374	MCRCBRLPB6P - MAPLE CRCPE BD Ring Low Priority B 6 Parameter		
0x0000037C	MCRCBRLPB7P - MAPLE CRCPE BD Ring Low Priority B 7 Parameter		
0x000003C4	MEQBRLPB0P - MAPLE EQPE BD Ring Low Priority B 0 Parameter	R/W	0x0000_0000
0x000003CC	MEQBRLPB1P - MAPLE EQPE BD Ring Low Priority B 1 Parameter		
0x000003D4	MEQBRLPB2P - MAPLE EQPE BD Ring Low Priority B 2 Parameter		
0x000003DC	MEQBRLPB3P - MAPLE EQPE BD Ring Low Priority B 3 Parameter		
0x000003E4	MEQBRLPB4P - MAPLE EQPE BD Ring Low Priority B 4 Parameter		
0x000003EC	MEQBRLPB5P - MAPLE EQPE BD Ring Low Priority B 5 Parameter		
0x000003F4	MEQBRLPB6P - MAPLE EQPE BD Ring Low Priority B 6 Parameter		
0x000003FC	MEQBRLPB7P - MAPLE EQPE BD Ring Low Priority B 7 Parameter		
0x00000444	MCGBRLPB0P - MAPLE CGPE BD Ring Low Priority B 0 Parameter	R/W	0x0000_0000
0x0000044C	MCGBRLPB1P - MAPLE CGPE BD Ring Low Priority B 1 Parameter		
0x00000454	MCGBRLPB2P - MAPLE CGPE BD Ring Low Priority B 2 Parameter		
0x0000045C	MCGBRLPB3P - MAPLE CGPE BD Ring Low Priority B 3 Parameter		
0x00000464	MCGBRLPB4P - MAPLE CGPE BD Ring Low Priority B 4 Parameter		
0x0000046C	MCGBRLPB5P - MAPLE CGPE BD Ring Low Priority B 5 Parameter		
0x00000474	MCGBRLPB6P - MAPLE CGPE BD Ring Low Priority B 6 Parameter		
0x0000047C	MCGBRLPB7P - MAPLE CGPE BD Ring Low Priority B 7 Parameter		
0x000004C4	MCONVBRLPB0P - MAPLE CONVPE BD Ring Low Priority B 0 Parameter	R/W	0x0000_0000
0x000004CC	MCONVBRLPB1P - MAPLE CONVPE BD Ring Low Priority B 1 Parameter		
0x000004D4	MCONVBRLPB2P - MAPLE CONVPE BD Ring Low Priority B 2 Parameter		
0x000004DC	MCONVBRLPB3P - MAPLE CONVPE BD Ring Low Priority B 3 Parameter		
0x000004E4	MCONVBRLPB4P - MAPLE CONVPE BD Ring Low Priority B 4 Parameter		
0x000004EC	MCONVBRLPB5P - MAPLE CONVPE BD Ring Low Priority B 5 Parameter		
0x000004F4	MCONVBRLPB6P - MAPLE CONVPE BD Ring Low Priority B 6 Parameter		
0x000004FC	MCONVBRLPB7P - MAPLE CONVPE BD Ring Low Priority B 7 Parameter		

Figure 26-247 describe the M<yy>BRLPbxP structure:

Offset see **Table 26-204** Access: User read/write

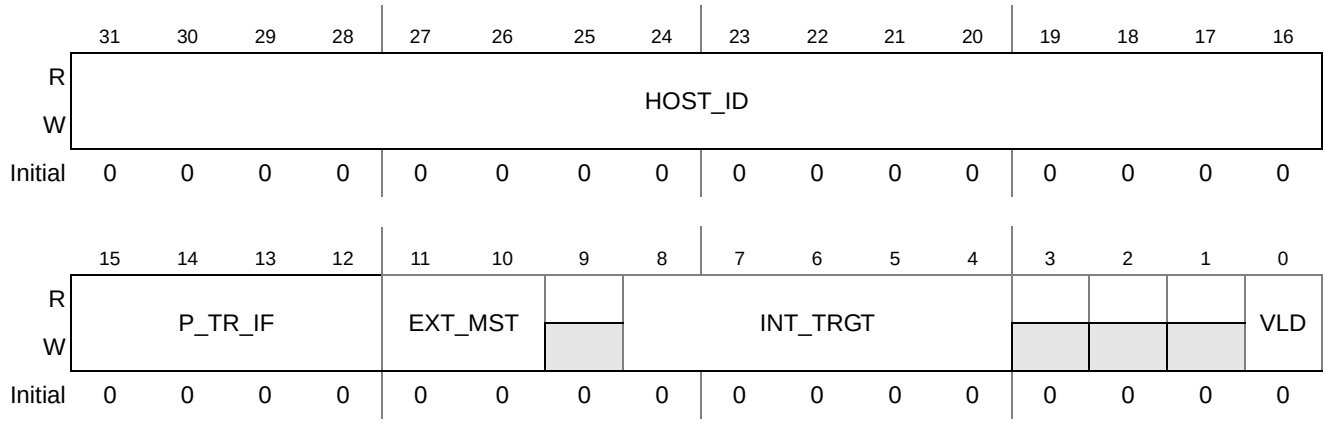


Figure 26-247. MAPLE <yy>PE BD Ring Low Priority B <x> Parameter

Table 26-205. MAPLE <yy>PE BD Ring Low Priority B <x> Parameter Fields Description

Bits	Description
31–16	HOST_ID–16 bits of Host ID for Serial RapidIO door-bell support for an external (to DSP device) master connected to the DSP device by Serial RapidIO port and is the owner of this ring. This field is valid only if M<yy>BRLPBxP[EXT_MST] equals ‘01’ or ‘10’.
15–12	P_TR_IF—Port Target Interface. Determines which RapidIO port to use for this doorbell. This field is used by the MAPLE-B2 for the Word4[TINT] field in the Type10 Outbound Doorbell descriptor.
11–10	EXT_MST—External Master. Describes if the Ring owner can receive any of the MAPLE-B2’s regular interrupt lines, and a regular interrupt (one out of 32) is generated if the [INT_EN] bit in the BD is set; or if the ring owner is connected to the DSP device via Serial RapidIO port and a Serial RapidIO door-bell interrupt is generated if the [INT_EN] bit in the BD is set. For details see **Section 26.4.4, *External Masters Support Using Serial RapidIO Doorbell*.”** ’00’ The master of this ring can receive any of the MAPLE-Bs regular interrupts, therefore a regular interrupt is generated according to MTVBRLPBxP[INT_TRGT]. ’01’ The master of this ring is external (connected via Serial RapidIO port), therefore a Serial RapidIO door-bell interrupt is generated. The door-bell should be initiated via Serial RapidIO port 1 (**Section 26.4.4, *External Masters Support Using Serial RapidIO Doorbell***”). ’10’ The master of this ring is external (connected via Serial RapidIO port), therefore a Serial RapidIO door-bell interrupt is generated. The door-bell should be initiated via Serial RapidIO port 2 (**Section 26.4.4, *External Masters Support Using Serial RapidIO Doorbell***”). ’11’ Reserved
9	Reserved.
8–4	INT_TRGT—Defines which regular interrupt is to be asserted due to task completion in this BD ring. This field is valid only if M<yy>BRLPBxP[EXT_MST] equals ’00’. 00000 irq0 00101 irq1 ... 11111 irq31
3—1	Reserved.
0	VLD—Valid Bit. Describes if the current BD ring is valid. The MAPLE-B2 uses this bit to determine the BD Ring’s validation, therefore, any change in any of the BD Ring’s parameters must be done while the VLD bit is cleared. 0 BD ring is not valid. 1 BD ring is valid.

26.5.4.2 eTVPE Buffer-Descriptor Structure

The following is a description of the eTVPE Buffer Descriptors structure as they should be written to the eTVPE BD rings by the host. Some of the fields may not be relevant for some configurations and are ignored by MAPLE-B2.

Offset BD_BASE + 0x0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OWNER	WRAP		INT_EN	L_MAP	STD	VIT_K			TBZE	ZTTB	CRC_EN	ENC_RATE		HAOE	ALG
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PUNC_SCHM				MAX_ITER				MIN_ITER					IN_D_STRCT		
W																
Offset BD_BASE + 0x4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		BS														
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LLMAP_LCF								CRC_POLY		NDRE				MB_PR	
W																
Offset BD_BASE + 0x8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HOE	SOE		DPNC	TASK_ID								VIT_SET		BUF_SIZE	
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BUF_SIZE															
W																
Offset BD_BASE + 0xC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HRD_RSLT_ADDR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HRD_RSLT_ADDR															
W																
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																

Figure 26-248. eTVPE Buffer Descriptor Structure

Offset	BD_BASE + 0x14															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VEC_OFFSET															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	W_16M					SFT_RSLT_OFF				HOFF			LLR_OUT_SF			
W																
Offset	BD_BASE + 0x18															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		IHBSA														
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FTH	IHBSZ														
W																
Offset	BD_BASE + 0x1C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			HUS			W1E	W2E	W3E	W1							
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	W2								W3							
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																
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															
W																
Offset	BD_BASE + 0x28															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HA_IN_BASE_ADDR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HA_IN_BASE_ADDR															
W																

Figure 26-248. eTVPE Buffer Descriptor Structure (Continued)

Offset	BD_BASE + 0x2C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PF_BUF_SIZE															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PS_BUF_SIZE															
W																
Offset	BD_BASE + 0x30															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HA_OUT_BASE_ADDR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HA_OUT_BASE_ADDR															
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																
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		HSC														
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		CMP_RSN			ERR_ST				ZT_MAX				ITER_CNT			
W																

Figure 26-248. eTVPE Buffer Descriptor Structure (Continued)

Table 26-206. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0

Name	Description
OWNER	**Owner bit**. When set by the host, MAPLE-B2 is the BD owner and the job is not complete. When cleared by MAPLE-B2, the job is done and the host is the BD owner. 0—The host is BD owner 1—MAPLE-B2 is BD owner
WRAP	**Wrap bit**. When set by the host, MAPLE-B2 updates the [BDR_RD_PTR[14:5]] field with [BDR_BA[14:8]], that is, it expects to find the next BD at the Base Address of the ring. When cleared by the host, MAPLE-B2 expects to find the next BD at: [BDR_RD_PTR[14:5]] + BD size. For details, see **Section 26.5.4.1.4** or **Section 26.5.4.1.6**

Table 26-206. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0

Name	Description
INT_EN	**Interrupt Enable**. If set, MAPLE-B2 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	**LogMAP Algorithm**. Valid only for Turbo decoding (ALG=0). Describes the LogMAP algorithm. 0—MaxLogMAP 1—Hybrid Linear LogMAP (Max*).
STD	**Standard**. Determines which standard technology this job relates to. 0—UMTS or WiMAX technology for 3G or WiMAX operation modes respectively. 1—3GLTE standard.
VIT_K[2:0]	**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	**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	**Zero Tail or Tail Biting trellis termination**. 0—Tail Biting trellis termination assumed. 1—Zero Tail trellis termination assumed. **Notes:** **1.** For Turbo decoding (ALG=0) it should be configured as Zero Tail for 3G technologies (3GLTE or UMTS) and as Tail Biting for WiMAX technology. **2.** For Viterbi operation Tail Biting configuration implies WAVA* algorithm as described in **Section 26.4.3.2.6.7**, *Tail Biting Viterbi Processing (WAVA*)*.
CRC_EN	**CRC check enable**. Valid only for Turbo decoding. If set the eTVPE perform CRC check on the Hard Output results generated in the last iteration. The polynomial type is to be determined by the [CRC_POLY] field of the eTVPE BD. The CRC results are described in the [ERROR_STATUS] field of the BD. If the MTVCP[SCRC] bit or MTVCP[CRC_AUTOSTOP] bit is set, that is, CRC stopping criteria is enabled, CRC check is executed regardless of this bit configuration. (see **Section 26.4.3.2.6.10**, *CRC Check*, on page 26-69). 0—CRC check on the Hard Output is disabled. 1—CRC check on the Hard Output is enabled.
ENC_RATE[1:0]	**Encoder Rate**. Describes the encoding rate used for this job. 00—1/2 (Viterbi decoding only) 01—1/3 10—1/4 (Viterbi decoding only) 11—Reserved.
HAOE	**HARQ Accumulator Output Enable**. Valid only if the HARQ Enable bit ([HE]) of the MTVCP parameter is set. Determines whether the MAPLE-B2 is to output the HARQ accumulator after the HARQ stage in the eTVPE. If enabled the MAPLE-B2 output the HARQ result to the address pointed by the HA_OUT_BASE_ADDR field. 0—No HARQ accumulator buffer is being output. 1—HARQ accumulator buffer is being output to address pointed by HA_OUT_BASE_ADDR field.
ALG	**Decoding Algorithm Type**. 0—Turbo algorithm 1—Viterbi algorithm

Table 26-206. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0

Name	Description
PUNC_SCHM[3:0]	**Puncturing Scheme**. Valid only for Viterbi processing ([ALG]=1). Selects one of the 10 possible depuncturing schemes as described in MTVPVxHCP and MTVPVxLCP. See also **Section 26.4.3.2.6.5.1**, *Viterbi Periodic Depuncturing*. 0000—Reserved. 0001—execute Periodic depuncture scheme #0 0010—execute Periodic depuncture scheme #1 ... 1010—execute Periodic depuncture scheme #9 1011 to 1111—Reserved.
MAX_ITER[3:0]	**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. If stop criteria is not enabled, the MAX_ITER field represent the total number of iterations to be executed by the eTVPE. The MAX_ITER value is increment internally by one to represent the values 2 to 16. This value should always be larger than or equals to MIN_ITER. 1 to 15—maximum number of iterations 2 to 16, respectively.
MIN_ITER[3:0]	**Minimum number of iterations**. Valid only during Turbo decoding and only if one of the stop criteria is enabled. Describes the Minimum number of iterations executed by the eTVPE when stop criteria is enabled. The eTVPE starts with its stop criteria checks only after it has reached the number of iterations described in MIN_ITER field. This value should always be smaller or equal to MAX_ITER. 1 to 15—minimum number of iterations 2 to 16, respectively.
IN_D_STRCT[2:0]	**Input Data Structure**. Describes the possible input data structures in system memory to be fetched by MAPLE-B2 into the eTVPE. 000—LTE HARQ. See **Section 26.4.3.2.5.2**, *LTE HARQ Input Data Structure*. 001—WiMAX HARQ. See **Section 26.4.3.2.5.3**, *WiMAX HARQ Input Data Structure*. 010—E-DCH HARQ with Mixed Vector. See **Section 26.4.3.2.5.4**, *E-DCH HARQ with Mixed Vector Input Data Structure*. 011—E-DCH HARQ with Separate Vectors. See **Section 26.4.3.2.5.5**, *E-DCH HARQ with Separate Vectors Input Data Structure* 100—Sub-Block Interleaved. See **Section 26.4.3.2.5.6**, *Sub-Block Interleaved Input Data Structure* 101—UMTS Mixed. See **Section 26.4.3.2.5.7**, *UMTS Mixed Input Data Structure*. 110—Separate Vectors. See **Section 26.4.3.2.5.8**, *Separate Vectors Input Data Structure*. 111—Reserved.

Table 26-207. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4

Field	Description
BS[14:0]	**Block Size**. Describes the current tasks decoded block size in bits. For Turbo decoding it does not include Tail bits (if exist). Following are the valid values for this files: For 3GLTE: 0x28 to 0x1800 (40 to 6144 bits) according to the allowed K values block sizes as described in Table 5.1.3-3 (3GPP TS 36.212 V9.3.0) For WiMAX: 0x30 to 0x12C0 (48 to 4800 bits) For UMTS: 0x28 to 0x13FA (40 to 5114 bits) For Viterbi: 0x14 to 0x1800 (20 to 6144 bits)
LLMAP_LCF[7:0]	**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 details, see **Section 26.4.3.2.6.4.2**, *Turbo Processing Implementation*." 128 possible even decimal values.
CRC_POLY[1:0]	**CRC Polynomial**. Describes which Polynomial to use if CRC check or CRC stop criteria is required. For details, refer to **Section 26.4.3.2.6.10, *CRC 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—Reserved.
NDRE[1:0]	**Number of DRE engines**. Valid only during Turbo decoding and only if MTVCP[ANDRE] is cleared. Determines the number of internal Turbo engines which participate in the decoding process. For details, see **Section 26.4.3.2.6.4.3**, *Number of Turbo Processing Elements (DREs)*. 00—Single DRE. 01—Two DRE engines. 10—Four DRE engines 11—Reserved.
MB_PR[1:0]	**MBus Priority**. Valid only if the [MB_PR_SCH] of the MMC0P parameter (see **Section 26.5.2.1**, *MAPLE Mode Configuration 0 Parameter (MMC0P)*) is set to '01' or '10'. Indicated the MBus accesses priority related to this BD. For details, see **Section 26.4.2.4**, *MBus Priority Scheme Configuration* 00— The MBus accesses related to that BD are initiated with priority '00' ... 11—The MBus accesses related to that BD are initiated with priority '11'

Table 26-208. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x8

Name	Description
HOE	**Hard Output Enable**. Controls the availability of hard-output results from the eTVPE decoder. If set the MAPLE-B2 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]	**Soft Output Enable**. Controls the availability, and type of the soft-output results from the decoder, applicable for Turbo decoding only. For details, see **Section 26.4.3.2.7**, *eTVPE Output Data Structure* 00—No Soft Output 01—Aposteriori Output Enabled (16 bit systematic data only) 10—Extrinsic Output Enabled (16 bits systematic data only). 11—Turbo Soft Outputs (8 bits systematic and parities).
DPNC	**Depuncturing**. Valid only during Turbo decoding (ALG=0), UMTS standard job and when input data structure is 'E-DCH HARQ with Mixed Vector' or 'E-DCH HARQ with Separate Vectors' ([IN_D_STRCT]=2 or 3). Indicates the eTVPE which Rate De-Matching process to execute. 0— The eTVPE performs De-repetition process. 1— The eTVPE performs Depuncturing process.
TASK_ID[7:0]	**Task ID**—Task ID tag, supplied by host, assists in identification of its task when directly accessing the eTVPE to inquire task status. When MAPLE-B2 assigns the job to the eTVPE, it copies the TASK_ID into the eTVPE, thus allowing the host to track the eTVPE jobs.
VIT_SET[1:0]	**Viterbi Set**. Valid only for Viterbi decoding ([ALG]=1). Describes which set of Viterbi polynomial is used for the current job: 00—Reserved. 01—MAPLE-B2 use MTVPVS1CxP parameters to configure the internal eTVPE polynomials. 10—MAPLE-B2 use MTVPVS2CxP parameters to configure the internal eTVPE polynomials. 11—MAPLE-B2 use MTVPVS3CxP parameters to configure the internal eTVPE polynomials. For details, see **Section 26.4.3.2.6.5.1**, *Viterbi Periodic Depuncturing*
BUF_SIZE[17:0]	**Buffer Size**. Describes the input data buffer size (in bytes) to be fetched by MAPLE-B2 during the input stage. For details, see **Section 26.4.3.2.5**, *eTVPE Input Data Structures* 40B to 100KB - possible input block sizes. **Warm-up Size**. When working with Viterbi large block sizes, the BUF_SIZE field indicates the warm-up size required by MAPLE-B2. For details see **Section 26.4.3.2.6.8, *Viterbi Large Blocks Partitioning Support*** Warm-up values must be a multiplication of 8

Table 26-209. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0xC

Name	Description
HRD_RSLT_ADDR[31:0]	**Hard Result Address**. Describes the base address in system memory in which the MAPLE-B2 provides Hard Output results, if enabled (HOE=1).

Table 26-210. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x10

Name	Description
BASE_ADDR[31:0]	**Base Address**. — Points to the start address of the input buffer in case single input stream is required. — Points to the start address in the system memory of the first out of several input vectors. The start address of the other vectors is distances form this pointer by [VEC_OFFSET] multiplications. For details, see **Section 26.4.3.2.5.8**, *Separate Vectors Input Data Structure*.

Table 26-211. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x14

Name	Description
VEC_OFFSET[15:0]	**Vectors Offset**. Valid only for Separate Vectors and E-DCH HARQ with Separate Vectors input data structures. Describes the relative offset between the vectors in the system memory. For details, see **Section 26.4.3.2.5.8**, *Separate Vectors Input Data Structure*.
W_16M	**WiMAX 802.16m indication**. Valid only for WiMAX technology jobs and for Separate Vectors input data structure. Indicates that the required technology is 802.16m. 0— WiMAX 802.16e is assumed. 1— WiMAX 802.16m is assumed.
SFT_RSLT_OFF[2:0]	**Soft Results Offset**. Valid only if 3 vectors of soft outputs are required ([SOE] = 3). Describe the offset of second/third output vector with respect t the first/second output vector in system memory. For details, see **Section 26.4.3.2.8**, *Output Data Types* 000—Second/third vector start address is 256 bytes after first/second vector start address. 001—Second/third vector start address is 512 bytes after first/second vector start address. 010—Second/third vector start address is 1K bytes after first/second vector start address. 011—Second/third vector start address is 2K bytes after first/second vector start address. 100—Second/third vector start address is 4K bytes after first/second vector start address. 101—Second/third vector start address is 8K bytes after first/second vector start address. 110 to 111—Reserved.
HOFF[2:0]	**Hard Output Offset**. Valid only for UMTS standard or Viterbi processing. Describe the bit offset of the first byte in which the first valid bit is to be written. If greater than zero, the MAPLE-B2 writes the first byte in the system memory with bit resolution. For details, see **Section 26.4.3.2.8**, *Output Data Types*. 000—The first output bit is to be written to first bit of first byte. 001—The first output bit is to be written to second bit of first byte. 010—The first output bit is to be written to third bit of first byte. ... 111—The first output bit is to be written to eighth bit of first byte.
LLR_OUT_SF[3:0]	**LLR Output Shift**. Determines which 8 bits of the internal 16 bit wide soft outputs are going to be output. For details, see **Section 26.4.3.2.8**, *Output Data Types*. 0—soft output bits 7:0 are selected. 1—soft output bits 8:1 are selected. 2—soft output bits 9:2 are selected. 3—soft output bits 10:3 are selected. 4—soft output bits 11:4 are selected. 5—soft output bits 12:5 are selected. 6—soft output bits 13:6 are selected. 7—soft output bits 14:7 are selected. 8—soft output bits 15:8 are selected.

Table 26-212. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x18

Name	Description
IHBSA[14:0]	**Input HARQ Buffer Start Address**. Describe the address location of the first bit of the input stream in the HARQ accumulator buffer. For details, see **Section 26.4.3.2.6.2**, *HARQ Combining*.
FTH	**First Time HARQ**. Indicates whether this HARQ buffer is the first one hence no HARQ accumulator exist nor should be supplied. 0—This job is not the first HARQ buffer, hence HARQ accumulator should be supplied. 1—This job is the first HARQ buffer, hence accumulator should be internally reset.
IHBSZ[14:0]	**Input HARQ Buffer Size**. Describe the size of the current HARQ input stream into the accumulator. For details, see **Section 26.4.3.2.6.2**, *HARQ Combining*.

Table 26-213. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x1C)

Name	Description
HUS[1:0]	**HARQ Up Scale**. Describe the required up scaling of the HARQ input stream before being added into the HARQ accumulator buffer. 0—The Input data is not pre scaled before being added to the accumulator 1—The Input data is up scaled by 2 before being added to the accumulator 2—The Input data is up scaled by 4 before being added to the accumulator 3—The Input data is up scaled by 8 before being added to the accumulator
W1E	**Weight 1 Enable**. Describe whether accumulator data that is covered by input data is multiplied by W1 before being added to input data and updated as new accumulator data. For details, see **Section 26.4.3.2.6.2**, *HARQ Combining*. 0—Accumulator data covered by input data is **not** multiplied by W1 and is being added as is to input data result. 1—Accumulator data covered by input data is multiplied by W1 before being added to input data result
W2E	**Weight 2 Enable**. Describe whether input data is multiplied by W2 before being added to accumulator data result and updated as new accumulator data. For details, see **Section 26.4.3.2.6.2**, *HARQ Combining*. 0—Input data is **not** multiplied by W2 and is being added as is to accumulator data result. 1—Input data is multiplied by W2 before being added to accumulator data result
W3E	**Weight 3 Enable**. Describe whether accumulator data which is not covered by input data is multiplied by W3 before being updated as new accumulator data. For details, see **Section 26.4.3.2.6.2**, *HARQ Combining*. 0—Accumulator data that is not covered by input data is **not** multiplied by W3 before being updated as new accumulator data. 1—Accumulator data that is not covered by input data is multiplied by W3 before being updated as new accumulator data.
W1[7:0]	**Weight 1**. Valid only if W1E is set. Un-singed number between 0 and 255. All Accumulator data that is covered by new data is multiplied by W1/256 before being added to the new accumulator. For details, see **Section 26.4.3.2.6.2**, *HARQ Combining*.
W2[7:0]	**Weight 2**. Valid only if W2E is set. Un-singed number between 0 and 255. All input data is multiplied by W2/256 before being added to the new accumulator.
W3[7:0]	**Weight 3**. Valid only if W3E is set. Un-singed number between 0 and 255. All Accumulator data that is not covered by new data is multiplied by W3/256 before being added to the new accumulator.

Table 26-214. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x20

Name	Description
SFT_RSLT_ADDR[31:3]	**Soft Result Address**. The address where the MAPLE-B2 provides the Soft Output results if enabled ([SOE]= 1, 2, 3). The [SFT_RSLT_ADDR] field must be aligned to 8 bytes.

Table 26-215. eTVPE Buffer Descriptor Names Description of 4 Bytes at Offset 0x24

Name	Description
E_PARAM_PTR[31:3]	**E parameters pointer**. Valid only during Turbo decoding ([ALG]=0), UMTS standard job and when input data structure is 'E-DCH HARQ with Mixed Vector' or 'E-DCH HARQ with Separate Vectors' ([IN_D_STRCT]=2 or 3). A pointer to the E parameters data structure in the system memory used by the eTVPE to perform the Rate De-Matching process. For more detail see **Section 26.4.3.2.6.1**, *E-DCH Rate De-Matching*.

Table 26-216. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x28

Name	Description
HA_IN_BASE_ADDR [31:0]	**HARQ Accumulator Input Base Address**. Points to base address of the HARQ Accumulator buffer to be fetched by MAPLE-B2 into the eTVPE before the new input stream.

Table 26-217. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x2C

Name	Description
PF_BUF_SIZE[15:0]	**Parity First Buffer Size**. Describe the actual buffer size of the PF vector during 'E-DCH HARQ with Separate Vectors' input data structure. For details, see **Section 26.4.3.2.5.5**, *E-DCH HARQ with Separate Vectors Input Data Structure*
PS_BUF_SIZE[15:0]	**Parity Second Buffer Size**. Describe the actual buffer size of the PS vector during 'E-DCH HARQ for Separate Vectors' input data structure.

Table 26-218. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x30

Name	Description
HA_OUT_BASE_ADDR [31:0]	**HARQ Accumulator Output Base Address**. Points to base address of the HARQ Accumulator buffer in the system memory where the MALE-B2 is to output the new HARQ buffer after eTVPE combining.

Table 26-219. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x34

Name	Description
[31:0]	Reserved.

Table 26-220. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x38

Name	Description
[31:0]	Reserved.

Table 26-221. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x3C

Name	Description
HSC[14:0]	**HARQ Saturation Counter**. Valid after [OWNER] bit is cleared by MAPLE-B2, and if the HARQ operation is enabled ([HE] bit of the MTVCP parameter). Indicates the number of accumulator bytes that were overflown during the HARQ operation, thus saturated.
CMP_RSN[2:0]	**Complete Reason**. Valid after [OWNER] bit is cleared by MAPLE-B2. Indicates the reason for eTVPE job completion. 000— Complete due to MAX_ITER value reached 001— Complete due to Aposteriori quality Stop Criteria 010— Complete due to CRC check Stop Criteria 011— Complete due to both Aposteriori quality check Stop Criteria and CRC check Stop Criteria 100— Complete due to CRC Steady Stop Criteria. 101— Complete due to both Aposteriori quality check Stop Criteria and CRC Steady Stop Criteria 110 to 111—Reserved.

Table 26-221. eTVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x3C

Name	Description
ERR_ST[3:0]	**Task Error Status Code**. Valid only after [OWNER] bit is cleared by MAPLE-B2. 0000—No Error 0001—CRC check on Hard Output results failed. 0010—eTVPE stopped due to CRC check or CRC steady stop criteria, but failed on CRC check or CRC steady on next iteration executed by the eTVPE. For details, see **Section 26.4.3.2.6.4.9**, *Stopping Criteria Status Reports*. 0011—Reserved. 0100—eTVPE stopped due to Aposteriori Quality stop criteria, but failed on Aposteriori Quality on next iteration executed by the eTVPE.For details, see **Section 26.4.3.2.6.4.9**, *Stopping Criteria Status Reports* 0101 to 1111—Reserved.
ZT_MAX	**Zero Tail Max**. Valid only during Viterbi Zero Tail decoding ([ALG]=1, [ZTTB]=1) while Trace Back from zero is enabled ([TBZE]=1) and after [OWNER] bit is cleared by MAPLE-B2. This bit is set by MAPLE-B2 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.
ITER_CNT[3:0]	**Iteration Count**. Valid only during Turbo decoding and after [OWNER] bit is cleared by MAPLE-B2. Status field which describes the actual number of full iterations executed for this job. The actual iteration number is ITER_CNT +1.

The following restrictions apply:

- 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.
- All reserved bits in the BD must be set to zero for future compatibility.
- Once the [OWNER] bit is set by the host, the BD content must not be changed. It is re-written only after the [OWNER] bit was clear by MAPLE-B2.
- The status fields in the BD are valid only after the [OWNER] bit is clear by the MAPLE-B2.

26.5.4.3 eFTPE Buffer Descriptor Structure

A description of the eFTPE Buffer Descriptors structure expected by the host for the eFTPE BD rings is outlined in **Figure 26-249**. Some of the fields are status fields which are written by the MAPLE-B2 upon job completion. Other fields may be relevant only for certain configurations and is ignored by the MAPLE-B2 if another configuration is used. The length of the basic BD is 48 bytes, and it includes a potential extension to 96 bytes (BD control bit dependent). It means that the next BD location should be located either 48 bytes or 96 bytes after the current BD address.

Offset	BD_BASE + 0x0															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OWNER	WRAP		INT_EN	TASK_ID								FC_CG		DSTZP	
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DSTZP										TL_ID					
W																
Offset	BD_BASE + 0x4															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	AIUS	PRMUR	PSMUR	PSTMV		ZP	SINS	CPRE	PO_SCL				GRDC	AFS	EXS	
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PSTME	CPIE	GR	PME		OVA_SCL	SCL_TYPE	ITE	GI					DSS		
W																
Offset	BD_BASE + 0x8															
	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														MB_PR	
W																
Offset	BD_BASE + 0xC															
	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														OFF16	OFFKB
W																

Figure 26-249. eFTPE Buffer Descriptor Structure

Offset	BD_BASE + 0x10															
	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			USR_SCL4		
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	USR_SCL5			IN_SCL					ADP_OVA_SCL							
W																
Offset	BD_BASE + 0x14															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRM_PTR[31:3]															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PRM_PTR[31:3]															
W																
Offset	BD_BASE + 0x18															
	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																
Offset	BD_BASE + 0x1C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PSTM_REAL[15:0]															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PSTM_IMG[15:0]															
W																

Figure 26-249. eFTPE Buffer Descriptor Structure (Continued)

Offset	BD_BASE + 0x20															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CPS									BD_RPT						
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADP_OVA_SCL_ST									CMP_RSN						
W																
Offset	BD_BASE + 0x24															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INP_EXP_PTR[31:3]															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INP_EXP_PTR[31:3]															
W																
Offset	BD_BASE + 0x28															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	POM_PTR[31:3]															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	POM_PTR[31:3]															
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																

Figure 26-249. eFTPE Buffer Descriptor Structure (Continued)

Table 26-222. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0

Name	Description
OWNER	**Owner bit**. When set by the host, MAPLE-B2 is the BD owner and the job is not complete. When cleared by MAPLE-B2, the job is done and the host is the BD owner. 0—The host is BD owner 1—MAPLE-B2 is BD owner
WRAP	**Wrap bit**. When set by the host, MAPLE-B2 updates the [BDR_RD_PTR[14:5]] field with [BDR_BA[14:8]], that is, it expects to find the next BD at the Base Address of the ring. When cleared by the host, MAPLE-B2 expects to find the next BD at: [BDR_RD_PTR[14:5]] + BD size. For details, see **Section 26.5.4.1.4**, *MAPLE <yy>PE BD Ring High Priority A <x> Parameter (M<yy>BRHPAxP)* or **Section 26.5.4.1.6**, *MAPLE <yy>PE BD Ring Low Priority A <x> Parameter (M<yy>BRLPAxP)*
INT_EN	**Interrupt Enable**. If set, MAPLE-B2 issues an 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 26-222. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0

Name	Description
TASK_ID[7:0]	**Task ID**. Task ID tag, supplied by host, assists it in identification of its task when directly accessing the eFTPE to inquire task status.
FC_CG	**Frequency Correction or Code Generation extension**. Indicates if either Frequency Correction feature or Code Generation feature are enabled hence BD extension is applicable. For details see **Section 26.5.4.3.2**, *eFTPE Buffer Descriptor's Extension*. 0—BD extension is not applicable. BD size is 48 bytes. 1—BD extension is applicable. BD size is 96 bytes.
DSTZP[11:0]	**Data Size to Zero Padding**. Indicates the actual amount of samples required to be padded by zeros. The eFTPE pads the given value with zeros till the transform size indicated by the *TL_ID* is reached. *DSTZP* is limited by the transform length (*DSTZP* < transform length) and its value should be a product of 4 if short input is disabled (*SINS* = 0) or product of 8 is short input is enabled (*SINS* = 1). Valid Range: 0 to 2044.
TL_ID[5:0]	**Transform Length ID**. This field determines the executed transform length. See **Table 26-246** for DFT/FFT Transform Length encoding.

Table 26-223. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4

Name	Description
AIUS	**Adaptive Input Scaling**. Valid only in Adaptive Scale mode (*SCL_TYPE*=0). Indicates if the eFTPE is to perform adaptive up scale of the input data to maximize calculation precision. For details see **Section 26.4.3.3.9.4.7**, *Adaptive Input Scaling*. 0—The input samples are processed as is. 1—The input samples are adaptively scaled up.
PRMUR	**Pre Multiplier Repeat**. Valid only if BD repeat (*BD_RPT* > 0) and vector pre multiplication are enabled (*PME*=1,3). Indicates the eFTPE if all the repeated iterations use the same Pre-Multiplier buffer, or if each iteration has a different buffer. For details see **Section 26.4.3.3.5.1**, *Pre-Multiplication Processing Support in the eFTPE*. 0—Use different buffer for each iteration. 1—Use the same repeated pre multiplier buffer for all iterations. **Note**: *PRMUR* should not be cleared if *PME*=1.
PSMUR	**Post Multiplier Repeat**. Valid only if BD repeat (*BD_RPT* > 0) and vector post multiplication are enabled (*PSTMV*=3). Indicates the eFTPE if all the repeated iterations use the same Post-Multiplier buffer, or if each iteration has a different buffer. For details see **Section 26.4.3.3.5.2**, *Post-Multiplication processing support in the eFTPE*. 0—Use different buffer for each iteration. 1—Use the same repeated pre multiplier buffer for all iterations. **Note**: *PSMUR* should not be cleared if *PSTMV*=1.
PSTMV	**Post multiplier Vector enable**. If enabled, the eFTPE multiplies each output sample with a complex value according the post-multiply memory content. For details see **Section 26.4.3.3.5.2**, *Post-Multiplication processing support in the eFTPE*. 00—Post-Multiplier is disabled. 01—Post-Multiplier is enabled. The eFTPE uses the post-multiplier memory without updating it 10—Reserved. 11—Post-Multiplier is enabled. The MAPLE-B2 update the post-multiplier memory with new data from system memory pointed by the *POM_PTR* field of the eFTPE BD.

Table 26-223. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4

Name	Description
ZP	**Zero Padding**. Indicates the eFTPE to add zero padding to the input data according to the *DSTZP* and *TL_ID* fields. For details see **Section 26.4.3.3.4.1.2**, *Zero Padding Support*. 0—No zero padding is done to input samples. 1—Zero padding is added according to the *DSTZP* and *TL_ID* fields. **Note:** Zero Padding should not be enabled along with zero insertion process enabled by ZI bit.
SINS	**Short Input Structure**. Valid only for FFT/iFFT processing. Indicates the input sample structure of the eFTPE. For details see **Section 26.4.3.3.4.1**, *Input Data Structures*. 0—Input sample structure is 32 bits: 16I and 16Q. 1—Input sample structure is 16 bits: 8I and 8Q.
CPRE	**Cyclic Prefix Removal Enable**. Valid only if BD Repeat option is enabled. If set, the MAPLE-B2 skip the cyclic prefix addition when fetching the next job during BD repeat. For details see **Section 26.4.3.3.4.1.3**, *Cyclic Prefix Removal for FFT BD Repeat*. 0—Cyclic Prefix Removal is disabled. 1—Cyclic Prefix Removal is enabled.
PO_SCL[1:0]	**Post Multiplier Scaling**. Valid during User Defined Scale mode only (*SCL_TYPE*=1) and if the Vector post multiplication is enabled (*PSTMV*=1). Indicates the scaling amount that should be applied on the vector post multiplication results. Valid Range: 0 to 2.
GRDC	**Guard Removal of DC**. Valid only if the *GR* bit of the eFTPE BD is set. Indicates the MAPLE-B2 whether a ‘DC’ sample is expected after FFT processing. Foe details see **Section 26.4.3.3.4.2.1**, *Guard Band Removal for FFT*. 0— The MAPLE-B2 outputs the ‘DC’ sample as part of the data. 1— The MAPLE-B2 does not output the ‘DC’ sample as part of the data.
AFS	**Align to First Scale**. Valid only if repeat mode and Adaptive Scaling are enabled (*BD_RPT* > 0, *SCL_TYPE* = 0). Allows applying identical scale value to all jobs of the BD repeat according to the scale result of the first job. For details see **Section 26.4.3.3.3**, *Identical Output Scale Alignment for BD Repeat*. 0—No special alignment is performed to all tasks in the repeat job. 1—All tasks in the repeat BD shares the same output scale value (according to the first task). If Extra Scaling is enabled (*EXS* > 0), it is applied for the first task only.
EXS[1:0]	**Extra Scaling**. Valid only if Adaptive Scaling is enabled and no if Overall Scaling is applied. Indicates the eFTPE to perform down scaling (by the value of this field) of the final scale results. Fro details see **Section 26.4.3.3.9.4.8**, *Extra Scaling*. Valid Range: 0 to 2.
PSTME	**Post Multiplication Enable**. If set, the eFTPE multiplies all FFT/DFT output samples with the complex value as described in *PSTM_REAL* and *PSTM_IMG* of the eFTPE BD. For details, refer to **Section 26.4.3.3.5.2, *Post-Multiplication processing support in the eFTPE*** 0—Post Multiplication is disabled. 1—Post Multiplication is enabled.
CPIE	**Cyclic Prefix Insertion Enable**. If set, the MAPLE-B2 generates additional Cyclic Prefix to the FFT output. For details, refer to **Section 26.4.3.3.4.2.2**, *Cyclic Prefix Insertion for iFFT* 0—Cyclic Prefix Insertion is disabled. 1—Cyclic Prefix Insertion is enabled.
GR	**Guard Removal**. Guard Removal output data mode enable. Valid only during FFT operation (*ITE* = 0) and must be cleared for any other operation. For details, see **Section 26.4.3.3.4.2.1**, *Guard Band Removal for FFT* 0—Guard Removal output data mode is disabled 1—Guard Removal output data mode is enabled

Table 26-223. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4

Name	Description
PME[1:0]	**Pre-Multiply Enable**. If enabled, the eFTPE multiplies each input sample with a complex value according the pre-multiply memory content. For details, see **Section 26.4.3.3.5.1**, *Pre-Multiplication Processing Support in the eFTPE* 00—Pre-Multiplier is disabled. 01—Pre-Multiplier is enabled. The eFTPE uses the pre-multiplier memory without updating it 10—Reserved. 11—Pre-Multiplier is enabled. The MAPLE-B2 update the pre-multiplier memory with new data from system memory pointed by the *PRM_PTR* filed of the eFTPE BD.
OVA_SCL	**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 eFTPE aligns the total accumulated scaling with the scaling value specified in the *ADP_OVA_SCL* filed in the BD.
SCL_TYPE	**Scaling Type**. Selects the scaling method to be used between the transform stages as described in **Section 26.4.3.3.9.4, *eFTPE Internal Scaling Calculations***." 0—Adaptive scaling is performed. 1—User defined scaling is performed.
ITE	**Inverse Transform Enable**. If set, an inverse transform is executed. 0—FFT/DFT is executed. 1—iFFT/iDFT is executed.
GI	**Guard Insertion**. Guard Insertion input data mode enable. Valid only during iFFT operation and must be cleared for any other operation. For details, see **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT* 0—Guard Insertion input data mode is disabled 1—Guard Insertion input data mode is enabled **Notes: 1.** Do not enable Guard Insertion with Zero Padding (enabled by *ZP* bit). **2.** Do not enable Guard Insertion with internal code generation (enabled by *CG* bit)
DSS[2:0]	**Data Size Set**. Indicates the MAPLE-B2 which set of Data Size parameters (see **Section 26.5.3.3.1**, *eFTPE Data Size Set x Parameter 0 (FTPEDSSxP0)* to **Section 26.5.3.3.3**, *eFTPE Data Size Set x Parameter 2(FTPEDSSxP2)*) to use for the Guard Insertion or Guard Removal operations. For details, see **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT* and **Section 26.4.3.3.4.2.1**, *Guard Band Removal for FFT* 000—Data Size register in eFTPE are assumed to be updated. Valid only if *GI* is set. 001—The Data size used for Guard Insertion or Guard Removal is taken from Set #1. 010—The Data size used for Guard Insertion or Guard Removal is taken from Set #2. ... 111—The Data size used for Guard Insertion or Guard Removal is taken from Set #7.

Table 26-224. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x8

Name	Description
IBA[31:3]	**Input Buffer Address**. This field points to the input buffer location in system memory, where MAPLE-B2 is to fetch the data. The address must be aligned to an 8 byte address.
MB_PR[1:0]	**MBus Priority**. Valid only if the *MB_PR_SCH* of the MMC0P parameter (see **Section 26.5.2.1**, *MAPLE Mode Configuration 0 Parameter (MMC0P)*) is set to '01' or '10'. Indicated the MBus accesses priority related to this BD. For details, see **Section 26.4.2.4, *MBus Priority Scheme Configuration*** 00— The MBus accesses related to that BD are initiated with priority '00' ... 11—The MBus accesses related to that BD are initiated with priority '11'

Table 26-225. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0xC

Name	Description
OBA[31:3]	**Output Buffer Address**. This field points to the output buffer location in system memory, where MAPLE-B2 is to write the result data. The address must be aligned to an 8 byte address.
OFF16	**Offset in 16 Bytes**. Valid only for DFT/iDFT operation and when the BD repeat option is enabled (BD_RPT > 0). Indicates the MAPLE-B2 that the input buffers during the repeat are expected with offset equal the CPS field value multiplied by 16 Bytes. For details see **Section 26.4.3.3.4.1.5**. 0— No offset between input buffers during repeat is expected. 1— The Offset between the input buffers during the repeat is the CPS field value multiplied by 16 Bytes.
OFFKB	**Offset in KBytes**. Valid only for FFT operation and when the BD repeat option is enabled (BD_RPT > 0). Indicates the MAPLE-B2 that the input buffers during the repeat are expected with offset equal the CPS field value in KBytes. For details see **Section 26.4.3.3.4.1.4**. 0— No offset between input buffers during repeat is expected. 1— The Offset between the input buffers during the repeat is described in the CPS field (in KBytes).

Table 26-226. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x10

Name	Description
PM_SCL	**User scaling after pre-multiplication**. 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 the pre-multiplication. 0—no scaling is performed after the pre-multiplication stage. 1—the results of the pre-multiplication stage are scaled down (shift right) by 1 bit.
USR_SCL0[2:0]	**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 bit. 010—the results of stage 0 are scaled down (shift right) by 2 bits. 011—the results of stage 0 are scaled down (shift right) by 3 bits. 100—the results of stage 0 are scaled down (shift right) by 4 bits. 101 to 111— Reserved
USR_SCL1[2:0]	**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 bit. 010—the results of stage 1 are scaled down (shift right) by 2 bits. 011— the results of stage 1 are scaled down (shift right) by 3 bits. 100—the results of stage 1 are scaled down (shift right) by 4 bits. 101 to 111— Reserved
USR_SCL2[2:0]	**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 bit. 010—the results of stage 2 are scaled down (shift right) by 2 bits. 011—the results of stage 2 are scaled down (shift right) by 3 bits. 100—the results of stage 2 are scaled down (shift right) by 4 bits. 101 to 111— Reserved

Table 26-226. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x10

Name	Description
USR_SCL3[2:0]	**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 bit. 010—the results of stage 3 are scaled down (shift right) by 2 bits. 011—the results of stage 3 are scaled down (shift right) by 3 bits. 100—the results of stage 3 are scaled down (shift right) by 4 bits. 101 to 111— Reserved
USR_SCL4[2:0]	**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 bit. 010—the results of stage 4 are scaled down (shift right) by 2 bits. 011—the results of stage 4 are scaled down (shift right) by 3 bits. 100—the results of stage 4 are scaled down (shift right) by 4 bits. 101 to 111— Reserved
USR_SCL5[2:0]	**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 bit. 010—the results of stage 5 are scaled down (shift right) by 2 bits. 011—the results of stage 5 are scaled down (shift right) by 3 bits. 100—the results of stage 5 are scaled down (shift right) by 4 bits. 101 to 111— Reserved
IN_SCL[3:0]	**Input Scaling**. Valid only if *AIUS* is disabled. 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. ... 1111—the input data is scaled up (shift left) by 15 before stage 0.
ADP_OVA_SCL[7:0]	**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 the eFTPE 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. Supported values: -128:127

Table 26-227. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x14

Name	Description
PRM_PTR[31:3]	**Pre-Multiplier Pointer**. Points to the address in the system memory where the new data for pre-multiplier memory is located. If the *PME* field equals ‘11’ the MAPLE-B2 updates the content of the eFTPE pre-multiplier memory with the data pointed by this field.

Table 26-228. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x18

Name	Description
BD_STAT_PTR[31:3]	**BD Status Pointer**. This field points to the location in system memory where the MAPLE-B2 outputs the *CMP_RSN*[2:0] and the *ADP_OVA_SCL_ST*[7:0] 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. Configuring this field to 0x00000000 disables this feature.

Table 26-229. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x1C

Name	Description
PSTM_REAL[15:0]	**Post Multiplication Real**. Valid only if the *PSTME* bit is enabled. Describe the Real part in 1q15 representation of the complex number to be multiplied with the FFT/DFT samples after the processing. Can receive any value with the only limitation that ([PSTM_REAL]^2 + [PSTM_IMG]^2) < 2
PSTM_IMG[15:0]	**Post Multiplication Imaginary**. Valid only if the *PSTME* bit is enabled. Describe the Imaginary part in 1q15 representation of the complex number to be multiplied with the FFT/DFT samples after the processing. Can receive any value with the only limitation that ([PSTM_REAL]^2 + [PSTM_IMG]^2) < 2

Table 26-230. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x20

Name	Description

CPS[8:0]	**Cyclic Prefix Size**. Valid only if either *CPIE*, *CPRE*, *OFF16 or OFFKB* bits are enabled. Indicates the following:
	If (*CPIE*==1 & *ITE*==1): the number of iFFT samples in the CP to be re-read from the eFTPE output buffer and concatenated to the system output buffer as described in **Section 26.4.3.3.4.2.2**, *Cyclic Prefix Insertion for iFFT*.
	If (*CPRE*==1 & *ITE*==0): the number of FFT samples in the CP to be skipped from the input buffer in the system memory during BD repeat operation as described in **Section 26.4.3.3.4.1.3**, *Cyclic Prefix Removal for FFT BD Repeat*.
	If (*OFF16*==1): the expected offset multiplied by 16 Bytes between the DFT/iDFT input buffers during DFT/iDFT BD repeat operation. For details see **Section 26.4.3.3.4.1.5**, *Input Buffers Offset (in 16 Bytes multiplication) for DFT/iDFT BD Repeat*.
	If (*OFFKB*==1 & *ITE*==0): the expected offset in KBytes between the FFT input buffers during FFT BD repeat operation. For details see **Section 26.4.3.3.4.1.4**, *Input Buffers Offset (in KBs) for FFT BD Repeat*.
	
	0 to 511 for 1 to 512 samples/KBytes or 16 bytes multiplications.

BD_RPT[6:0]	**BD Repeat**. This field allows the use of the same BD, for multiple jobs with identical input parameters. For details, see **Section 26.4.3.3.2, *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.
CMP_RSN[2:0]	**Complete Reason**. This field describes the work completion status. 000—Task completed OK. 001— Reserved. 010—Task completed with saturation event.[1] 011 to 111—Reserved.
ADP_OVA_SCL_ST[7:0]	**Adaptive Overall Scaling Status**. Valid only if Adaptive Scaling is enabled (*SCL_TYPE* = 0). It describes the total Adaptive scaling applied during the processing stages[2]. Supported values: -128:127

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 26-231. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x24

Name	Description
INP_EXP_PTR[31:3]	In non-repeat mode (*BD_RPT*=0): **Input Exponent[7:0]**. Should be written to bits [31:24] of this field. Indicates the input exponent value (scale) of the current task. Used by the eFTPE when calculating the overall scaling and when Adaptive Overall Scaling is used. For details see **Section 26.4.3.3.9.4.5**, *Input Exponent* Supported values: -128:127 In Repeat mode (*BD_RPT* > 0): **Input Exponent Pointer**. Points to an address in the system memory where the Input Exponent values of all the tasks in the job are placed. Setting this pointer to zero indicates the MAPLE-B2 to use Input Exponent value of zero to all repeated tasks. For details see **Section 26.4.3.3.9.4.5**, *Input Exponent*. The pointer must be 8 bytes aligned.

Table 26-232. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x28

Name	Description
POM_PTR[31:3]	**Post-Multiplier Pointer**. Points to the address in the system memory where the new data for post-multiplier memory is located. If the *PSTMV* field equals '11' the MAPLE-B2 updates the content of the eFTPE post-multiplier memory with the data pointed by this field.

Table 26-233. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x2C

Name	Description
[31:0]	Reserved.

26.5.4.3.1 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 PSIF2 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.
- All reserved bits in the BD must be set to zero for future compatibility.
- Once the *OWNER* bit is set by the host, the BD content must not be changed. It is re-written only after the *OWNER* bit was clear by MAPLE-B2.
- The status fields in the BD are valid only after the *OWNER* bit is clear by the MAPLE-B2.

26.5.4.3.2 eFTPE Buffer Descriptor's Extension

If either the Frequency Correction feature (see **Section 26.4.3.3.7**, *Frequency Correction Support in eFTPE*) or the internal Code Generation feature (see **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*) are enabled, the eFTPE BD size is increased from 48 bytes to 96 bytes. **Figure 26-250** outlines the 48 bytes extension to the eFTPE BD as described in **Figure 26-249**. The validity of the BD extension is determined by the *FC_CG* bit of the eFTPE BD.

Offset	BD_BASE + 0x30															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FCG	CG														FCPRE
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FCISC															
W																
Offset	BD_BASE + 0x34															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FCB_REAL															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FCB_IMAG															
W																
Offset	BD_BASE + 0x38															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FCS															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FCSS															
W																
Offset	BD_BASE + 0x3C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FCS_REAL															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FCS_IMAG															
W																

Figure 26-250. eFTPE extension Buffer Descriptor Structure

Offset	BD_BASE + 0x40															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FCF_REAL_ST															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FCF_IMAG_ST															
W																
Offset	BD_BASE + 0x44															
	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	SCOS															
W																
Offset	BD_BASE + 0x48															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NNPI															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CO															
W																
Offset	BD_BASE + 0x4C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CNPI			SGO				LSS	SSN							
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SSN															
W																
Offset	BD_BASE + 0x50															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R				CL												
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													SN			
W																

Figure 26-250. eFTPE extension Buffer Descriptor Structure

Offset	BD_BASE + 0x54															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R						NNPIS			NSGO				NSN			
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NCO															
W																
Offset	BD_BASE + 0x58															
	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 + 0x5C															
	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 26-250. eFTPE extension Buffer Descriptor Structure

Table 26-234. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x30

Name	Description
FCG	**Frequency Correction Generation**. Indicates the eFTPE to perform frequency correction calculation as described in **Section 26.4.3.3.7**, *Frequency Correction Support in eFTPE* on the data either before processing (pre frequency correction) or after the processing (post frequency correction). 0—No frequency correction generation occur. 1—Frequency correction generation function is enabled either before or after the processing.
CG	**Code Generation**. If set, indicates the eFTPE to internally generate the scrambled pilot code as described in **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation* and to perform the relevant processing as described in the BD. When Code Generation is enabled no input data is expected by the eFTPE. 0—Code Generation is disabled. 1—Code Generation is enabled.
FCPRE	**Frequency Correction Pre Multiply**. Valid only if *FCG* is enabled. Indicates the eFTPE if to perform the Frequency Correction calculation on the data prior to the DFT/FFT processing or after the processing. For details see **Section 26.4.3.3.7**, *Frequency Correction Support in eFTPE*. 0—Use frequency correction function as vector post multiplication argument. 1—Use frequency correction function as pre multiplication argument. **Note:** Frequency correction cannot be applied in the same phase with other vector multiplication process.
FCISC[15:0]	**Frequency Correction Input Step Counter**. Valid only if *FCG* is enabled. Indicates the step counter initial value for starting the frequency correction argument generation. Its value should be small or equal to the *FCSS* field. If Step Size (*FCSS*) equals 1, this field should be set to 0x0. For details see **Section 26.4.3.3.7**, *Frequency Correction Support in eFTPE*. Supported values: 0 to 65532 which are products of 4.

Table 26-235. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x34

Name	Description
FCB_REAL[15:0]	**Frequency Correction Base Value Real Part**. Valid only if *FCG* is enabled. Describe the real part value of the Frequency Correction base value. Supported values: 1 to 65535
FCB_IMAG[15:0]	**Frequency Correction Base Value Imaginary Part**. Valid only if *FCG* is enabled. Describe the imaginary part value of the Frequency Correction base value. Supported values: 1 to 65535

Table 26-236. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x38

Name	Description
FCS[11:0]	**Frequency Correction Size**. Valid only if *FCG* is enabled. The number of frequency correction arguments that should be generated. When using zero padding or Guard insertion, the size of correction should be equal to the actual data size, i.e *DSTZP* if using zero padding, or 2*half_data_size if using guard insertion. For details see **Section 26.4.3.3.7**, *Frequency Correction Support in eFTPE*. Supported values: 4 to 2048 which are products of 4.
FCSS[15:0]	**Frequency Correction Step Size**. Valid only if *FCG* is enabled. The number of samples that should be corrected with the same correction factor before the correction factor is increment by the shift value. For details see **Section 26.4.3.3.7**, *Frequency Correction Support in eFTPE*. Supported values: 1 or values between 4–65532 which are products of 4.

Table 26-237. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x3C

Name	Description
FCS_REAL[15:0]	**Frequency Correction Shift Value Real Part**. Valid only if *FCG* is enabled. Describe the real part value of the Frequency Correction shift value. Supported values: 1 to 65535
FCS_IMAG[15:0]	**Frequency Correction Shift Value Imaginary Part**. Valid only if *FCG* is enabled. Describe the imaginary part value of the Frequency Correction shift value. Supported values: 1 to 65535

Table 26-238. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x40

Name	Description
FCF_REAL_ST[15:0]	**Frequency Correction Factor Real Part**. Valid only if *FCG* is enabled. Status field which indicates the real part of the next argument to be generated after current frequency correction job end. Can be used as the *FCB_RAEL* input of the next job.
FCF_IMAG_ST[15:0]	**Frequency Correction Factor Imaginary Part**. Valid only if *FCG* is enabled. Status field which indicates the imaginary part of the next argument to be generated after current frequency correction job end. Can be used as the *FCB_IMAG* input of the next job.

Table 26-239. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x44

Name	Description
SCOS[15:0]	**Step Counter State**. Valid only if *FCG* is enabled. Status field which indicates the internal step counter state after the last frequency correction argument was generated for the current job. Can be used as the *FCISC* field of the next successive job. Supported values: 0 to 65532.

Table 26-240. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x48

Name	Description
NNPI[2:0]	**Next Number of Pilots**. Valid only if *CG* is enabled. Indicates the number of pilot bits in the next frame. For details see **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*. Supported values: 0 to 5 representing 3 to 8.
CO[15:0]	**Chip Offset**. Valid only if *CG* is enabled. Describe the offset to be taken as starting point for scrambling code generation. For details see **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*. Supported values: 0 to 38399.

Table 26-241. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4C

Name	Description
CNPI[2:0]	**Current Number of Pilots**. Valid only if *CG* is enabled. Describe the number of pilot bits in the current frame. For details see **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*. Supported values: 0 to 5 representing 3 to 8.
SGO[3:0]	**Slot Generation Off**. Valid only if *CG* is enabled. Each bit represent one slot starting from the current slot (LSB bit) and to the next 3 slots. If set, the bit represents a slot in which the current generated user's pilot does not exist. For details see **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*.
LSS	**Long Scrambling Sequence**. Valid only if *CG* is enabled. Indicates the scrambling sequence generation mode. For details see **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*. 0—The generation machine works in short scrambling sequence mode. 1—The generation machine works in long scrambling sequence mode.
SSN[23:0]	**Scrambling Sequence Number**. Valid only if *CG* is enabled. Indicates the number of the Scrambling sequence. Supported values: 0 to 2^{24} - 1.

Table 26-242. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x50

Name	Description
CL[12:0]	**Code Length**. Valid only if *CG* is enabled. Indicates the number of scrambled symbols to be generated. For details see **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*. Supported values: 0 to 2048 which are products of 4.
SN[3:0]	**Slot Number**. Valid only if *CG* is enabled. Indicates the index of slot in the frame. For details see **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*. Supported values: 0 to 14.

Table 26-243. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x54

Name	Description
NNPIS[2:0]	**Next Number of Pilots State**. Valid only if *CG* is enabled. A status field which indicates the number of pilots to be taken for the next job. For details see **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*. Supported reported values: 0 to 5 represents number of pilots 3 to 8.
NSGO[3:0]	**Next Slot Generation Off**. Valid only if *CG* is enabled. A status field which indicates that slot generation is off for next job. Bit [0] of this field represent the *SGO* bit for the next job position, and the following bits represent the next 3 slots. For details see **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*. Supported reported values: **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*.
NSN[3:0]	**Next Slot Number**. Valid only if *CG* is enabled. A status field which indicates the slot number of next chip to be generated for next job. For details see **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*. Supported reported values: 0 to 14.
NCO[15:0]	**Next Chip Offset**. Valid only if *CG* is enabled. A status field which indicates the chip offset to be taken as starting point for scrambling code generation for next job. The reported chip offset is the next chip following the last chip included in the current code generation processing. For details see **Section 26.4.3.3.8**, *WCDMA Scrambled Pilot Code Generation*. Supported reported values: 0 to 38399.

Table 26-244. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x58

Name	Description
[31:0]	Reserved

Table 26-245. eFTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x5C

Name	Description
[31:0]	Reserved

26.5.4.3.3 Transform Length Encoding

Table 26-246 describe the transform length size encoding for the [TL_ID] field in the eFTPE BDs.

Table 26-246. [TL_ID] Field Encoding for the Different Transform Lengths of the eFTPE BD

[TL_ID] Field in eFTPE BD	Transform Length	[TL_ID] Field in eFTPE BD	Transform Length	[TL_ID] Field in eFTPE BD	Transform Length	[TL_ID] Field in eFTPE BD	Transform Length
0	12	10	180	20	480	30	972
1	24	11	192	21	540	31	1080
2	36	12	216	22	576	32	1152
3	48	13	240	23	600	33	1200
4	60	14	288	24	648	34	1536
5	72	15	300	25	720	35	128

Table 26-246. [TL_ID] Field Encoding for the Different Transform Lengths of the eFTPE BD

[TL_ID] Field in eFTPE BD	Transform Length	[TL_ID] Field in eFTPE BD	Transform Length	[TL_ID] Field in eFTPE BD	Transform Length	[TL_ID] Field in eFTPE BD	Transform Length
6	96	16	324	26	768	36	256
7	108	17	360	27	864	37	512
8	120	18	384	28	900	38	1024
9	144	19	432	29	960	39	2048

26.5.4.4 DEPE BD and Header Structures

The DEPE uses BDs to configure the general DEPE operation. In addition, the DEPE uses header structures to specialize the operations for 3GLTE, WiMAX, and UMTS operations.

26.5.4.4.1 DEPE Buffer Descriptor Structure

A description of the DEPE Buffer Descriptor structure expected by the host for the DEPE BD rings is outlined in **Figure 26-251**. Some of the fields are status fields which are written by the MAPLE-B2 upon job completion. Other fields may be relevant only for certain configurations and is ignored by the MAPLE-B2 if another configuration is used. The total length of the BD is 64 bytes, and its size does not depend on the configuration, that is, the next BD location should be located 64 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		INT_EN		16M	STD		IBS							
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NOH					OBS										
W																
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															
W																

Figure 26-251. DEPE Buffer Descriptor Structure

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

Offset	BD_BASE + 0x10															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BD_STS_PTR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BD_STS_PTR															
W																

Offset	BD_BASE + 0x14															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MB_PR			OBO					TASK_ID							
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																COE
W																

Figure 26-251. DEPE 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	ESVO					PFS										
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SVE	TB				PSS										
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		RMNA														
W																

Offset	BD_BASE + 0x20															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DEPE_HEADER_1															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DEPE_HEADER_1															
W																

Offset	BD_BASE + 0x24															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DEPE_HEADER_2															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DEPE_HEADER_2															
W																

Figure 26-251. DEPE Buffer Descriptor Structure (Continued)

Offset	BD_BASE + 0x28															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DEPE_HEADER_3															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DEPE_HEADER_3															
Offset	BD_BASE + 0x2C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DEPE_HEADER_4															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DEPE_HEADER_4															
Offset	BD_BASE + 0x30															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DEPE_HEADER_5															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DEPE_HEADER_5															
Offset	BD_BASE + 0x34															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DEPE_HEADER_6															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DEPE_HEADER_6															

Figure 26-251. DEPE 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	DEPE_HEADER_7															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DEPE_HEADER_7															
Offset	BD_BASE + 0x3C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DEPE_HEADER_8															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DEPE_HEADER_8															

Figure 26-251. DEPE Buffer Descriptor Structure (Continued)

Table 26-247. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0

Name	Description
OWNER	**Owner bit**. When set by the host, MAPLE-B2 is the BD owner and the job is not complete. When cleared by MAPLE-B2, the job is done and the host is the BD owner. 0—The host is BD owner 1—MAPLE-B2 is BD owner
WRAP	**Wrap bit**. When set by the host, MAPLE-B2 updates [BDR_RD_PTR] with [BDR_BA] (see **Section 26.5.4.1.4**), that is, it expects to find the next BD at the Base Address of the ring. When cleared by the host, MAPLE-B2 expects to find the next BD at: [BDR_RD_PTR] + BD size.
INT_EN	**Interrupt Enable**. If set, MAPLE-B2 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.
16M	**WiMAX 802.16m**. If set, indicates the MAPLE-B2 that this DEPE job is to be processed using the 802.16m parameters. 0—The job is to be processed assuming WiMAX 802.16e is required. 1—The job is to be processed assuming WiMAX 802.16m is required.
STD	**Standard**. Indicates to which of the two optional supported technologies this job relates in a given MAPLE-B2 operation mode. 0—The current job is UMTS or WiMAX job for 3G or WiMAX operation mode respectively. 1—The current job is 3GLTE job
IBS[7:0]	**Input Buffer Size**. Describes the number of 32 bits words which should be fetched by MAPLE-B2 to execute the current BD. If the current BD includes multiple Headers ([NOH]>0), the IBS should describe the input data size of all Headers. For details, see **Section 26.4.3.4.4.2**, *Input Data Structure for Multiple Tasks*. For UMTS with [TB] bit set, this field is ignored. For details, refer to **Section 26.4.3.4.5.4.6**, *UMTS Transport Block (TB) Support*. 0x2 - 0xC8 possible values representing 8 to 800 bytes.

Table 26-247. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0 (Continued)

Name	Description
NOH[4:0]	**Number of Headers**. Describes the number of Headers related to this BD. If set to Zero and [HBA] is also set to zero, the MAPLE-B2 expect to find the DEPE Header inside the BD, else the MAPLE-B2 fetches the Header from system memory address pointed by [HBA]. 00000—The Current job includes 1 Header. 00001—The current job includes 2 Headers. ... 11111—The current job includes 32 Headers.
OBS[10:0]	**Output Buffer Size**. Describes the total number of 32 bits words which should be written by MAPLE-B2 from DEPE output buffer and into the system memory on job completion.If the current BD includes multiple Headers ([NOH]>0), the [OBS] should describe the output data size of all Headers. For 3GLTE standard this field is ignored. For UMTS Separate Vectors this field indicates the size of the systematic vector only. For details, see **Section 26.4.3.4.6**, *DEPE Output Data Structure*. For UMTS Transport Block BD, this field indicates the total size of the TB (see **Section 26.4.3.4.5.4.6**, *UMTS Transport Block (TB) Support*). 0x2 to 0x580 possible values representing 8 bytes to 5.5 Kbytes.

Table 26-248. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4

Name	Description
IBA[31:2]	**Input Buffer Address**. This field points to the input buffer location in system memory, where MAPLE-B2 is to fetch the data into the DEPE input buffer. The address pointer to system memory must be 4 bytes aligned.

Table 26-249. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x8

Name	Description
OBA[31:2]	**Output Buffer Address**. This field points to the output buffer location in system memory, where MAPLE-B2 is to write the data from the DEPE output buffer. The address pointer to the output buffer in system memory must be 4 bytes aligned.

Table 26-250. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0xC

Name	Description
HBA[31:3]	**Header Buffer Address**. This field points to the DEPE Headers buffer location in the system memory if multiple Headers are included in this job ([NOH]>0). If single Header is included in this job ([NOH]=0), this field is optional: by clearing it to zero the MAPLE-B2 takes the Header from the BD (offsets 0x20 to 0x3C). By assigning the [HBA] with any value different than zero, the MAPLE-B2 fetches the single Header from system memory as pointed by this field and ignores the Header fields in the BD. The address pointer must be 8 bytes aligned.

Table 26-251. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x10

Name	Description
BD_STS_PTR[31:2]	**BD Status Pointer**. Points to an address in system memory where MAPLE-B2 is to write the RMNA status fields in case [NOH] > 0 or in case [NOH] = 0 and this field is not zeroed. For details, see **Section 26.4.3.4.5.1**, *3GLTE Processing*.

Table 26-252. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x14

Name	Description
MB_PR[1:0]	**MBus Priority**. Valid only if the [MB_PR_SCH] of the MMC0P parameter (see **Section 26.5.2.1**, *MAPLE Mode Configuration 0 Parameter (MMC0P)*) is set to '01' or '10'. Indicated the MBus accesses priority related to this BD. For details, see **Section 26.4.2.4**, *MBus Priority Scheme Configuration* 00— The MBus accesses related to that BD are initiated with priority '00' ... 11—The MBus accesses related to that BD are initiated with priority '11'
OBO[4:0]	**Output Buffer Offset**. Indicates the value of the Output Buffer Offset filed of the DEPE Header. Used by the MAPLE-B2 to calculate the amount of data to be output from the DEPE output buffer and into the system memory.
TASK_ID[7:0]	**Task ID**. Supplied by host and used by MAPLE-B2 if a serial RapidIO doorbell interrupt is required. For details, refer to **Section 26.4.4, *External Masters Support Using Serial RapidIO Doorbell***
COE	**Concatenate Output Enable**. Valid only for single Header job ([NOH]=0).If set, the MAPLE-B2 merges the first 32 bits word of the DEPE output buffer with the first 32 bits word of the output buffer in the system memory pointed by [OBA], thus allowing concatenating the current job to a previous job in the system memory. For details, refer to **Section 26.4.3.4.6.1**, *Output Buffer Offset and Concatenate Output Support (3GLTE and UMTS).* 0—Concatenate Output is disabled. MAPLE-B2 output the DEPE output buffer as is. 1—Concatenate Output is enabled. MAPLE-B2 merges the first word of the DEPE output buffer with the first word of the system output buffer.

Table 26-253. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x18

Name	Description
ESVO[3:0]	**Encoded Separate Vectors Offset**. Valid only for UMTS standard and when Separate vectors output data structure is enabled ([SVE] = 1). Indicates the MAPLE-B2 the offset between the 3 output vectors in the system memory. For details, refer to **Section 26.4.3.4.6.3**, *Separate Vectors Output Data Structure for UMTS*. 0000—The offset between the vectors in system memory is 128 bits 0001—The offset between the vectors in system memory is 256 bits 0010—The offset between the vectors in system memory is 512 bits 0011—The offset between the vectors in system memory is 1K bits 0100—The offset between the vectors in system memory is 2K bits 0101—The offset between the vectors in system memory is 4K bits 0110—The offset between the vectors in system memory is 8K bits 0111—The offset between the vectors in system memory is 16K bits 1000—The offset between the vectors in system memory is 32K bits 1001—The offset between the vectors in system memory is 64K bits 1010—The offset between the vectors in system memory is 128K bits 1011 to 1111—Reserved.
PFS[10:0]	**Parity First Size**. Valid only for UMTS standard and when Separate vectors output data structure is enabled ([SVE] = 1). Describes the total number of 32 bits words which should be written by MAPLE-B2 from DEPE output buffer and into the system memory to output the parity first vector from DEPE output buffer.For details, refer to **Section 26.4.3.4.6.3**, *Separate Vectors Output Data Structure for UMTS*. 0x1 to 0x7FF possible values representing 4 bytes to 2 Kbytes.

Table 26-253. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x18 (Continued)

Name	Description
SVE	**Separate Vectors Enable**. Valid only for UMTS standard and for single Header BD (NOH=0). Indicates the MAPLE-B2 that the Bit Collection is disabled hence the DEPE outputs 3 separate vectors and not single stream. For details, see **Section 26.4.3.4.6.3**, *Separate Vectors Output Data Structure for UMTS*. 0—Separate Vectors mode is disabled. 1—Separate Vectors mode is enabled.
TB	**Transport Block indication.** Valid only for UMTS standard and for certain configuration of the Bit Collection Scheme. Indicates the MAPLE-B2 that the current BD represent a full Transport Block job for the DEPE. The MAPLE-B2 uses the single Header attached to the BD (either in the BD or pointed by the [HBA] field of the DEPE BD) to internally generate the Headers for all CBs composing this TB. For more detail, refer to **Section 26.4.3.4.5.4.6**, *UMTS Transport Block (TB) Support*. 0— The current job is a Code Block job. 1— The current job is a Transport Block job.
PSS[10:0]	**Parity Second Size**. Valid only for UMTS standard and when Separate vectors output data structure is enabled ([SVE] = 1). Describes the total number of 32 bits words which should be written by MAPLE-B2 from DEPE output buffer and into the system memory to output the parity second vector from DEPE output buffer.For details, refer to **Section 26.4.3.4.6.3**, *Separate Vectors Output Data Structure for UMTS*. 0x1 to 0x7FF possible values representing 4 bytes to 2 Kbytes.

Table 26-254. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x1C

Name	Description
RMNA[14:0]	**Rate Matching Number of Actual output bits**. Relevant only for 3GLTE. Describes the actual number of output bits generated by the DEPE and output to system memory by MAPLE-B2.if [NOH] > 0 the RMNA status field of all the tasks is concatenated by MAPLE-B2 and written to the system memory to the location pointed by [BD_STS_PTR]. For details, see **Section 26.4.3.4.5.1**, *3GLTE Processing*

Table 26-255. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x20

Name	Description
DEPE_HEADER_1[31:0]	**First 4 bytes word of DEPE Header**. Relevant for all Standards. See **Section 26.5.4.4.3**, *DEPE Headers Structure*

Table 26-256. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x24

Name	Description
DEPE_HEADER_2[31:0]	**Second 4 bytes word of DEPE Header**.Relevant for all Standards. See **Section 26.5.4.4.3**, *DEPE Headers Structure*

Table 26-257. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x28

Name	Description
DEPE_HEADER_3[31:0]	**Third 4 bytes word of DEPE Header**.Relevant for 3GLTE and UMTS Standards only. See **Section 26.5.4.4.3**, *DEPE Headers Structure*

Table 26-258. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x2C

Name	Description
DEPE_HEADER_4[31:0]	**Fourth 4 bytes word of DEPE Header**. Relevant for 3GLTE and UMTS Standards only. See **Section 26.5.4.4.3**, *DEPE Headers Structure*

Table 26-259. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x30

Name	Description
DEPE_HEADER_5[31:0]	**Fifth 4 bytes word of DEPE Header**. Relevant for UMTS Standard only. See **Section 26.5.4.4.3**, *DEPE Headers Structure*

Table 26-260. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x34

Name	Description
DEPE_HEADER_6[31:0]	**Sixth 4 bytes word of DEPE Header**. Relevant for UMTS Standard only. See **Section 26.5.4.4.3**, *DEPE Headers Structure*

Table 26-261. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x38

Name	Description
DEPE_HEADER_7[31:0]	**Seventh 4 bytes word of DEPE Header**. Relevant for UMTS Standard only. See **Section 26.5.4.4.3**, *DEPE Headers Structure*

Table 26-262. DEPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x3C

Name	Description
DEPE_HEADER_8[31:0]	**Eighth 4 bytes word of DEPE Header**. Relevant for UMTS Standard only. See **Section 26.5.4.4.3**, *DEPE Headers Structure*

26.5.4.4.2 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-B2.
- All reserved bits in the BD must be set to zero for future compatibility.
- All the status fields in the BD are updated by the MAPLE-B2 once the job is complete, that is, they are valid only after the owner bit is cleared by the MAPLE-B2.
- 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 PSIF2 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.

26.5.4.4.3 DEPE Headers Structure

The DEPE Headers are used to provide the DEPE with the Code Block (CB) encoding parameters. Each DEPE CB encoding task requires its Header. A single DEPE BD may include up to 32 CBs encoding tasks, thus requires up to 32 Headers. The size of the DEPE Header differs according to the standard. The WiMAX Header is of 8 bytes size, the 3GTLE is of 16 bytes size and the UMTS is of 32 bytes size. If a DEPE BD includes single CB encoding task, its Header can be included in the DEPE BD as described in **Section 26.5.4.4.1**, *DEPE Buffer Descriptor Structure*. A description of the DEPE Headers structure for the different supported standards is outlined in sections **Section 26.5.4.4.3.1**, *DEPE Header Structure for 3GLTE* to **Section 26.5.4.4.3.4**, *DEPE Header Structure for UMTS*.

26.5.4.4.3.1 DEPE Header Structure for 3GLTE

The DEPE Header Structure for 3GLTE is outlined in **Figure 26-252**:

Offset	DEPE_HEADER_1															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W											NOF					

Offset	DEPE_HEADER_2															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W				OBO								IBO				
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	CBSI								CM	LH	CD	NOH				

Offset	DEPE_HEADER_3															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W		RMNO														
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W		RMS														

Offset	DEPE_HEADER_4															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W		RMNI														

Figure 26-252. DEPE Header Structure for 3GLTE

Table 26-263. DEPE Header Structure for 3GLTE - Field Description

Name	Description
NOF[5:0]	**Number of Filler bits**. Number of filler bits needed to be added to the current data block. For more detail see **Section 26.4.3.4.5.1**, *3GLTE Processing* 000000— No filler bits are needed 000001— 1 filler bit is needed ... 111111— 63 filler bits are needed
OBO[4:0]	Output Buffer Offset. Offset of the first bit of the encoded block in the DEPE output buffer. Used for concatenating the current CB with previous CB from the same transport block in system memory. For details, see **Section 26.4.3.4.6.1**, *Output Buffer Offset and Concatenate Output Support (3GLTE and UMTS)*. 00000— The first bit of the encoded block is at offset 0. ... 11111— The first bit of the encoded block is at offset 31.
IBO[4:0]	**Input Buffer Offset**. Offset of the first bit of the input CB in the MAPLE-B2 input buffer pointed by [IBA] field of the DEPE BD. Used for supporting CB segmentation.Required due to the lack of filler bits in expected input. For details, see **Section 26.4.3.4.4.1**, *Input Buffer Offset Support (3GLTE and UMTS only)*. 00000— The first bit of the input CB is at offset 0. ... 11111— The first bit of the input CB is at offset 31.
CBSI [7:0]	Code Block Size Index Select one of 188 CB sizes supported in 3GLTE 0—CB size is 40 1—CB size is 44 ... 186— CB size is 6080 187— CB size is 6144 188 to 255— Reserved.
CM	CRC Mode Select the CRC polynomial. Valid only if [CD] = 0. For details, see **Section 26.4.3.4.5.1**, *3GLTE Processing*. 0— CB CRC polynomial ($D^{24} + D^{23} + D^6 + D^5 + D + 1$) 1— Transport block CRC polynomial ($D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1$)
LH	**Last Header indication**. Indicated the MAPLE-B2 that this Header is the last Header in the BD. If [NOH] > 0, this bit should be set only for the last Header. If [NOH] == 0, this bit should be set for the single Header. 0—This Header is **not** the last Header in the BD. 1—This Header is the last Header in the BD.

Table 26-263. DEPE Header Structure for 3GLTE - Field Description

Name	Description
CD	CRC Disable Indicates the DEPE whether to perform CRC calculation. If enabled, the CRC polynomial is determined using [CM]. 0—CRC is enabled. 1—CRC is disabled.
NOH[4:0]	**Number of Headers**. Number of Headers included in this job. 00000— The Current job includes 1 Header. 00001— The current job includes 2 Headers. ... 11111— The current job includes 32 Headers.
RMNO[14:0]	Rate Matching Number of Output bits The required number of encoded bits after rate matching. Since the DEPE does not support repetition, the value of this field must **not** be greater than the value of [RMNI].
RMS[14:0]	Rate Matching Start bit The starting bit in the virtual circular buffer, that is, the virtual circular buffer index of the first bit in the output stream. For details, refer to **Section 26.4.3.4.5.1**, *3GLTE Processing*.
RMNI[14:0]	Rate Matching Number of Input bits Number of encoded bits in the virtual circular buffer. For details, see **Section 26.4.3.4.5.1**, *3GLTE Processing*.

26.5.4.4.3.2 DEPE Header Structure for WiMAX (802.16e)

The DEPE Header Structure for WiMAX (802.16e) is outlined in **Figure 26-253**:

Offset	DEPE_HEADER_1															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W			RMNO													
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W			RMS													

Offset	DEPE_HEADER_2															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W				CBSI					RD	LH	CD	NOH				

Figure 26-253. DEPE Header Structure for WiMAX

Table 26-264. DEPE Header Structure for WiMAX - Field Description

Name	Description
RMNO[13:0]	Rate Matching Number of Output bits The required number of encoded bits after rate matching. Since the DEPE does not support repetition, the value of this field must **not** be greater than 3*(Code Block Size).
RMS[13:0]	Rate Matching Start bit The starting bit in the virtual circular buffer, that is, the virtual circular buffer index of the first bit in the output stream. For details, refer to **Section 26.4.3.4.5.2**, *WiMAX Processing (802.16e)*.
CBSI [4:0]	Code Block Size Index 0— CB size is 48 1— CB size is 72 2— CB size is 96 3— CB size is 144 4— CB size is 192 5— CB size is 216 6— CB size is 240 7— CB size is 288 8— CB size is 360 9— CB size is 384 10—CB size is 432 11— CB size is 480 12— CB size is 960 13— CB size is 1920 14— CB size is 2880 15— CB size is 3840 16— CB size is 4800 17-31— Reserved
RD	**Randomizer Disable** 0— Randomizer is enabled. 1— Randomizer is disabled.
LH	**Last Header indication**. Indicated the MAPLE-B2 that this Header is the last Header in the BD. If [NOH] > 0, this bit should be set only for the last Header. If [NOH] == 0, this bit should be set for the single Header. 0—This Header is **not** the last Header in the BD. 1—This Header is the last Header in the BD.
CD	CRC Disable Indicates the DEPE whether to perform CRC calculation. 0—CRC is enabled. 1—CRC is disabled.
NOH[4:0]	**Number of Headers**. Number of Headers included in this job. 00000— The Current job includes 1 Header. 00001— The current job includes 2 Headers. ... 11111— The current job includes 32 Headers.

26.5.4.4.3.3 DEPE Header Structure for WiMAX (802.16m)

The DEPE Header Structure for WiMAX (802.16m) is outlined in **Figure 26-254**:

Offset	DEPE_HEADER_1															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W	CLSB		RMNO													
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W		CLSY														

Offset	DEPE_HEADER_2															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W		CLSW														
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W			CBSI						RD	LH	CD	NOH				

Figure 26-254. DEPE Header Structure for WiMAX

Table 26-265. DEPE Header Structure for WiMAX - Field Description

Name	Description
CLSB	Circularly Left Shift B Sub-block. 0— B Sub-block is circularly left shift by 0 bits. 1— B Sub-block is circularly left shift by 1 bit.
RMNO[13:0]	Rate Matching Number of Output bits The required number of encoded bits after rate matching. As the DEPE supports internal bit collection only (no bit selection support), this field must be set with **CB_Size * 3**, where CB_Size is the actual Code Block Size encoded in the CBSI field. Valid Range: 3 * Code Block Size
CLSY	Circularly Left Shift Y Sub-block. 0— Y Sub-block is circularly left shift by 0 bits. 1— Y Sub-block is circularly left shift by 1 bit.
CLSW	Circularly Left Shift W Sub-block. 0— W Sub-block is circularly left shift by 0 bits. 1— W Sub-block is circularly left shift by 1 bit.
CBSI [5:0]	**Code Block Size Index**. See **Table 26-266**

Table 26-265. DEPE Header Structure for WiMAX - Field Description

Name	Description
RD	**Randomizer Disable**. Must be set to 1 as Randomization is not supported for 802.16m
LH	**Last Header indication**. Indicated the MAPLE-B2 that this Header is the last Header in the BD. If [NOH] > 0, this bit should be set only for the last Header. If [NOH] == 0, this bit should be set for the single Header. 0—This Header is **not** the last Header in the BD. 1—This Header is the last Header in the BD.
CD	CRC Disable. Must be set to 1 as CRC calculation is not supported for 802.16m
NOH[4:0]	**Number of Headers**. Number of Headers included in this job. 00000— The Current job includes 1 Header. 00001— The current job includes 2 Headers. ... 11111— The current job includes 32 Headers.

Table 26-266. CBSI Encoding for 802.16m

CBSI	Code Block Size	CBSI	Code Block Size	CBSI	Code Block Size
0	48	**15**	320	**30**	1856
1	64	**16**	352	**31**	2112
2	72	**17**	400	**32**	2368
3	80	**18**	456	**33**	2624
4	88	**19**	512	**34**	2944
5	96	**20**	568	**35**	3328
6	104	**21**	640	**36**	3776
7	120	**22**	720	**37**	4224
8	136	**23**	800	**38**	4800
9	152	**24**	912	**39-63**	Reserved
10	176	**25**	1024		
11	200	**26**	1152		
12	216	**27**	1312		
13	248	**28**	1440		
14	288	**29**	1632		

26.5.4.4.3.4 DEPE Header Structure for UMTS

The DEPE Header Structure for UMTS is outlined in **Figure 26-255**:

Offset	DEPE_HEADER_1															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W	PFEM2															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	PFEM2			CBS												
Offset	DEPE_HEADER_2															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W	BCT			OBO								IBO				
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	CBC	P				BCM	REP	RRM	SD	LH	CD	NOH				
Offset	DEPE_HEADER_3															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W	PSEM2															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	PEM1															
Offset	DEPE_HEADER_4															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W	SEI															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	SEM															
Offset	DEPE_HEADER_5															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W	PSEI2															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	PFEI2															
Offset	DEPE_HEADER_6															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W	PSEI1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	PFEI1															
Offset	DEPE_HEADER_7															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W	PSO															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	PFO															
Offset	DEPE_HEADER_8															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W	NCB													EPFEI2		
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	EPSEI2		EPSEI1		EPFEI1			EPSEM2		EPEM1			ESEI		ESEM	

Figure 26-255. DEPE Header Structure for UMTS

Table 26-267. DEPE Header Structure for UMTS - Field Description

Name	Description
PFEM2[18:0]	Parity First E_{minus} #2 Error decrement variable (e_{minus}) for second rate matching stage of the parity first stream.
CBS[12:0]	Code Block Size The number of bits in the current CB. The size can vary from 40 to 5114. Other values are reserved.
BCT[1:0]	Bit Collection Type 00— Systematic, first parity and second parity are collected in separate vectors. 01— Systematic, first parity and second parity bits are interleaved by a rectangular interleaver with 2 rows 10— Systematic, first parity and second parity bits are interleaved by a rectangular interleaver with 4 rows 11— Systematic, first parity and second parity bits are interleaved by a rectangular interleaver with 6 rows
OBO[4:0]	Output Buffer Offset. Offset of the first bit of the encoded block in the DEPE output buffer. Used for concatenating the current CB with previous CB from the same transport block in system memory. For details, see **Section 26.4.3.4.6.1**, *Output Buffer Offset and Concatenate Output Support (3GLTE and UMTS)*. 00000— The first bit of the encoded block is at offset 0. ... 11111— The first bit of the encoded block is at offset 31.
IBO[4:0]	**Input Buffer Offset**. Offset of the first bit of the input CB in the MAPLE-B2 input buffer (in system memory) pointed by [IBA] field of the DEPE BD. Used for supporting CB segmentation. For details, see **Section 26.4.3.4.4.1**, *Input Buffer Offset Support (3GLTE and UMTS only)*. 00000— The first bit of the input CB is at offset 0. ... 11111— The first bit of the input CB is at offset 31.
CBC	**Continuous Bit Collection**. Indicates the DEPE that the current CB is a following CB in a current TB executed successively and in order in the DEPE to support full TB bit collection. For details, see **Section 26.4.3.4.5.4.6**, *UMTS Transport Block (TB) Support* 0— Continuous Bit Collection is disabled. 1— Continuous Bit Collection is enabled.
P[3:0]	Number of Physical channels This field indicates the number of physical channel allocated for the current job. This is ignored if bit collection is not done by a rectangular interleaver (that is, if BCM=0 or BCT=0). 0— reserved 1— one physical channel is allocated for the current job. 2— two physical channels are allocated for the current job. ... 15—fifteen physical channels are allocated for the current job.
BCM	Bit Collection Mode 0— Systematic, first parity and second parity are interlaced. 1— Bit collection is done according to BCT
REP	Repetition. For details, refer to **Section 26.4.3.4.5.4**, *UMTS Processing*. 0— RM2 implements puncturing. 1— RM2 implements repetition.
RRM	Reset Rate Matching machine Indicates the DEPE whether to reset the error parameters or to resume with the current internal error parameters values assuming the current CB is the successive CB in the transport block of the previous task. For details, see **Section 26.4.3.4.5.4**, *UMTS Processing*. 0— Rate matching machines are not reset and initial error parameters are not loaded to the RM machines 1— Rate matching machines are reset and are load with initial error parameters.

Table 26-267. DEPE Header Structure for UMTS - Field Description

Name	Description
SD	Scrambling Disable Indicated the DEPE whether to perform scrambling. Should be enabled for FDD processing only. 0— Scrambling is enabled. 1— Scrambling is disabled.
LH	**Last Header indication**. Indicated the MAPLE-B2 that this Header is the last Header in the BD. If [NOH] > 0, this bit should be set only for the last Header. If [NOH] == 0, this bit should be set for the single Header. 0—This Header is **not** the last Header in the BD. 1—This Header is the last Header in the BD.
CD	CRC Disable Indicates the DEPE whether to perform CRC calculation. 0—CRC is enabled. 1—CRC is disabled.
NOH[4:0]	**Number of Headers**. Number of Headers included in this job. 00000— The Current job includes 1 Header. 00001— The current job includes 2 Headers. ... 11111— The current job includes 32 Headers.
PSEM2[15:0]	Parity Second E_{minus} #2 Error decrement variable (e_{minus}) for the second RM stage of the second parity stream.
PEM1[15:0]	Parity E_{minus} #1 This parameter is used to calculate the error decrement variable (e_{minus}) for the first RM stage of the parity streams. For E-DCH processing must be disabled by clearing to zero.
SEI[15:0]	Systematic E_{ini} Initial error variable (e_{ini}) for systematic stream RM.
SEM[15:0]	Systematic E_{minus} Error decrement variable (e_{minus}) for systematic stream RM.
PSEI2[15:0]	Parity Second E_{ini} #2 Initial error variable (e_{ini}) for the second rate matching stage of the second parity stream.
PFEI2[15:0]	Parity First E_{ini} #2 Initial error variable (e_{ini}) for the second rate matching stage of the first parity stream.
PSEI1[15:0]	Parity Second E_{ini} #1 Initial error variable (e_{ini}) for the first rate matching stage of the second parity stream. For E-DCH processing must be disabled by assigning to any value bigger than zero.
PFEI1[15:0]	Parity First E_{ini} #1 Initial error variable (e_{ini}) for the first rate matching stage of the first parity stream. For E-DCH processing must be disabled by assigning to any value bigger than zero.
PSO[15:0]	Parity Second Offset This field is valid when [BCM] is set, and indicates the offset of the parity second bits in the output vector
PFO[15:0]	Parity First Offset This field is valid when [BCM] is set, and indicates the offset of the parity first bits in the output vector
NCB[5:0]	Number of Code Block Indicates the number of CBs in the current TB. Used for calculating the X parameter. For details, see **Section 26.4.3.4.5.4**, *UMTS Processing*
EPFEI2[2:0]	Extension for Parity First E_{ini} #2 3 bits extension for [PFEI2]

Table 26-267. DEPE Header Structure for UMTS - Field Description

Name	Description
EPSEI2[1:0]	Extension for Parity Second E_{ini} #2 2 bits extension for [PSEI2].
EPSEI1[1:0]	Extension for Parity Second E_{ini} #1 2 bits extension for [PSEI1]
EPFEI1[2:0]	Extension for Parity First E_{ini} #1 3 bits extension for [PFEI1]
EPSEM2[1:0]	Extension for Parity Second E_{minus} #2 2 bits extension for [PSEM2]
EPEM1[2:0]	Extension for Parity E_{minus} #1 3 bits extension for [PEM1]. For E-DCH processing must be disabled by clearing to zero.
ESEI[1:0]	Extension for systematic E_{ini} 2 bits extension for [SEI]
ESEM[1:0]	Extension for Systematic E_{minus} 2 bits extension for [SEM]

26.5.4.5 EQPE BD Structure

A description of the EQPE Buffer Descriptor structure expected by the host for the EQPE BD rings is outlined in **Figure 26-256**. Some of the fields are status fields which are written by the MAPLE-B2 upon job completion. Other fields may be relevant only for certain configurations and is ignored by the MAPLE-B2 if another configuration is used. The total length of the BD is 128 bytes, and its size does not depend on the configuration, that is, the next BD location should be located 128 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		INT_EN	POST_IDFT	FH	ALG		INTRP		LX			RX		
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COLS				ROWS											
W																
Offset	BD_BASE + 0x4															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MB_PR		PST_SCLR	F_DFT	C_EN	F_EN	LTC		ALIGN0		ALIGN1		ALIGN2		ALIGN3	
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LD0	LD1	LD2	LD3	PST_VEC	M			MOD0		MOD1		MOD2		MOD3	
W																
Offset	BD_BASE + 0x8															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ALIGN_VAL0								ALIGN_VAL1							
W																

Figure 26-256. EQPE Buffer Descriptor Structure

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ALIGN_VAL2								ALIGN_VAL3							
W																
Offset	BD_BASE + 0xC															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									W_ROW							
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R				S_ TYPE	S_COL				S_ROW							
W																
Offset	BD_BASE + 0x10															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	W_SIZE															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	S_SIZE															
W																
Offset	BD_BASE + 0x14															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TASK_ID															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R							RM_S CL_TY PE	S_SC L_TYP E	C_SCL							
W																
Offset	BD_BASE + 0x018															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RR0															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RR1															
W																
Offset	BD_BASE + 0x1C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RR2															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RR3															
W																
Offset	BD_BASE + 0x20															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Y_POINTER															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Y_POINTER															
W																

Figure 26-256. EQPE 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			Y_OFFSET													
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			Y_GAP													
W																
Offset	BD_BASE + 0x28															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	H_POINTER0 / R_POINTER / M_POINTER															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	H_POINTER0 / R_POINTER / M_POINTER															
W																
Offset	BD_BASE + 0x2C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	H_POINTER1															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	H_POINTER1															
W																
Offset	BD_BASE + 0x30															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	H_POINTER2															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	H_POINTER2															
W																
Offset	BD_BASE + 0x34															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	H_POINTER3															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	H_POINTER2															
W																
Offset	BD_BASE + 0x38															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	S_POINTER															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	S_POINTER															
W																

Figure 26-256. EQPE Buffer Descriptor Structure (Continued)

Offset	BD_BASE + 0x3C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	W_POINTER															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	W_POINTER															
W																
Offset	BD_BASE + 0x40															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SCL_PTR_BA															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SCL_PTR_BA															
W																
Offset	BD_BASE + 0x44															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	S_SCL_OFFSET								H_SCL_OFFSET							
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Y_SCL_OFFSET								F_SCL_OFFSET							
W																
Offset	BD_BASE + 0x48															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OUT_POINTER0															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OUT_POINTER0															
W																
Offset	BD_BASE + 0x4C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OUT_POINTER1															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OUT_POINTER1															
W																
Offset	BD_BASE + 0x50															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OUT_POINTER2															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OUT_POINTER2															
W																

Figure 26-256. EQPE Buffer Descriptor Structure (Continued)

Offset	BD_BASE + 0x54															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OUT_POINTER3															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OUT_POINTER3															
W																
Offset	BD_BASE + 0x58															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	F_POINTER															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	F_POINTER															
W																
Offset	BD_BASE + 0x5C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	C_POINTER															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	C_POINTER															
W																
Offset	BD_BASE + 0x60															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OUT_SCL_OFF0								OUT_SCL_OFF1							
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OUT_SCL_OFF2								OUT_SCL_OFF3							
W																
Offset	BD_BASE + 0x64															
	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		SING														
W																
Offset	BD_BASE + 0x68															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PST_SCLR_PTR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PST_SCLR_PTR															
W																

Figure 26-256. EQPE Buffer Descriptor Structure (Continued)

Offset	BD_BASE + 0x6C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PST_VEC_PTR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PST_VEC_PTR															
W																
Offset	BD_BASE + 0x70															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			H_OFFSET													
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			H_GAP													CONS_H
W																
Offset	BD_BASE + 0x74															
	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 + 0x78															
	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 + 0x7C															
	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 26-256. EQPE Buffer Descriptor Structure (Continued)

Table 26-268. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0

Name	Description
OWNER	**Owner bit**. When set by the host, MAPLE-B2 is the BD owner and the job is not complete. When cleared by MAPLE-B2, the job is done and the host is the BD owner. 0—The host is BD owner 1—MAPLE-B2 is BD owner
WRAP	**Wrap bit**. When set by the host, MAPLE-B2 updates [BDR_RD_PTR] with [BDR_BA] (see **Section 26.5.4.1.4**), that is, it expects to find the next BD at the Base Address of the ring. When cleared by the host, MAPLE-B2 expects to find the next BD at: [BDR_RD_PTR] + BD size.

Table 26-268. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0 (Continued)

Name	Description
INT_EN	**Interrupt Enable**. If set, MAPLE-B2 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.
POST_IDFT	**Post IDFT**. Valid only for LNEQ processing. If set, MAPLE-B2 performs additional iDFT processing on the EQPE output columns. This operation is performed internally using one of the eFTPE machines of the MAPLE-B2. For details see **Section 26.4.3.5.9**, *Pre/Post DFT/iDFT Processing*. 0—Post iDFT processing is disabled. No iDFT processing is performed on the EQPE outputs. 1—Post iDFT processing is enabled.
FH	**Full / Hermitian Matrix**. Valid only for Matrix Inversion processing (ALG=3). Indicates if the input and output matrices are Hermitian or Full. 0—The input and output matrices are Hermitian. 1—The input and output matrices are Full.
ALG[1:0]	**Algorithm**. Indicates the required processing type of the job. 0—Linear Equalization (LNEQ). 1—Maximum Likelihood Equalization (MLEQ). 2—Tree Search processing (TS). 3—Matrix Inversion processing (MIV).
INTRP[1:0]	**Interpolation mode**. Valid only for LNEQ and MLEQ processing (ALG=0,1). Indicates the interpolation mode for the current job. For details see **Section 26.4.3.5.5.7**, *CE Matrix Interpolation*. 0—II2 (Internal Interpolation with 2 references). 1—II4 (Internal Interpolation with 4 references). 2—External Interpolation 3—No Interpolation. **Note:** II4 is not supported for the case of Rx=7 with Lx=2,3 (8x4 and 8x3 antenna configurations).
Lx[1:0]	For LNEQ, MLEQ and TS processing (ALG=0,1,2): **Number of Layers**. Indicates the number of the transmit layers. 0—1 Layer. 1—2 Layers. 2—3 Layers. 3—4 Layers. For MIV processing (ALG=3): **Matrix Dimension**. Indicates the dimensions of the input matrix to invert. 0—1x1. 1—2x2. 2—3x3. 3—4x4.
Rx[2:0]	**Number of Receive antennas**. Valid only for LNEQ and MLEQ processing (ALG=0,1). Indicates the number of receive antennas. 0—1 Rx antenna. 1—2 Rx antennas. 2—Reserved. 3—4 Rx antennas. 4 to 6—Reserved. 7—8 Rx antennas.

Table 26-268. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0 (Continued)

Name	Description
COLS[3:0]	**Number of Columns**. Valid only for LNEQ, MLEQ and TS modes (ALG=0,1,2). Indicates the number of columns in the sub-carrier grid. For details see **Section 26.4.3.5.3**, *EQPE Sub-Carrier Grid*. 0—1 column 1—2 columns. ... 15—16 columns. **Note:** For LNEQ mode the allowed range of COLS is 0 to 11.

ROWS[11:0]	For LNEQ, MLEQ and TS processing modes (ALG=0,1,2):
	Number of Rows. Indicates the number of rows in the sub-carrier grid. For details see **Section 26.4.3.5.3**
	• Valid Range for LNEQ:
	if (*Lx* = 0,1) then supported values are: 11 to 2047 indicating 12 to 2048 rows respectively.
	if (*Lx* = 2,3) the supported values are: 11 to 1991 indicating 12 to 1992 rows respectively.
	Note: In LNEQ mode (ROWS+1) must be a multiplication of 12
	• Valid Range for MLEQ and TS[1]:
	if (*Lx* = 0,1) then supported values are: 0 to 2047 indicating 1 to 2048 rows respectively.
	if (*Lx* = 2,3) the supported values are: 0 to 1356 indicating 1 to 1357 rows respectively.
	For MIV processing: (ALG=3): **Number of matrices**. Indicates the number of matrix to invert in current job. Valid Range: 0 to 4095 indicating 1 to 4096 matrices respectively.

1. In TS mode only, if *R_SCL_TYPE*=1 then the size of the sub-carrier grid is limited to 4096, that is (*ROWS*+1)*(*COLS*+1) <= 4096

Table 26-269. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4

Name	Description
MB_PR[1:0]	**MBus Priority**. Valid only if the *MB_PR_SCH* of the MMC0P parameter (see **Section 26.5.2.1**, *MAPLE Mode Configuration 0 Parameter (MMC0P)*) is set to '01' or '10'. Indicated the MBus accesses priority related to this BD. For details, see **Section 26.4.2.4**, *MBus Priority Scheme Configuration* 00— The MBus accesses related to that BD are initiated with priority '00' ... 11—The MBus accesses related to that BD are initiated with priority '11'
PST_SCLR	**Post Scalar Multiplication**. Valid only if the *POST_IDFT* bit is set. If enabled, the iDFT results (*(COLS+1)*(Lx+1)* different results) are multiplied by different scalar values specified in the address pointed by the *PST_SCLR_PTR*. For details see **Section 26.4.3.5.9.2**, *Post iDFT Processing*. 0—Post scalar multiplication is disabled. 1—Post scalar multiplication is enabled.
F_DFT	**Feedback DFT**. Valid only for LNEQ processing and if the F_EN bit is set. If set, MAPLE-B2 performs DFT processing on the feedback cancellation inputs prior to the EQPE processing. For details see **Section 26.4.3.5.9**, *Pre/Post DFT/iDFT Processing*. 0—Pre DFT processing of the feedback inputs is disabled. 1—Pre DFT processing of the feedback inputs is enabled.
C_EN	**C Matrix Enable**. Valid only for LNEQ processing (ALG=0). Indicates whether external C matrix is to be used for the current job or assume internally that C=I (Identity matrix). For details see **Section 26.4.3.5.4.6**, *C Samples*. 0—EQPE assume C=I (no need to supply C matrix) 1—EQPE uses external C matrix.

Table 26-269. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4

Name	Description
F_EN	**Feedback Cancellation Enable**. Valid only for LNEQ processing. Indicates that feedback cancellation is required for the current job. For details see **Section 26.4.3.5.4.8**, *F Samples*. 0—Cancellation is disabled for the current job. 1—Cancellation is enabled for the current job.
LTC[1:0]	**Layer To Cancel**. Valid only for LNEQ processing and when the F_EN bit is set. Indicates which layer is required to be cancelled for the current job. 0—Perform cancellation of layer 0 1—Perform cancellation of layer 1 2—Perform cancellation of layer 2 3—Perform cancellation of layer 3 **Note:** LTC configuration must maintain the following: LTC <= Lx.
ALIGN0[1:0]	**Output Alignment Type #0**. Indicates the output alignment type of layer #0 for LNEQ, MLEQ and TS modes, or the output alignment type for MIV mode. For details see **Section 26.4.3.5.6.2**, *Output Samples Alignment (Scaling)*. 0—Max scale output alignment (valid for LNEQ or MIV only). For LNEQ it means max scale per column. For MIV it indicates single scale per matrix. 1—User defined output alignment. 2—No alignment, that is, scale per output sample (floating point representation) 3—Global max alignment. Valid for MIV only. Indicates shared scale for all output matrices.
ALIGN1[1:0]	**Output Alignment Type #1**. Indicates the output alignment type of layer #1 for LNEQ, MLEQ and TS modes. For details see **Section 26.4.3.5.6.2**, *Output Samples Alignment (Scaling)*. 0—Max scale output alignment per column (valid for LNEQ only). 1—User defined output alignment. 2—No alignment, that is, scale per output sample (floating point representation) 3—Reserved.
ALIGN2[1:0]	**Output Alignment Type #2**. Indicates the output alignment type of layer #2 for LNEQ, MLEQ and TS modes. For details see **Section 26.4.3.5.6.2**, *Output Samples Alignment (Scaling)*. 0—Max scale output alignment per column (valid for LNEQ only). 1—User defined output alignment. 2—No alignment, that is, scale per output sample (floating point representation) 3—Reserved
ALIGN3[1:0]	**Output Alignment Type #3**. Indicates the output alignment type of layer #3 for LNEQ, MLEQ and TS modes. For details see **Section 26.4.3.5.6.2**, *Output Samples Alignment (Scaling)*. 0—Max scale output alignment per column (valid for LNEQ only). 1—User defined output alignment. 2—No alignment, that is, scale per output sample (floating point representation) 3—Reserved
LD0	**Layer Discard of Layer #0**. Valid for LNEQ processing only. Indicates that layer #0 can be discarded. for details see **Section 26.4.3.5.6.3.3**, *Layer Discarding*. 0—Output layer #0 1—Discard layer #0.
LD1	**Layer Discard of Layer #1**. Valid for LNEQ processing only and if Lx >= 1. Indicates that layer #1 can be discarded. for details see **Section 26.4.3.5.6.3.3**, *Layer Discarding*. 0—Output layer #1 1—Discard layer #1
LD2	**Layer Discard of Layer #2**. Valid for LNEQ processing only and if Lx >= 2. Indicates that layer #2 can be discarded. for details see **Section 26.4.3.5.6.3.3**, *Layer Discarding*. 0—Output layer #2 1—Discard layer #2

Table 26-269. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4

Name	Description
LD3	**Layer Discard of Layer #3**. Valid for LNEQ processing only and if Lx >= 3. Indicates that layer #3 can be discarded. for details see **Section 26.4.3.5.6.3.3**, *Layer Discarding*. 0—Output layer #3 1—Discard layer #3
PST_VEC	**Post Vector Multiplication**. Valid only if the *POST_IDFT* bit is set. If enabled, the iDFT results are multiplied by the vectors pointed by the *PST_VEC_PTR* pointer of the EQPE BD. The MAPLE-B2 supports shared vector for all iDFT results of each layer (a total of *Lx* vectors).For details see **Section 26.4.3.5.9.2**, *Post iDFT Processing*. 0—Post vector multiplication is disabled. 1—Post vector multiplication is enabled.
M[2:0]	**M parameter**. Tree search width for QRD-M Equalization. Valid only for $Lx \geq 1$. For details see **Section 26.4.3.5.5.3**, *QRD-M Equalization (MLEQ)*. 0—M=4. 1—M=8 2—M=16 3—M=32 4—M=64. 5 to 7—Reserved **Note:** M=64 is not supported for $Lx > 1$.
MOD0[1:0]	**Modulation of layer #0**. Valid for MLEQ and TS processing only (ALG=1,2). Indicates the modulation type of layer #0. For details see **Section 26.4.3.5.8.2**, *EQPE Outputs for MLEQ and TS modes* 0—QPSK modulation. 1—16QAM modulation. 2—64QAM modulation. 3—Reserved.
MOD1[1:0]	**Modulation of layer #1**. Valid for MLEQ and TS processing only (ALG=1,2). Indicates the modulation type of layer #1. 0—QPSK modulation. 1—16QAM modulation. 2—64QAM modulation. 3—Reserved.
MOD2[1:0]	**Modulation of layer #2**. Valid for MLEQ and TS processing only (ALG=1,2). Indicates the modulation type of layer #2. 0—QPSK modulation. 1—16QAM modulation. 2—64QAM modulation. 3—Reserved.
MOD3[1:0]	**Modulation of layer #3**. Valid for MLEQ and TS processing only (ALG=1,2). Indicates the modulation type of layer #3. 0—QPSK modulation. 1—16QAM modulation. 2—64QAM modulation. 3—Reserved.

Table 26-270. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x8

Name	Description
ALIGN_VAL0[7:0]	**User defined output alignment value for layer 0**. Valid only when ALIGN0=1. Indicates the required scale of the output results of layer #0 (in case of LNEQ, MLEQ or TS) or of the matrix results (in case of MIV processing). All output samples are aligned to this value. Range: -128 to 127.
ALIGN_VAL1[7:0]	**User defined output alignment value for layer 1**. Valid only when ALIGN1=1. Indicates the required scale of the output results of layer #1 (in case of LNEQ, MLEQ or TS). All output samples are aligned to this value. Range: -128 to 127.
ALIGN_VAL2[7:0]	**User defined output alignment value for layer 2**. Valid only when ALIGN2=1. Indicates the required scale of the output results of layer #2 (in case of LNEQ, MLEQ or TS). All output samples are aligned to this value. Range: -128 to 127.
ALIGN_VAL3[7:0]	**User defined output alignment value for layer 3**. Valid only when ALIGN3=1. Indicates the required scale of the output results of layer #3 (in case of LNEQ, MLEQ or TS). All output samples are aligned to this value. Range: -128 to 127.

Table 26-271. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0xC

Name	Description
W_ROW[7:0]	**Weight shared Rows**. Valid only when internal interpolation modes are enabled (INTRP=0,1). Indicates the number of rows in the sub-carrier grid that share the same interpolation weights. For details see **Section 26.4.3.5.4.7**, *W Samples*. 0—All the rows in the sub-carrier of the current job share the same interpolation weights. 1 to 255—Number of rows after which the weights changes. (*ROWS* + 1) must be an integer multiplication of *W_ROW*
S_TYPE	**S Matrix Type**. Valid for LNEQ processing only. Indicates the type of the input S matrix. 0—Real, diagonal matrix 1—Hermitian matrix
S_COL[3:0]	**S Shared Columns**. Valid for LNEQ, MLEQ and TS processing only. Indicates the number of columns in the sub-carrier grid which share the same S matrix. For details see **Section 26.4.3.5.4.5**, *S Matrix*. 0—All the columns in the sub carrier grid share the same S matrix. 1—Switch the S matrix every column. 2—Switch the S matrix every 2 columns. ... 8—Switch the S matrix every 8 columns. 9 to 15—Reserved.
S_ROW[7:0]	**S Shared Rows**. Valid for LNEQ, MLEQ and TS processing only. Indicates the number of rows in the sub-carrier grid which share the same S matrix. For details see **Section 26.4.3.5.4.5**, *S Matrix*. 0—All the rows in the sub-carrier grid share the same S matrix. 1—Switch the S matrix every row. 2—Switch the S matrix every 2 rows. ... 255—Switch the S matrix every 255 rows.

Table 26-272. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x10

Name	Description
W_SIZE[15:0]	**Weight Size**. Used by the MAPLE-B2 to calculate the total size (in bytes) of the weights. For details see **Section 26.4.3.5.4.7**, *W Samples*.
S_SIZE[15:0]	**S matrix Size**. Used by the MAPLE-B2 to calculate the total size of the S matrices (in bytes) to be fetched into the EQPE. For details see **Section 26.4.3.5.4.5**, *S Matrix*. Valid range: 0x1 to 0x4000

Table 26-273. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x14

Name	Description
TASK_ID[15:0]	**Task ID**. Targeted for user tracing and identifying of jobs. Used by MAPLE-B2 if a serial RapidIO doorbell interrupt is required. For details, refer to **Section 26.4.4, *External Masters Support Using Serial RapidIO Doorbell***
RM_SCL_TYPE	**R/M Matrix Scale Type**. Valid for TS and MIV modes only. Indicates the scale type of the input R matrix (for TS mode) or input matrix (for MIV mode). For details see **Section 26.4.3.5.4.3**, *R Samples* and **Section 26.4.3.5.4.4**, *M Samples*. 0—Single shared scale value for all matrices in the job. 1—Different scale value for each input matrix.
S_SCL_TYPE	**S matrix Scale Type**. Valid for LNEQ, MLEQ and TS modes only (ALG=0,1,2). Indicates the scale type of the S matrix. for more detail see **Section 26.4.3.5.4.5**, *S Matrix*. 0—Single shared scale value for all input S matrices. 1—Different scale value for each input S matrix.
C_SCL	**C matrix Scale value**. Valid only for LNEQ processing (ALG=0) and only if C_EN=1. Describe the scale value of the input C matrices. Range: -128 to 127.

Table 26-274. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x18

Name	Description
RR0[15:0]	**Rank Reduction of Layer #0 columns**. Valid for LNEQ processing (ALG=0) only. Each bit indicates wether to perform rank reduction to the matching column of layer #0. For details see **Section 26.4.3.5.6.3.2**, *Rank Reduction*. RR0[x]=0: Do not reduce column x of layer #0. RR0[x]=1: Reduce column x of layer #0.
RR1[15:0]	**Rank Reduction of Layer #1 columns**. Valid for LNEQ processing only (ALG=0) and if Lx >= 1. Each bit indicates wether to perform rank reduction to the matching column of layer #1. For details see **Section 26.4.3.5.6.3.2**, *Rank Reduction*. RR0[x]=0: Do not reduce column x of layer #1. RR0[x]=1: Reduce column x of layer #1.

Table 26-275. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x1C

Name	Description
RR2[15:0]	**Rank Reduction of Layer #2 columns**. Valid for LNEQ processing only (ALG=0) and if Lx >= 2. Each bit indicates wether to perform rank reduction to the matching column of layer #2. For details see **Section 26.4.3.5.6.3.2**, *Rank Reduction*. RR0[x]=0: Do not reduce column x of layer #2. RR0[x]=1: Reduce column x of layer #2.
RR3[15:0]	**Rank Reduction of Layer #3 columns**. Valid for LNEQ processing only (ALG=0) and if Lx = 3. Each bit indicates wether to perform rank reduction to the matching column of layer #3. For details see **Section 26.4.3.5.6.3.2**, *Rank Reduction*. RR0[x]=0: Do not reduce column x of layer #3. RR0[x]=1: Reduce column x of layer #3.

Table 26-276. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x20

Name	Description
Y_POINTER[31:4]	**Y pointer**. Valid for LNEQ, MLEQ and TS modes only (ALG=0,1,2). A pointer to the base address in the system memory where the MAPLE-B2 expects to find the Y inputs of the EQPE. For details see **Section 26.4.3.5.4.1.1**, *Y Samples Input Data Structure*. The pointer must be 16 bytes aligned (bits [3:0] are reserved).

Table 26-277. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x24

Name	Description
Y_OFFSET[11:0]	**Y Offset**. Valid for LNEQ, MLEQ and TS modes only (ALG=0,1,2). Indicates the offset from the Y pointer base address where the Y inputs for the current job are to be fetched by MAPLE-B2. For details see **Section 26.4.3.5.4.1.1**, *Y Samples Input Data Structure*.
Y_GAP[11:0]	**Y Gap**. Valid for LNEQ, MLEQ and TS modes only (ALG=0,1,2). Indicates the gap between Y vectors. Used by MAPLE-B2 to calculate the Y inputs location. For details see **Section 26.4.3.5.4.1.1**, *Y Samples Input Data Structure*.

Table 26-278. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x28

Name	Description
H_POINTER0[31:4] or R_POINTER[31:4] or M_POINTER[31:4]	**For LNEQ or MLEQ modes**: **H Pointer #0**. A pointer to the base address in the system memory where the MAPLE-B2 expects to find the first reference input data. In case of external interpolation or non interpolation (INTRP=2,3) modes, this pointer points to the only reference input data buffer. The pointer must be 16 bytes aligned (bits [3:0] are reserved). For details see **Section 26.4.3.5.4.2**, *H Samples*. **For TS mode:** **R Pointer**. A pointer to the base address in the system memory where the MAPLE-B2 expects to find the R matrices input data buffer. The pointer must be 16 bytes aligned (bits [3:0] are reserved). For details see **Section 26.4.3.5.4.3**, *R Samples*. **For MIV mode:** **M Pointer**. A pointer to the base address in the system memory where the MAPLE-B2 expects to find the input matrices data buffer. The pointer must be 16 bytes aligned (bits [3:0] are reserved). For details see **Section 26.4.3.5.4.4**, *M Samples*.

Table 26-279. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x2C

Name	Description
H_POINTER1[31:4]	**H Pointer #1**. Valid only for Internal interpolation mode (INTRP=0,1). A pointer to the base address in the system memory where the MAPLE-B2 expects to find the second reference input data. The pointer must be 16 bytes aligned (bits [3:0] are reserved). For details see **Section 26.4.3.5.4.2**, *H Samples*.

Table 26-280. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x30

Name	Description
H_POINTER2[31:4]	**H Pointer #2**. Valid only for Internal interpolation mode with 4 reference (INTRP=1). A pointer to the base address in the system memory where the MAPLE-B2 expects to find the third reference input data. The pointer must be 16 bytes aligned (bits [3:0] are reserved). For details see **Section 26.4.3.5.4.2**, *H Samples*.

Table 26-281. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x34

Name	Description
H_POINTER3[31:4]	**H Pointer #3**. Valid only for Internal interpolation mode with 4 references (INTRP=1). A pointer to the base address in the system memory where the MAPLE-B2 expects to find the forth reference input data. The pointer must be 16 bytes aligned (bits [3:0] are reserved). For details see **Section 26.4.3.5.4.2**, *H Samples*.

Table 26-282. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x38

Name	Description
S_POINTER[31:1]	**S Pointer**. Valid only for LNEQ, MLEQ and TS modes (ALG=0,1,2). A pointer to the base address in the system memory where the MAPLE-B2 expects to find the S matrix input data. The pointer must be 2 bytes aligned (bit [0] is reserved). For details see **Section 26.4.3.5.4.5**, *S Matrix*.

Table 26-283. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x3C

Name	Description
W_POINTER[31:4]	**W Pointer**. Valid only for LNEQ and MLEQ modes (ALG=0,1) and only when internal interpolation is used (INTRP=0,1). A pointer to the base address in the system memory where the MAPLE-B2 expects to find the interpolation weights input data buffer. The pointer must be 16 bytes aligned (bits [3:0] are reserved). For details see **Section 26.4.3.5.4.7**, *W Samples*.

Table 26-284. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x40

Name	Description
SCL_PTR_BA[31:4]	**Scale Pointers Base Address**. A pointer to a base address in the system memory which is used for calculating the actual scale input buffers for the current job. The pointer must be 16 bytes aligned (bits [3:0] are reserved). For details see **Section 26.4.3.5.4**, *EQPE Input Samples and Data Structures*.

Table 26-285. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x44

Name	Description
S_SCL_OFFSET[7:0]	**S matrix Scale Offset**. An offset address relative to the SCL_PTR_BA pointer where the S matrices scale values are expected. The MAPLE-B2 expect to find the S matrices scale values at address: SCL_PTR_BA + (16 x S_SCL_OFFSET). For details see **Section 26.4.3.5.4.5**, *S Matrix*.
H_SCL_OFFSET[7:0]	**H reference matrix Scale Offset**. An offset address relative to the SCL_PTR_BA pointer where the H reference matrices scale values are expected for NLEQ and MLEQ modes or the R matrices scale values are expected for the TS mode. The MAPLE-B2 expect to find the H reference matrices (or R matrices) scale values at address: SCL_PTR_BA + (16 x H_SCL_OFFSET). For details see **Section 26.4.3.5.4.2**, *H Samples*.
Y_SCL_OFFSET[7:0]	**Y samples Scale Offset**. An offset address relative to the SCL_PTR_BA pointer where the Y matrices scale values are expected. The MAPLE-B2 expect to find the Y matrices scale values at address: SCL_PTR_BA + (16 x Y_SCL_OFFSET). For details see **Section 26.4.3.5.4.1.1**, *Y Samples Input Data Structure*.
F_SCL_OFFSET[7:0]	**F samples Scale Offset**. An offset address relative to the SCL_PTR_BA pointer where the F matrices scale values are expected. The MAPLE-B2 expect to find the F matrices scale values at address: SCL_PTR_BA + (16 x F_SCL_OFFSET). For details see **Section 26.4.3.5.4.8**, *F Samples*.

Table 26-286. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x48

Name	Description
OUT_POINTER0[31:4]	**For LNEQ, MLEQ and TS modes:** **Output Pointer for output samples of layer #0**. A pointer to the base address in the system memory where the MAPLE-B2 is to write the output samples (mantissa) of layer #0. The pointer must be 16 bytes aligned (bits [3:0] are reserved). For details see **Section 26.4.3.5.8.1**, *EQPE Outputs for LNEQ* and **Section 26.4.3.5.8.2**, *EQPE Outputs for MLEQ and TS modes*. **For MIV mode:** **Output pointer.** A pointer to the base address in the system memory where the MAPLE-B2 is to write the output samples (mantissa) of MIV processing. The pointer must be 16 bytes aligned (bits [3:0] are reserved). For details see **Section 26.4.3.5.8.3**, *EQPE Outputs for MIV*.

Table 26-287. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4C

Name	Description
OUT_POINTER1[31:4]	**Output Pointer for output samples of layer #1**. Valid only for LNEQ, MLEQ and TS modes and only if $Lx > 0$. A pointer to the base address in the system memory where the MAPLE-B2 is to write the output samples (mantissa) of layer #1. The pointer must be 16 bytes aligned (bits [3:0] are reserved).

Table 26-288. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x50

Name	Description
OUT_POINTER2[31:4]	**Output Pointer for output samples of layer #2**. Valid only for LNEQ, MLEQ and TS modes and only if $Lx > 1$. A pointer to the base address in the system memory where the MAPLE-B2 is to write the output samples (mantissa) of layer #2. The pointer must be 16 bytes aligned (bits [3:0] are reserved).

Table 26-289. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x54

Name	Description
OUT_POINTER3[31:4]	**Output Pointer for output samples of layer #3**. Valid only for LNEQ, MLEQ and TS modes and only if *Lx* > 2. A pointer to the base address in the system memory where the MAPLE-B2 is to write the output samples (mantissa) of layer #3. The pointer must be 16 bytes aligned (bits [3:0] are reserved).

Table 26-290. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x58

Name	Description
F_POINTER[31:4]	**F Pointer**. Valid only for LNEQ mode (ALG=0) and only when F_EN is set. A pointer to the base address in the system memory where the MAPLE-B2 expects to find the feedback input data buffer to be cancelled by the EQPE. The pointer must be 16 bytes aligned (bits [3:0] are reserved). For details see **Section 26.4.3.5.4.8**, *F Samples*.

Table 26-291. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x5C

Name	Description
C_POINTER[31:4]	**C Pointer**. Valid only for LNEQ mode (ALG=0) and only when C_EN is set. A pointer to the base address in the system memory where the MAPLE-B2 expects to find the C matrices input data buffer. The pointer must be 16 bytes aligned (bits [3:0] are reserved). For details see **Section 26.4.3.5.4.6**, *C Samples*.

Table 26-292. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x60

Name	Description
OUT_SCL_OFF0[7:0]	**For LNEQ, MLEQ and TS modes:** **Output Scaling Offset for output scales of layer #0**. An offset address in the system memory relative to the OUT_POINTER0 field where the MAPLE-B2 is t output the scales vector of layer #0. The address of the scales vector is calculates as follows: Address = (OUT_POINTER0 x 16Bytes) + (OUT_SCL_OFF0 x 256Bytes). If the value of this field is zeroed then the MAPLE-B2 places the scale vector on the next byte aligned address after the completion of the mantissa samples pointed by the OUT_POINTER0. For details see **Section 26.4.3.5.8.1**, *EQPE Outputs for LNEQ* and **Section 26.4.3.5.8.2**, *EQPE Outputs for MLEQ and TS modes*. **For MIV mode:** **Output pointer.** An offset address in the system memory relative to the OUT_POINTER0 field where the MAPLE-B2 is to write the output scales vector of the MIV processing. The address of the scales vector is calculates as follows: Address = (OUT_POINTER0 x 16Bytes) + (OUT_SCL_OFF0 x 256Bytes). If the value of this field is zeroed then the MAPLE-B2 places the scale vector on the next byte aligned address after the completion of the mantissa samples pointed by the OUT_POINTER0. For details see **Section 26.4.3.5.8.3**, *EQPE Outputs for MIV*.
OUT_SCL_OFF1[7:0]	**Output Scaling Offset for output scales of layer #1**.Valid only for LNEQ, MLEQ and TS modes and only if *Lx* > 0. An offset address in the system memory relative to the OUT_POINTER1 field where the MAPLE-B2 is t output the scales vector of layer #1. The address of the scales vector is calculates as follows: Address = (OUT_POINTER1 x 16Bytes) + (OUT_SCL_OFF1 x 256Bytes). If the value of this field is zeroed then the MAPLE-B2 places the scale vector on the next byte aligned address after the completion of the mantissa samples pointed by the OUT_POINTER1.
OUT_SCL_OFF2[7:0]	**Output Scaling Offset for output scales of layer #2**.Valid only for LNEQ, MLEQ and TS modes and only if *Lx* > 1. An offset address in the system memory relative to the OUT_POINTER2 field where the MAPLE-B2 is t output the scales vector of layer #2. The address of the scales vector is calculates as follows: Address = (OUT_POINTER2 x 16Bytes) + (OUT_SCL_OFF2 x 256Bytes). If the value of this field is zeroed then the MAPLE-B2 places the scale vector on the next byte aligned address after the completion of the mantissa samples pointed by the OUT_POINTER2.
OUT_SCL_OFF3[7:0]	**Output Scaling Offset for output scales of layer #3**.Valid only for LNEQ, MLEQ and TS modes and only if *Lx* > 2. An offset address in the system memory relative to the OUT_POINTER3 field where the MAPLE-B2 is t output the scales vector of layer #3. The address of the scales vector is calculates as follows: Address = (OUT_POINTER3 x 16Bytes) + (OUT_SCL_OFF3 x 256Bytes). If the value of this field is zeroed then the MAPLE-B2 places the scale vector on the next byte aligned address after the completion of the mantissa samples pointed by the OUT_POINTER3.

Table 26-293. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x64

Name	Description
SING[14:0]	**Singularity**. Status indication writing by MAPLE-B2 and valid only after OWNER bit negation. Valid only for LNEQ, MLEQ MIV modes (ALG=0,1,3). Indicates the number of R matrices (after QR de-composition) which have one or more diagonal elements (positive real values) with exponent which is smaller than the threshold value set in the EQ_THRESH[SING] register (**Section 26.5.5.4.1**). For details see **Section 26.4.3.5.10**, *EQPE Status Indications*.

Table 26-294. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x68

Name	Description
PST_SCLR_PTR[31:3]	**Post Scalar Pointer**. Valid only if the *PST_SCLR* bit is enabled. A pointer to the system memory where the scalar post multiplication values (L_X x *COLS* different values) in 1q15 representation for the real part and 1q15 representation for the imaginary part are expected. These values are used as scalar values to be multiplied by the iDFT results of each job. For details see **Section 26.4.3.5.9.2**, *Post iDFT Processing*. The only limitation on the scalar value is: ([*PST_SCLR_REAL*]^2 + [*PST_SCLR_IMAG*]^2) < 2

Table 26-295. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x6C

Name	Description
PST_VEC_PTR[31:3]	**Post Vector Multiplication Pointer**. Valid only if *PST_VEC* bit is set. Points to the address in the system memory where the vectors for the iDFT post-multiplier memory are located. The MAPLE-B2 expects *Lx* vectors to be used for all iDFT results related to that layer. The MAPLE-B2 fetches the vectors prior to the iDFT processing. For details see **Section 26.4.3.5.9.2**, *Post iDFT Processing*.

Table 26-296. EQPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x70

Name	Description
H_OFFSET[11:0]	**H Offset**. Valid for LNEQ and MLEQ modes only (ALG=0,1). Indicates the offset from the H pointer base address (of the relevant reference symbol) where the H inputs for the current job are to be fetched by MAPLE-B2. For details see **Section 26.4.3.5.4.2.2**, *H Samples Input Data Structure*.
H_GAP[11:0]	**H Gap**. Valid for LNEQ and MLEQ modes only (ALG=0,1). Indicates the gap between H vectors to be fetched by MAPLE-B2. Used by MAPLE-B2 to calculate the H inputs location. For details see **Section 26.4.3.5.4.2.2**, *H Samples Input Data Structure*.
CONS_H	**Consecutive H**. Valid for LNEQ and MLEQ modes only (ALG=0,1). If reset, the MAPLE-B2 assume that the H vectors are consecutive in the system memory (H_OFFSET=0, H_GAP=(ROWS+1)). For details see **Section 26.4.3.5.4.2.2**, *H Samples Input Data Structure*. 0—The H input vectors are assumed consecutive. H_OFFSET and H_GAP fields are written with the values of 0 and ROWS+1 respectively. 1—The H input vectors offsets are calculated according to H_OFFSET and H_GAP.

The following restrictions apply:

- 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-B2.
- All reserved bits in the BD must be set to zero for future compatibility.
- All the status fields in the BD are updated by the MAPLE-B2 once the job is complete, that is, they are valid only after the owner bit is cleared by the MAPLE-B2.

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 PSIF2 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.

26.5.4.6 CRPE-ULB Core Descriptor, Finger and PCH Commands Structures

The following sections describe the CRPE-ULB data structures used for controlling the CRPE-ULB operation.

26.5.4.6.1 CRPE-ULB Core Descriptors

Cores belonging to group g are organized in a list inside the VCOP parameter memory starting at address *MCUBGxC1P[GCDLBA]*.This address must be within the Group Core Descriptors memory area as described in **Table 26-153**. Each such Core Descriptor structure is described in **Figure 26-257**.

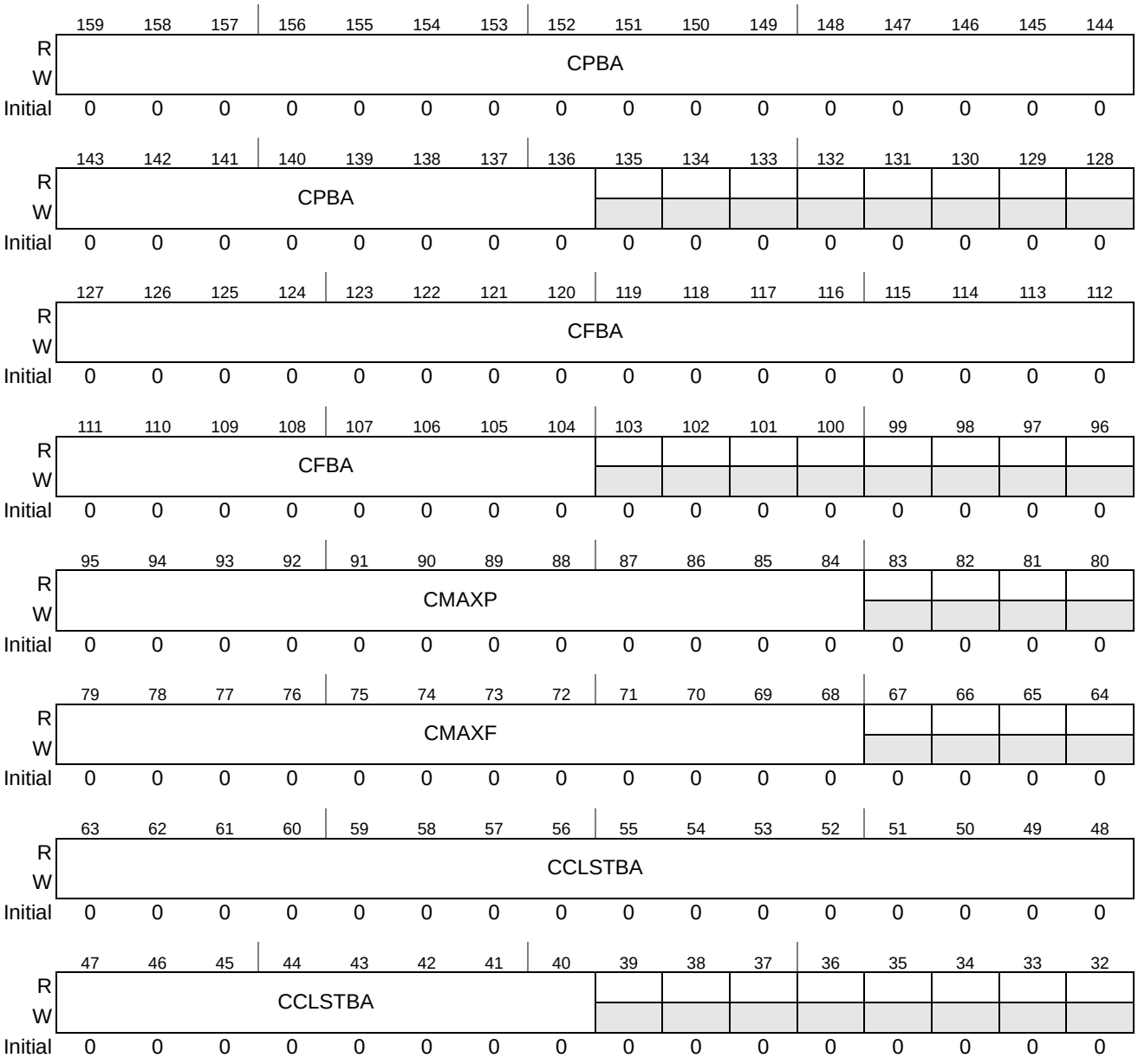


Figure 26-257. CRPE-ULB Core Descriptor Structure

	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								CSWP								
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-257. CRPE-ULB Core Descriptor Structure

Table 26-297. MAPLE CRPE-ULB Core Descriptor Fields Description

Name	Description
CPBA[159:136]	**Core PCH commands Base Address**. Indicates the base address of the first PCH commands list of the first Sub-Slot of the Core. Each list has a reserved allocation of *CMAXP**16 bytes. All the lists are concatenated.
CFBA[127:104]	**Core Fingers command Base Address**. Indicates base address of the first Finger command list of the first Sub-Slot of the Core. Each list has a reserved allocation of *CMAXF**16 bytes. All the lists are concatenated.
CMAXP[95:84]	**Command List Maximum Size of PCH**. Indicates the size in 16 bytes of memory allocated to PCH command list for each Sub-Slot of the Core.
CMAXF[79:68]	**Command List Maximum Size of Fingers**. Indicates the size in 16 bytes of memory allocated to Finger command list for each Sub-Slot of the Core.
CCLSTBA[63:40]	**Core Command List Sizes Table Base Address**. Indicates the Base Address (256 bytes granularity) of the Command List Size table for the Core. For details see **Section 26.4.3.6.1.5.1**, *Core Descriptors and Command List Size Table*.
CSWP[8:0]	**Core Sub Slot Write Pointer**. The Sub-Slot index indicating the index of the next Sub-Slot for which PCH and Finger commands belonging to the core are to be written. Should be maintained by Core (increment when PCH/Fingers command of Sub-slot are ready). Valid Range: 0-299.

26.5.4.6.2 CRPE-ULB Finger Command Structure

Each antenna finger related to a certain PCH of a certain Core should maintain a list of parameters describing its attributes. These attributes are held in a Finger command data structure as described in **Figure 26-258**. It is up to the Core to update the fingers attributes in the finger command which is then being fetched by the MAPLE-B2 into the CRPE-ULB for processing.

Access: User read/write

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Access: User read/write

	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
R				FMO					AID							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
R								PCH_ID								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Access: User read/write

	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
R	GAIN_I															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
R	GAIN_Q															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Access: User read/write

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

Figure 26-258. CRPE-ULB Finger Command Entry

Table 26-298. CRPE-ULB Finger Command Entry Fields Description

Field	Bits	Description
	127-93	Reserved
FMO	92-88	FMO - Finger Memory Offset Valid in Finger Combining bypass mode only (*CRUBMCP[FC_MODE]*=0); Indicates the finger offset relative to the PCH base address in the PCH output buffer. for details see **Section 26.4.3.6.1.14**, *Output Buffer Data Structure*. Valid values 0 to 31.
AID	87-80	Antenna ID Indicates the antenna index in which the finger exists. Valid values: • 0 to 47 if *CRUBMCP[DBL_ANT_EN]* = 0; • 0 to 95 if *CRUBMCP[DBL_ANT_EN]* = 1;

Field	Bits	Description
	79-73	Reserved
PCH_ID	72-64	PCH ID indicates the PCH the finger is associated with. Valid values: 0 to 511
GAIN_I	63-48	Complex Gain I portion Represent the I portion of the complex gain to be multiplied by the symbols extracted from the finger during Finger Combining stage. The number is fixed point number of 1Q15. for details see **Section 26.4.3.6.1.12**, *Finger Combining*. Valid values (-2^{15}) to ($+2^{15} - 1$).
GAIN_Q	47-32	Complex Gain Q portion Represent the Q portion of the complex gain to be multiplied by the symbols extracted from the finger during Finger Combining stage. The number is fixed point number of 1Q15. for details see **Section 26.4.3.6.1.12**, *Finger Combining*. Valid values (-2^{15}) to ($+2^{15} - 1$).
	31-30	Reserved
FOFFSET_H	29-20	Finger Offset High Portion Indicating the finger start offset from the first sample/chip in the input buffer. Valid values: • 0 to 767 if *CRUBMCP[DLY_SPRD]* = 1; • 0 to 511 if *CRUBMCP[DLY_SPRD]* = 0;
FOFFSET_L	19–16	Finger Offset Low Portion Indicating the finger start offset in (up to) 1/16 resolution. For details see **Section 26.4.3.6.1.8.5**, *Internal Interpolation Control* Valid values: • 0 to 15 if *CRUBMCP[DBL_ANT_EN]* = 0; • 0,2,4,6,8,10,12,14 if *CRUBMCP[DBL_ANT_EN]* = 1;
CMD	15–12	Finger Command type. 0 - NULL command 1 - modify finger gain 2 - add finger 3- remove finger 4- reserved 5 - modify finger 6 to 15 - reserved
FID	11–0	Finger ID Contains a unique Finger ID number. Valid values 0 to 3,327

26.5.4.6.3 CRPE-ULB PCH Command Structure

Each PCH of a certain Core should maintain a list of parameters describing its attributes. These attributes are held in a PCH command data structure as described in **Figure 26-259**. It is up to the Core to update the PCH attributes in the PCH command which is then being fetched by the MAPLE-B2 into the CRPE-ULB for processing.

Access: User read/write

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
R				NOF							ODT					CS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96

R				GRP_ID						EXP						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Access: User read/write

	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
R	FCOR_I															
W																
Reset																
	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
R	FCOR_Q															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Access: User read/write

	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
R	OVSF								SCRI							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
R	SCRI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Access: User read/write

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

Figure 26-259. CRPE-ULB PCH Command Entry

Table 26-299. CRPE-ULB PCH Command Entry Fields Description

Field	Bits	Description
—	127–125	Reserved
NOF	124–120	NOF - Number of Fingers Valid only if *CRUB_MCP[FC_MODE]* = 0. Indicate the number of maximum fingers for the PCH; Must be valid and unchanged throughput the entire duration of the PCH activity. Valid values: 0 to 31
—	119–118	Reserved

Table 26-299. CRPE-ULB PCH Command Entry Fields Description

Field	Bits	Description
ODT	117–116	Output Data Type. Selects which portion of processed PCH is to be output from the CRPE-ULB. for details see **Section 26.4.3.6.1.14**, *Output Buffer Data Structure* 0—Real portion 1—Imaginary portion 2—Complex (both portions) 3—Compressed.
—	115–113	Reserved
CS	112	Code Select. Selected between long and short scrambling code 0—Long Code 1—Short Code.
—	111–109	Reserved
GRP_ID	108–104	Group ID Selects to which group the PCH belongs. Valid values: 0 to 23.
—	103–102	Reserved
EXP	101–96	Output Exponent. Valid only if *CRUB_MCP[FC_MODE]* = 1. Indicates the value of the constant exponent (signed) to be used for the Finger Combining results. Values are -32 to +31
FCOR_I	95–80	Frequency Correction I portion Represent the I portion of the frequency correction factor to be multiplied by the finger complex gain in the finger Combining stage. The number is fixed point format of 2Q14. For details see **Section 26.4.3.6.1.12**, *Finger Combining*. Valid values (–128) to (+127)
FCOR_Q	79–64	Frequency Correction Q portion Represent the Q portion of the frequency correction factor to be multiplied by the finger complex gain in the finger Combining stage. The number is fixed point format of 2Q14. For details see **Section 26.4.3.6.1.12**, *Finger Combining*. Valid values (-128) to (+127)
OVSF	63–56	**OVSF Code ID** - Orthogonal Variable Spreading Factor Code ID as defined in the 3GPP technology. For details see **Section 26.4.3.6.1.10**, *Despreading*. Valid values: 0 to 255.
SCRI	55–32	Scrambling Init code value. The initial value of the scrambling code, as its defined in the 3GPP technology. Values are 0 to (2^{24} – 1).
—	31–25	Reserved
PCH_ID	24–16	PCH ID indicates the PCH ID the command is intended for. Valid values of 0 to 511.
CMD	15–12	CMD - PCH Command 0 to 5 - reserved. 6 - PCH command - add / modify PCH 7 - PCH command - remove PCH 8 - PCH command - reset *FCOR_I_VAL* and *FCOR_Q_VAL*. See **Section 26.4.3.6.1.12.2**, *Frequency Correction Factor* for details. 9 to 15 - reserved
	11	Reserved
SF	10–8	Spreading Factor valid values: 0 to 7 representing Spreading Factor of : $2^{(SF+1)}$
CSBO	7–0	Channel SubSlot Offset. Indicates the offset (in units of Sub-Slot), between the PCH scrambling code initial point and its first sub-slot. For details see **Section 26.4.3.6.1.9.1**, *Scrambling Data Offset*. Values are 0 to 149

26.5.4.7 CRCPE Buffer Descriptor Structure

A description of the CRC Buffer Descriptor structure expected by the host for the CRCPE BD rings is outlined in **Figure 26-260**. Some of the fields are status fields which are written by the MAPLE-B2 upon job completion. Other fields may be relevant only for certain configurations and is ignored by the MAPLE-B2 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		INT_EN			MB_PR		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															
W																
Offset	BD_BASE + 0xC															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CRC_RSLT															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CRC_RSLT															
W																

Figure 26-260. CRCPE Buffer Descriptor Structure

Offset	BD_BASE + 0x10															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FAIL								TASKID							
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																
Offset	BD_BASE + 0x14															
	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 + 0x18															
	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 + 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 26-260. CRCPE Buffer Descriptor Structure (Continued)

Table 26-300. CRCPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0

Name	Description
OWNER	**Owner bit**. When set by the host, MAPLE-B2 is the BD owner and the job is not complete. When cleared by MAPLE-B2, the job is done and the host is the BD owner. 0—The host is BD owner 1—MAPLE-B2 is BD owner
WRAP	**Wrap bit**. When set by the host, MAPLE-B2 updates [BDR_RD_PTR] with [BDR_BA] (see **Section 26.5.4.1.4**), that is, it expects to find the next BD at the Base Address of the ring. When cleared by the host, MAPLE-B2 expects to find the next BD at: [BDR_RD_PTR] + BD size.
INT_EN	**Interrupt Enable**. If set, MAPLE-B2 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.
MB_PR[1:0]	**MBus Priority**. Valid only if the [MB_PR_SCH] of the MMC0P parameter (see **Section 26.5.2.1**, *MAPLE Mode Configuration 0 Parameter (MMC0P)*) is set to ‘01’ or ‘10’. Indicated the MBus accesses priority related to this BD. For details, see **Section 26.4.2.4**, *MBus Priority Scheme Configuration* 00— The MBus accesses related to that BD are initiated with priority ‘00’ ... 11—The MBus accesses related to that BD are initiated with priority ‘11’
CHECK	**CRC Check**. If set, MAPLE-B2 performs CRC check on the input buffer. A CRC pass/fail indication is described in the FAIL status bit. If reset, MAPLE-B2 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-B2 performs CRC calculation on the input buffer 1—MAPLE-B2 performs CRC check on the input buffer.
UPDATE	**Update result in system memory**. Valid only if CHECK bit is reset. If set, MAPLE-B2 copies the CRC calculation result described in CRC_RSLT status field into the system memory at the end of the input buffer. 0—MAPLE-B2 does not update the input buffer with the CRC result. 1—MAPLE-B2 copies the CRC result described in the CRC_RSLT field into the system memory at the and of the input buffer
RVRS_IN	**Reverse Input CRC**. If reset, MAPLE-B2 performs byte reverse CRC calculation/check. For details, refer to **Section 26.4.3.7.3**, *Byte Reverse CRC Processing*. 0—MAPLE-B2 performs byte reverse CRC calculation/check. 1—MAPLE-B2 performs straight forward CRC calculation/check.

Table 26-300. CRCPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0

Name	Description
RVRS_OUT	**Reverse Output**. If set, MAPLE-B2 performs reveres operation on the CRC result. For details, refer to **Section 26.4.3.7.6.1**, *Reverse Output Operation* 0—MAPLE-B2 performs reverse on the CRC result. 1—MAPLE-B2 outputs the CRC result as is.
INV_OUT	**Inverse Output**. If set, MAPLE-B2 performs inverse operation on the CRC result. For details, refer to **Section 26.4.3.7.6.2**, *Inverse Output Operation* 0—MAPLE-B2 does not perform inverse operation on the CRC result. 1—MAPLE-B2 performs inverse operation on the CRC result.
CRC_POLY[2:0]	**CRC Polynomial**. Describes which Polynomial to use for the CRC calculation/check. 000— CRC24 with the following Polynomial: $D^{24} + D^{23} + D^6 + D^5 + D + 1$ 001— CRC24 with the following Polynomial: $D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1$ 010— CRC16-CCITT with the following polynomial: $D^{16} + D^{12} + D^5 + 1$ 011— CRC16 with the following polynomial: $D^{16} + D^{15} + D^2 + 1$ 100— CRC32 with the following polynomial: $D^{32} + D^{26} + D^{23} + D^{22} + D^{16} + D^{12} + D^{11} + D^{10} + D^8 + D^7 + D^5 + D^4 + D^2 + D + 1$ 101— CRC18 with the following polynomial: $D^{18} + D^{17} + D^{14} + D^{13} + D^{11} + D^{10} + D^8 + D^7 + D^6 + D^3 + D^2 + 1$ 110— CRC12 with the following polynomial: $D^{12} + D^{11} + D^{10} + D^8 + D^5 + D^4 + 1$ 111— CRC6 with the following polynomial: $D^6 + D^5 + D^3 + D^2 + D^1 + 1$
CRC_BS[15:0]	**CRC Block Size**. Describes the input buffer size (in bytes). 2^1 to 0xFFFF possible values (2 bytes to 64Kbytes)

1. For CRC word generation the minimum block size is 2 bytes. For CRC check the minimum block size is 5 bytes (40 bits)

Table 26-301. CRCPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4

Name	Description
CRC_IB[31:0]	**CRC Input Buffer**. This field points to the CRC input buffer location in system memory, where MAPLE-B2 is to read to perform CRC check/calculation.

Table 26-302. CRCPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x8

Name	Description
CRC_INIT[31:0]	**CRC Initialization value**. MAPLE-B2 uses this field as the CRC initialization value for the CRC processing. In case the CRC polynomial is of size of 6 bits (CRC_POLY =(111)), then the valid bits for this filed are [5:0] only. In case the CRC polynomial is of size of 12 bits (CRC_POLY =(110)), then the valid bits for this filed are [11:0] only. In case the CRC polynomial is of size of 16 bits (CRC_POLY =(010) or (011)), then the valid bits for this filed are [15:0] only. In case the CRC polynomial is of size of 18 bits (CRC_POLY =(101)), then the valid bits for this filed are [17:0] only. In case the CRC polynomial is of size of 24 bits (CRC_POLY =(000) or (001)), then the valid bits for this filed are [23:0] only. In case the CRC polynomial is of size of 32 bits (CRC_POLY =(100), then the valid bits for this filed are [31:0]. For details, refer to section **Section 26.4.3.7.4**, *CRC Initial Value*

Table 26-303. CRCPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0xC

Name	Description
CRC_RSLT[31:0]	**CRC Result**. Valid only if [CHECK] bit is reset and after [OWNER] bit is cleared by MAPLE-B2. Describes the result of the CRC calculation performed on the input buffer.

Table 26-304. CRCPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x10

Name	Description
FAIL	**CRC Fail**. Valid only if [CHECK] bit is set and after [OWNER] bit is cleared by MAPLE-B2. Status bit which indicates whether the CRC check has passed or failed. 0—CRC check has passed. 1—CRC check has failed.
TASKID[7:0]	**Task ID tag**. Supplied by host to track BDs. Also used by MAPLE-B2 if a serial RapidIO doorbell interrupt is required. For details refer to **Section 26.4.4, *External Masters Support Using Serial RapidIO Doorbell***

Table 26-305. CRCPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x14

Name	Description
[31:0]	**Reserved**.

Table 26-306. CRCPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x18

Name	Description
[31:0]	**Reserved**.

Table 26-307. CRCPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x1C

Name	Description
[31:0]	**Reserved**.

The following restrictions apply to the CRCPE BDs:

- 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-B2.
- All reserved bits in the BD must be cleared to zero for future compatibility.
- All the status fields in the BD are updated by the MAPLE-B2 once the job is complete, that is, they are valid only after the owner bit is cleared by the MAPLE-B2.

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 PSIF2 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.

26.5.4.8 CGPE Buffer Descriptor Structure

A description of the Code Generator Buffer Descriptor structure expected by the host for the CGPE BD rings is outlined in **Figure 26-261**. Some of the fields are status fields which are written by the MAPLE-B2 upon job completion. Other fields may be relevant only for certain configurations and is ignored by the MAPLE-B2 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		INT_EN			MB_PR		CODE_SIZE							
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																
Offset	BD_BASE + 0x4															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TASK_ID								PN_NUM							
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PN_NUM															
W																
Offset	BD_BASE + 0x8															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R					SF				OVSF_NUM							
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	START_OFFSET															
W																
Offset	BD_BASE + 0xC															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R													CONJ_EN	LONG_MODE	UL_CODE	OUT_MODE
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CFG_VAL_1								CFG_VAL_0							
W																
Offset	BD_BASE + 0x10															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CODE_RSLT_PTR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CODE_RSLT_PTR															
W																
Offset	BD_BASE + 0x14															

Figure 26-261. CGPE Buffer Descriptor Structure

	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 + 0x18															
	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 + 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 26-261. CGPE Buffer Descriptor Structure (Continued)

Table 26-308. CGPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0

Name	Description
OWNER	**Owner bit**. When set by the host, MAPLE-B2 is the BD owner and the job is not complete. When cleared by MAPLE-B2, the job is done and the host is the BD owner. 0—The host is BD owner 1—MAPLE-B2 is BD owner
WRAP	**Wrap bit**. When set by the host, MAPLE-B2 updates [BDR_RD_PTR] with [BDR_BA] (see **Section 26.5.4.1.4**), that is, it expects to find the next BD at the Base Address of the ring. When cleared by the host, MAPLE-B2 expects to find the next BD at: [BDR_RD_PTR] + BD size.
INT_EN	**Interrupt Enable**. If set, MAPLE-B2 issues an 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.
MB_PR[1:0]	**MBus Priority**. Valid only if the [MB_PR_SCH] of the MMC0P parameter (see **Section 26.5.2.1**, *MAPLE Mode Configuration 0 Parameter (MMC0P)*) is set to ‘01’ or ‘10’. Indicated the MBus accesses priority related to this BD. For details, see **Section 26.4.2.4**, *MBus Priority Scheme Configuration* 00— The MBus accesses related to that BD are initiated with priority ‘00’ ... 11—The MBus accesses related to that BD are initiated with priority ‘11’
CODE_SIZE[7:0]	**Code Size**. Indicated the size of the required generated code. The size value is of 128 bits multiplication: 0—Generated Code size is 128 bits. 1—Generated Code size is 256 bits. 2—Generated Code size is 384 bits. ... 255—Generated Code size is 32768 bits.

Table 26-309. CGPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4

Name	Description
TASKID[7:0]	**Task ID tag**. Supplied by host to track BDs. Also used by MAPLE-B2 if a serial RapidIO doorbell interrupt is required. For details, refer to **Section 26.4.4, *External Masters Support Using Serial RapidIO Doorbell***
PN_NUM[23:0]	**Pseudo random Number**. Uplink Scrambling (UL_CODE=1): The initial state of X sequence (bits [0:13]) used for uplink long Code generation. Downlink Scrambling (UL_CODE=0): Scrambling Code number (bits [0:14]) used for Downlink scrambling code generation. Valid range: 0x0000 to 0x5FFF

Table 26-310. CGPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x8

Name	Description
SF[3:0]	**Spreading Factor**. Required for hadamard code generation. The actual spreading factor is 2^{SF}: 0—Spreading Factor is 1. 1—Spreading Factor is 2. 2—Spreading Factor is 4. ... 8—Spreading Factor is 256. 9 to 15—Reserved.
OVSF_NUM[7:0]	**OVSF Number**. Orthogonal Variable Spreading Factor code number corresponding to the selected Spreading Factor (SF). Valid Range: $0 \leq OVSF\_NUM \leq (2^{SF} - 1)$.
START_OFFSET[15:8]	**CG Start Offset**. Indicates the start point for codes generation corresponding to specific chip number in the frame. Required for the scrambling code generation. The START_OFFSET value represent Sub-Slot alignment, that is, the value of 0x1 represent chip offset of 256. Valid range: 0 to 149.

Table 26-311. CGPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0xC

Name	Description
CONJ_EN	**Conjugate Enable**. Indicates whether the resulting code is conjugated. 0—The Code results (I,Q) are not conjugated. 1—The Code results (I,Q) are conjugated.
LONG_MODE	**Long Mode**. Valid only during Uplink mode (UL_CODE = 1). Indicates whether long or short code is generated. 0—Generate Uplink Scrambling Short code. 1—Generate Uplink Scrambling Long code.
UL_CODE	**Uplink Code**. Indicates if Uplink or Downlink codes are generated. 0—Generate Downlink codes 1—generate Uplink codes.
OUT_MODE	**Output Mode**. Indicates the code output format. 0—The generated code format is pairs of 8 bits I and 8 bits Q, that is, 8 I,Q pairs for each 128 bits of generated code. 1— The generated code format is pairs of single bit I and single bit Q, that is, 64 I,Q pairs for each 128 bits of generated code.
CFG_VAL_1[7:0]	**Configuration Value of 1**. Valid only if output mode is 8 bits representation (OUT_MODE = 1). Indicates that the value represent is a generated code bit with the value of 1.
CFG_VAL_0[7:0]	**Configuration Value of 0**. Valid only if output mode is 8 bits representation (OUT_MODE = 1). Indicates that the value represented is a generated code bit with the value of 0.

Table 26-312. CGPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x10

Name	Description
CODE_RSLT_PTR[31:0]	**Code Result Pointer**. Pointer to the system memory where the MAPLE-B2 is to write the generated code.

Table 26-313. CGPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x14

Name	Description
[31:0]	**Reserved**.

Table 26-314. CGPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x18

Name	Description
[31:0]	**Reserved**.

Table 26-315. CGPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x1C

Name	Description
[31:0]	**Reserved**.

The following restrictions apply to the CGPE BDs:

- 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-B2.
- All reserved bits in the BD must be cleared to zero for future compatibility.

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 PSIF2 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.

26.5.4.9 CONVPE Buffer Descriptor Structure

The following is a description of the CONVPE Buffer Descriptors structure as they should be written to the CONVPE BD rings by the host.

Offset BD_BASE + 0x0

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

Offset BD_BASE + 0x4

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

Figure 26-262. CONVPE Buffer Descriptor Structure

Table 26-316. CONVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0

Name	Description
OWNER	**Owner bit**. When set by the host, MAPLE-B2 is the BD owner and the job is not complete. When cleared by MAPLE-B2, the job is done and the host is the BD owner. 0—The host is BD owner 1—MAPLE-B2 is BD owner
WRAP	**Wrap bit**. When set by the host, MAPLE-B2 updates [BDR_RD_PTR] with [BDR_BA] (see **Section 26.5.4.1.4**, *MAPLE <yy>PE BD Ring High Priority A <x> Parameter (M<yy>BRHPAxP)*), that is, it expects to find the next BD at the Base Address of the ring. When cleared by the host, MAPLE-B2 expects to find the next BD at: [BDR_RD_PTR] + BD size.
INT_EN	**Interrupt Enable**. If set, MAPLE-B2 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.
TASK_ID	Task ID. Describes the task number, set by host and used by the host.
TASK_TYPE [2:0]	Task Type. Describes the type of the task to be executed. 000 - Reserved 001 - UMTS RACH correlation 010 - UMTS Path Searcher correlation 011 - Reserved 100 - Reserved 101 - Reserved 110 - Reserved 111 - Reserved
TASK_DESCR_SIZE[11:0]	**Task Descriptor Size.** Describes the detailed task descriptor size in bytes.

Table 26-317. CONVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4

Name	Description
TASK_ADDR [31:3]	Task Descriptor Address. Address for detailed task descriptor in system memory. The *TASK_ADDR* field must be aligned to 8 bytes.

26.5.4.10 CONVPE RACH Preamble Correlations Task Descriptor

The following is a description of the CONVPE UMTS RACH correlations task fields as they should be written by the host in system memory pointed by the CONVPE buffer descriptor *TASK_ADDR* field.

Offset TD_BASE + 0x0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R					ANT_DIV_BCH				RSLT_OVA_SCL	ANT_FFT_WBE	ANT_FD	CCW				
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SIGN_INDX															
W																
Offset TD_BASE + 0x4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSLT_ADP_OVA_SCL								AD_UPDATE_EN		TL_ID					
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AD_UPDATE_NUM				OVRLP_SZ											
W																
Offset TD_BASE + 0x8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ANT_DATA_FD_ADDR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ANT_DATA_FD_ADDR															
W																
Offset TD_BASE + 0xC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ANT_DATA_SCALE_ADDR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ANT_DATA_SCALE_ADDR															
W																

Figure 26-263. CONVPE RACH Task Descriptor Structure

Offset	TD_BASE + 0x10															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	AD_OFFSET															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AD_OFFSET															
Offset	TD_BASE + 0x14															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	AD_BUFFER_SIZE															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AD_BUFFER_SIZE															
Offset	TD_BASE + 0x18															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SIGN_ADDR															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SIGN_ADDR															
Offset	TD_BASE + 0x1C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SING_OFFSET															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CORR_RSLT_OFFSET															
Offset	TD_BASE + 0x20															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CORR_RSLT_ADDR															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CORR_RSLT_ADDR															
Offset	TD_BASE + 0x24															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CORR_RSLT_SCALE_ADDR															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CORR_RSLT_SCALE_ADDR															

Figure 26-263. CONVPE RACH Task Descriptor Structure (Continued)

Offset	TD_BASE + 0x28+X*0x4 (X is antenna index)															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ADx_BASE															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADx_BASE															
W																

Figure 26-263. CONVPE RACH Task Descriptor Structure (Continued)

Table 26-318. UMTS RACH Task Descriptor Fields Description

Name	Description
ANT_DIV_BCH[3:0]	Antenna Diversity Branches. Describes number of data streams for correlation. 0,1—Reserved. 2— Two data streams. 3— Reserved. 4— Four data streams. 5— Reserved. 6— Six data streams. 7— Reserved. 8— Eight data streams. 9 to 15— Reserved.
RSLT_OVA_SCL	**Results Overall Scaling**. If set, the total scaling of the correlation results is aligned with the *ADP_OVA_SCL* field in the Task Descriptor (TD). 0—No overall scaling is specified. 1—The CONVPE aligns the total accumulated scaling with the scaling value specified in the *ADP_OVA_SCL* filed in the TD.
ANT_FFT_WBE	**Antenna Data FFT Write Back Enable.** Relevant only if *ANT_FD* =0. Indicates if frequency domain chips need to be written to system memory. 0—Frequency domain results are **not** written to system memory. 1—Frequency domain results are written to system memory as pointed in the *ANT_DATA_FD_ADDR* field.
ANT_FD	**Antenna Data in Frequency Domain.** Indicates if antenna data is stored in system memory in frequency domain (already passed FFT processing) or in time domain 0—Time domain antenna data, represented as 8I, 8Q per chip, fixed point time domain chips vector 1—Frequency domain antenna data, represented as 16I, 16Q, block floating frequency domain chips vector
CCW[4:0]	**Coherent Correlation Window**. Describes the Coherent Correlation Window (CCW) size in multiples of FFT transforms needed to perform the task, including the overlap. FFT size is defined in by *TL_ID* field. For details see **Figure 26-172**. 0x0—Reserved 0x1—CCW of single FFT transform length ..... 0x18—CCW of 24 FFT transform lengths 0x19—0x1F - Reserved
SIGN_INDX [15:0]	**Signatures Index.** Describes number and index of RACH signatures to correlate with the data streams Every bit enables the relevant preamble signature as described in *Table 3 of 3GPP specification 25.213 - Preamble Signatures*. The signatures are zero padded according to the *OVRLP_SIZE*, conjugated, flipped and stored in frequency domain as part of pre-processing done before the CONVPE RACH correlation task. bit 0—represents $P_0(n)$ .... bit 15—represents $P_{15}(n)$

Table 26-318. UMTS RACH Task Descriptor Fields Description

Name	Description
RSLT_ADP_OVA_SCL [7:0]	**Results Adaptive Overall Scaling**. Valid only if the *RSLT_OVA_SCL* field in the TD is set. Determines the Overall scale the CONVPE is to perform on the correlation results. 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. The field apply to all the results of current correlation task. Supported values: –128 to 127.
AD_UPDATE_EN	Update the Antenna Data buffer. 0—Write CCW amount of antenna data, starting at offset 0 of the antenna buffer. 1—Write AD_UPDATE_NUM amount of antenna data, starting at the last offset used of the antenna buffer.
TL_ID[5:0]	**Transform Length ID**. This field determines the executed transform length. See **Table 26-246** for DFT/FFT Transform Length encoding. Valid values for RACH correlation operation are: 35, 36, 37, 38, 39 representing 128, 256, 512, 1024 and 2048 points transforms respectively.
AD_UPDATE_NUM[4:0]	**Number of FFT transforms to write to the antenna buffer.** Valid only if AD_UPDATE_EN is set. 0x0— Write to the antenna buffer is disabled. 0x1— Write a single FFT transform length. ..... 0x18—Write 24 FFT transform lengths. 0x19—0x1F - Reserved
OVRLP_SZ [10:0]	**Overlap Size.** Describes number of chips to overlap for data streams processing. Represents the delay spread in chips for correlation in frequency domain. 0x000 - No overlap 0x7FF - 2047 chips overlap
ANT_DATA_FD_ADDR [31:0]	**Antenna Data in Frequency Domain Address.** Applicable only if *ANT_FFT_WBE* bit is set. This is the pointer to a data structure that contains the antenna datApost FFT for further use by other tasks. The data structure is organized as shown in **Figure 26-176**
AD_FD_SCALE_ADDR[31:0]	**Antenna Data Scale Values Pointer.** Points to scale values (exponents) for block floating point representation of the antenna data. Each exponent represented by 8 bit value. The scale values are concatenated one after another in same order as antenna data vectors. Number of values is equivalent to *ANT_DIV_BCH * CCW* • If *ANT_FD* =1, the MAPLE-B2 fetches the concatenated scale values according to this pointer. • If *ANT_FD* = 0 & *ANT_FFT_WBE* = 1, the MAPLE-B2 uses this pointer to write the concatenated scale values of the FFT processing results.
AD_OFFSET [31:0]	**Antenna Data Offset**. Describes the offset of the antenna data belonging to antenna diversity to be processed in this correlation task. *ADx_BASE* + *AD_OFFSET* defines the start address from which to read the relevant antenna data for the diversity X in this correlation task. For details, see **Figure 26-173**.
AD_BUFFER_SIZE[31:0]	**Antenna Data Buffer Size.** Defines the size (in bytes) of the antenna diversity circular buffer. For details, see **Figure 26-173**.
SIGN_ADDR [31:3]	**Signatures Base Address**. Points to the start address in the system memory of the first out of several signature vectors stored in frequency domain. The start address of the next vectors is distances form this pointer by *SIGN_OFFSET* multiplications. For details, see **Figure 26-174**. Total number of vectors is based on *SIGN_INDX* field. The size of each signature vector depends on FFT size and overlap size. The *SIGN_ADDR* field must be aligned to 8 bytes.
SIGN_OFFSET [15:0]	**Signature Offset**. Describes the relative offset between the signature vectors in the system memory. For details, see **Figure 26-174**.
CORR_RSLT_OFFSET [15:0]	**Correlation Results Offset**. Describes the relative offset between the correlation results vectors in the system memory. For details, see **Figure 26-174**.
CORR_RSLT_ADDR[31:0]	**Correlation Results Address.** Points to the start address of all the correlation results requested in this job. Organized as shown in **Figure 26-174**

Table 26-318. UMTS RACH Task Descriptor Fields Description

Name	Description
CORR_RSLT_SCALE_ADDR [31:0]	**Correlation Results Scale Values Pointer.** Points to scale values (exponents) for block floating point representation of the correlation results. Each exponent represented by 8 bit value. The scale values are concatenated one after another in same order as antenna data vectors.
ADx_BASE [31:3]	Antenna Data X Base Pointer. Points to the start address of circular buffer in the system memory with the antenna diversity X, representing one of the diversity branches. For details, see **Figure 26-173**. Total number of pointers is defined by *ANT_DIV_BCH* The *ADx_BASE* field must be aligned to 8 bytes.

26.5.4.11 CONVPE Path Searcher Task Descriptor

The following is a description of the CONVPE UMTS Path searcher correlations task fields as they should be written by the host in system memory pointed by the CONVPE buffer descriptor *TASK_ADDR* field.

Offset TD_BASE + 0x0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R					ANT_DIV_BCH				RSLT_OVA_SCL	ANT_FFT_WBE	ANT_FD	CCW				
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									NUSERS							
W																
Offset TD_BASE + 0x4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									AD_UPDATE_EN		TL_ID					
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AD_UPDATE_NUM					OVRLP_SZ										
W																
Offset TD_BASE + 0x8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ANT_DATA_FD_ADDR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ANT_DATA_FD_ADDR															
W																
Offset TD_BASE + 0xC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	AD_FD_SCALE_ADDR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AD_FD_SCALE_ADDR															
W																

Figure 26-264. CONVPE Path Searcher Task Descriptor Structure

Offset	TD_BASE + 0x10															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	AD_OFFSET															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AD_OFFSET															
Offset	TD_BASE + 0x14															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	AD_BUFFER_SIZE															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AD_BUFFER_SIZE															
Offset	TD_BASE + 0x18															
	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	U_CORR_RSLT_OFFSET															
Offset	TD_BASE + 0x1C+X*0x4 (X is antenna diversity index ; 0 ≤ X ≤ (*ANT_DIV_BCH* - 1)															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ADx_BASE															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADx_BASE															
Offset	TD_BASE + 0x1C+*ANT_DIV_BCH**0x4+Y*0x1C (Y is user index; 0 ≤ Y ≤ *NUSERS)*															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SCRM_SEQ															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SCRM_SEQ									CNPI					FC_EN	LS
Offset	TD_BASE + 0x20+*ANT_DIV_BCH**0x4+Y*0x1C (Y is user index; 0 ≤ Y ≤ *NUSERS)*															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W					SOIF				RSLT_ADP_OVA_SCL							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COIF															

Figure 26-264. CONVPE Path Searcher Task Descriptor Structure (Continued)

Offset	TD_BASE + 0x24+*ANT_DIV_BCH**0x4+Y*0x1C (Y is user index; 0 ≤ Y ≤ *NUSERS)*															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FCB_REAL															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FCB_IMAG															
Offset	TD_BASE + 0x28+*ANT_DIV_BCH**0x4+Y*0x1C (Y is user index; 0 ≤ Y ≤ *NUSERS)*															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FCISC															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FCSS															
Offset	TD_BASE + 0x2C+*ANT_DIV_BCH**0x4+Y*0x1C (Y is user index; 0 ≤ Y ≤ *NUSERS)*															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FCS_REAL															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FCS_IMAG															
Offset	TD_BASE + 0x30+*ANT_DIV_BCH**0x4+Y*0x1C (Y is user index; 0 ≤ Y ≤ *NUSERS)*															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	U_CORR_RSLT_ADDR															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	U_CORR_RSLT_ADDR															
Offset	TD_BASE + 0x34+*ANT_DIV_BCH**0x4+Y*0x1C (Y is user index; 0 ≤ Y ≤ *NUSERS)*															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	U_CORR_RSLT_SCALE_ADDR															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	U_CORR_RSLT_SCALE_ADDR															

Figure 26-264. CONVPE Path Searcher Task Descriptor Structure (Continued)

Table 26-319. UMTS Path Searcher Task Descriptor Fields Description

Name	Description
ANT_DIV_BCH[3:0]	Antenna Diversity Branches. Describes number of data streams for correlation. 0,1—Reserved. 2— Two data streams. 3— Reserved. 4— Four data streams. 5— Reserved. 6— Six data streams. 7— Reserved. 8— Eight data streams. 9 to 15— Reserved.
RSLT_OVA_SCL	**Results Overall Scaling**. If set, the total scaling of the correlation results is aligned with the *ADP_OVA_SCL* field in the TD. 0—No overall scaling is specified. 1—The CONVPE aligns the total accumulated scaling with the scaling value specified in the *ADP_OVA_SCL* filed in the BD.
ANT_FFT_WBE	**Antenna Data FFT Write Back Enable.** Relevant only if *ANT_FD* =0. Indicates if frequency domain chips need to be written to system memory. 0—Frequency domain results are **not** written to system memory. 1—Frequency domain results are written to system memory as pointed in the *ANT_DATA_FD_ADDR* field.
ANT_FD	**Antenna Data in Frequency Domain.** Indicates if antenna data is stored in system memory in frequency domain (already passed FFT processing) or in time domain 0—Time domain antenna data, represented as 8I, 8Q per chip, fixed point time domain chips vector 1—Frequency domain antenna data, represented as 16I, 16Q, block floating frequency domain chips vector
CCW[4:0]	**Coherent Correlation Window**. Describes the Coherent Correlation Window (CCW) size in multiples of FFT transforms needed to perform the task, including the overlap. FFT size is defined in by *TL_ID* field. For details see **Figure 26-172**. 0x0—Reserved 0x1—CCW of single FFT transform length ..... 0x18—CCW of 24 FFT transform lengths 0x19—0x1F - Reserved
NUSERS[7:0]	**Number of Users.** Describes number of users for Path Searcher task in this descriptor. 0x00—1 user 0x01—2 users ..... 0xFF—256 users
AD_UPDATE_EN	Update the Antenna Data buffer. 0—Write CCW amount of antenna data, starting at offset 0 of the antenna buffer. 1—Write AD_UPDATE_NUM amount of antenna data, starting at the last offset used of the antenna buffer.
TL_ID[5:0]	**Transform Length ID**. This field determines the executed transform length. See **Table 26-246** for DFT/FFT Transform Length encoding. Valid values for RACH correlation operation are: 35, 36, 37, 38, 39 representing 128, 256, 512, 1024 and 2048 points transforms respectively.
AD_UPDATE_NUM[4:0]	**Number of FFT transforms to write to the antenna buffer.** Valid only if AD_UPDATE_EN is set. 0x0— Write to the antenna buffer is disabled. 0x1— Write a single FFT transform length. ..... 0x18—Write 24 FFT transform lengths. 0x19—0x1F - Reserved

Table 26-319. UMTS Path Searcher Task Descriptor Fields Description

Name	Description
OVRLP_SZ [10:0]	**Overlap Size.** Describes number of chips to overlap for data streams processing. Represents the delay spread in chips for correlation in frequency domain. 0x000—No overlap 0x7FF—2047 chips overlap
ANT_DATA_ FD_ADDR [31:0]	**Antenna Data in Frequency Domain Address.** Applicable only if *ANT_FFT_WBE* bit is set. This is the pointer to a data structure that contains the antenna datApost FFT for further use by other tasks. The data structure is organized as shown in **Figure 26-176**.
AD_FD_ SCALE_ ADDR[31:0]	**Antenna Data Scale Values Pointer.** Points to scale values (exponents) for block floating point representation of the antenna data. Each exponent represented by 8 bit value. The scale values are concatenated one after another in same order as antenna data vectors. Number of values is equivalent to *ANT_DIV_BCH * CCW* • If *ANT_FD* =1, the MAPLE-B2 fetches the concatenated scale values according to this pointer. • If *ANT_FD* = 0 & *ANT_FFT_WBE* = 1, the MAPLE-B2 uses this pointer to write the concatenated scale values of the FFT processing results.
AD_OFFSET [15:0]	**Antenna Data Offset**. Describes the offset of the antenna data belonging to antenna diversity X to be processed in this correlation task. *ADx_BASE* + *AD_OFFSET* defines the start address from which to read the relevant antenna data for the diversity X in this correlation task. For details, see **Figure 26-173**.
AD_BUFFER_ SIZE[15:0]	**Antenna Data Buffer Size.** Defines the size (in bytes) of the antenna diversity circular buffer. For details, see **Figure 26-173**.
U_CORR_ RSLT_ OFFSET[15:0]	**User Correlation Results Offset.** Describes the relative offset between the correlation results vectors in the system memory. For details, see **Figure 26-175**
ADx_BASE [31:3]	Antenna Data X Base Pointer. Points to the start address of circular buffer in the system memory with the antenna diversity X, representing one of the diversity branches. For details, see **Figure 26-173**. Total number of pointers is defined by *ANT_DIV_BCH* The *AD_BASE* field must be aligned to 8 bytes.
The fields below are per user, and appear in task descriptor according to *NUSERS* field.	
SCRM_SEQ [23:0]	**Scrambling Sequence.** A 24-bit word that specifies user scrambling sequence, as defined in 3GPP specification.
CNPI[2:0]	**Number of Pilots.** Number of pilots per slot in current radio frame. Valid Range: 0 to 5 representing 3 to 8.
FC_EN	**Frequency Correction Enable.** If set indicates that frequency correction should be performed on the pilot sequences generated for this user. 1—Frequency correction enabled 0—Frequency correction disabled, all the fields related to frequency correction are not relevant.
LS	**Long or short.** Scrambling sequence selection 1—Long sequence 0—Short sequence
SOIF[3:0]	**Slot Offset in Frame.** Slot number in frame to start the correlation
RSLT_ADP_ OVA_SCL [7:0]	**Results Adaptive Overall Scaling**. Valid only if the *RSLT_OVA_SCL* field in the BD is set. Determines the Overall scale the CONVPE is to perform on the correlation results. 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. The field apply to all the results of current correlation task. Valid values: -128 to +127.
COIF[15:0]	**Chip Offset in Frame.** Specifies the user chip's offset to start the correlation in current window. Should meet the following condition: *SOIF**2560 ≤ *COIF* < (*SOIF*+1)*2560 The delay is specified in chips with possible values range 0 to 38399 chips

Table 26-319. UMTS Path Searcher Task Descriptor Fields Description

Name	Description
FCB_REAL [15:0]	**Frequency Correction Base Value Real Part**. Valid only if *FC_EN* is enabled. Describes the real part value of the Frequency Correction initialization. Valid values: 1 to 65535
FCB_IMAG [15:0]	**Frequency Correction Base Value Imaginary Part**. Valid only if *FC_EN* is enabled. Describe the imaginary part value of the Frequency Correction initialization. Valid values: 1 to 65535
FCISC[15:0]	**Frequency Correction Input Step Counter**. Valid only if *FC_EN* is enabled. Indicates the step counter initial value for starting the frequency correction argument generation. Its value should be small or equal to the *FCSS* field. If Step Size *FCSS* equals 1, this field should be set to 0x0. For details see **Section 26.4.3.3.7**, *Frequency Correction Support in eFTPE*. Valid values: 0 to 65532 which are products of 4.
FCSS[15:0]	**Frequency Correction Step Size**. Valid only if *FC_EN* is enabled. The number of samples that should be corrected with the same correction factor before the correction factor is increment by the shift value. For details see **Section 26.4.3.3.7**, *Frequency Correction Support in eFTPE* Supported values: 1 or values between 4–65532 which are products of 4.
FCS_REAL [15:0]	**Frequency Correction Shift Value Real Part**. Valid only if *FC_EN* is enabled. Describe the real part value of the Frequency Correction shift value. Supported values: 1 to 65535
FCS_IMAG [15:0]	**Frequency Correction Shift Value Imaginary Part**. Valid only if *FC_EN* is enabled. Describe the imaginary part value of the Frequency Correction shift value. Supported values: 1 to 65535
U_CORR_RSLT_ADDR [31:0]	**User Correlation Results Address.** Points to the start address of all the correlation results requested in this job per user. Organized as shown in **Figure 26-175**
U_CORR_RSLT_SCALE_ADDR[31:0]	User Correlation Results Scale Values Pointer. Points to scale values (exponents) for block floating point representation of the correlation results. Each exponent represented by 8 bit value. The scale values are concatenated one after another in same order as antenna data vectors.

26.5.5 Registers

The MAPLE-B2 includes several sets of programmable registers that configure and monitor the operation of MAPLE-B2 and the individual PEs. The following subsections provide detailed listings of the registers along with detailed descriptions of the register layouts and bit field descriptions

26.5.5.1 PSIF2 Registers (SBus)

The PSIF2 registers can be accessed via the MAPLE-B2s SBus port only. The SBus interface of the MAPLE-B2 is used to access the PSIF2 internal registers only. **Table 26-126** describes the SBus Memory Map. The base address is 0xFFF1D000.

26.5.5.1.1 PSIF Command Register (PCR)

This register is the MAPLE-B2 command register. It is used for initializing a specific MAPLE-B2 routines.

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 26-265. PSIF2 Command Register

Table 26-320. PSIF2 Command Register Fields Description

Field	Description
31–17	Reserved.
16	FLG - Command semaphore flag. Set by the host and cleared by the MAPLE-B2 once execution is done. Once set, the MAPLE-B2 executes the requested routine as described in PCR[OPCODE], and cleared the FLG bit.

Table 26-320. PSIF2 Command Register Fields Description

Field	Description
15–6	Reserved.
5–0	OPCODE - On assertion of the PCR[FLG], the MAPLE-B2 executes the task encoded in this field according to the following encoding: ‘000000’ - Reserved. ‘000001’ - Maple_parse_bd. For details, see **page 26-312**. ‘000010’ - Maple_parse_tvpe_bd. For details, see **page 26-312**. ‘000011’ - Maple_parse_ftpe_0_bd. For details, see **page 26-312**. ‘000100’ - Maple_parse_ftpe_1_bd. For details, see **page 26-312**. ‘000101’ - Maple_parse_ftpe_2_bd. For details, see **page 26-312**. ‘000110’ - Maple_parse_depe_bd. For details, see **page 26-312**. ‘000111’ - Maple_parse_crcpe_bd. For details, see **page 26-312**. ‘001000’ - Maple_parse_eqpe_bd. For details, see **page 26-312**. ‘001001’ - Maple_parse_convpe_bd. For details, see **page 26-312**. ‘001010’ - Maple_parse_cgpe_bd. For details, see **page 26-312**. ‘001011’ - Maple_reset_crpe. For details, see **page 26-312**. ‘001100’ - Maple_crpe_ulb_init. For details see **page 26-313** ‘001101’ - Maple_crpe_ulf_init. For details see **page 26-313** ‘001110’ - Maple_crpe_3x_eftpe_clock_gate_control. For details see **page 26-313** ‘001111’ - Reserved. ‘010000’ - Maple_update_eftpe_pre_post_buffers. For details see **page 26-313** ‘010001’ to ‘111111’ - Reserved.

26.5.5.1.2 PSIF PIC Event Register 0 (PSPICER0)

This register describes which Buffer Descriptors Rings events are set. These events can be cleared only by the host writing this register with the value of 1 in the relevant bit. The bits in this event register are set only if the interrupt configuration is level (see **Section 26.5.5.1.5**, *PSIF PIC Edge/Level Register 0 (PSPICELR0)*

Offset 0x00001600 (PSPICER0) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BDR_DONE[31:16]															
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[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-266. PSIF PIC Event Register 0

Table 26-321. PSIF PIC Event Register 0 Fields Description

Bits	Description
31–0	BDR_DONE[31:0]—Buffer Descriptor Done. 32 interrupt lines which can be asserted due to Buffer Descriptors completion. Each of these interrupts can be assigned to any of the BD rings (see **Section 26.5.4.1.5** or **Section 26.5.4.1.7**), and can be configured as edge or level. Read ‘0’ Buffer Descriptor Ring Done event did not occur Read ‘1’ Buffer Descriptor Ring Done event occur Write ‘1’ Clear bit if set.

26.5.5.1.3 PSIF PIC Event Register 1 (PSPICER1)

This register describes which event caused the assertion of the MAPLEs general error interrupt, which is a level interrupt only. Any assertion of any of these events causes the assertion of the MAPLEs general error interrupt. These events can be cleared only by the host writing this register with the value of 1 in the relevant bit.

Offset 0x00001604 (PSPICER1) 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												CRPE_ ULF_ ERR				eMSG _ERR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-267. PSIF PIC Event Register 1

Table 26-322. PSIF PIC Event Register 1 Fields Description

Bits	Description
31–5	Reserved
[4]	CRPE_ULF_ERR. CRPE-ULF Error Indication. For error type see **Section 26.5.5.6.6**, *ULF Event Status Register (ULFESR)*. Assertion of CRPE-ULF error indication requires CRPE reset as described in **Section 26.4.3.6.4**, *CRPE Reset*. 0 — No CRPE-ULF error indication occurred. 1 — CRPE-ULF error occurred. See **Section 26.5.5.6.6**, *ULF Event Status Register (ULFESR)* for error type.
[3–1]	Reserved.
0	eMSG_ERR

26.5.5.1.4 PSIF PIC Event Register 2 (PSPICER2)

This register describes ECC error events from all MAPLE-B2 memory groups which are ECC protected. Assertion of any of these bits causes the assertion of the MAPLE-B2s general ECC error event, which is a level interrupt only. These events can be cleared only by the host writing this register with the value of 1 in the relevant bit. Enabling the ECC memory protection in MAPLE-B2 is done by setting the [ECC_EN] bit of the MMC0P parameter in the API (see *MAPLE-B Application Programmer Interface (API) User's Guide* (MAPLEAPIUG)).

Offset 0x00001608 (PSPICER2) 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				CRUF_ ECC	CRUB_ ECC	CRDL_ ECC	EQ_E CC	EF2_ ECC	EF1_ ECC	EF0_ ECC	TV_CD _PERR	TV_D_ ECC	TV_E_ ECC	TV_H_ ECC	DRAM _ECC	IRAM_ ECC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-268. PSIF PIC Event Register 2

Table 26-323. PSIF PIC Event Register 2 Fields Description

Bits	Description
31–13	reserved.
12	CRUF_ECC — CRPE-ULF memory ECC Error. This bit indicates that an ECC has occur in one of the CRPE-ULF internal memories. For details see **Section 26.4.3.1.1, *Memory Error Correction/Detection Support***. (need to replace with section's ECC) Read '0' ECC Error event in one of the CRPE-ULF internal memories did not occur Read '1' ECC Error event in one of the CRPE-ULF internal memories occur Write '1' Clear bit if set.
11	CRUB_ECC — CRPE-ULB memory ECC Error. This bit indicates that an ECC has occur in one of the CRPE-ULB internal memories. For details see **Section 26.4.3.1.1, *Memory Error Correction/Detection Support***. (need to replace with section's ECC) Read '0' ECC Error event in one of the CRPE-ULB internal memories did not occur Read '1' ECC Error event in one of the CRPE-ULB internal memories occur Write '1' Clear bit if set.
10	CRDL_ECC — CRPE-DL memory ECC Error. This bit indicates that an ECC has occur in one of the CRPE-DL internal memories. For details see **Section 26.4.3.1.1, *Memory Error Correction/Detection Support***. (need to replace with section's ECC) Read '0' ECC Error event in one of the CRPE-DL internal memories did not occur Read '1' ECC Error event in one of the CRPE-DL internal memories occur Write '1' Clear bit if set.

Table 26-323. PSIF PIC Event Register 2 Fields Description (Continued)

Bits	Description
9	EQ_ECC — EQPE memory ECC Error. This bit indicates that an ECC has occur in one of the EQPE internal memories. For details see **Section 26.4.3.5.11**, *EQPE ECC Support*. Read ‘0’ ECC Error event in one of the EQPE internal memories did not occur Read ‘1’ ECC Error event in one of the EQPE internal memories occur Write ‘1’ Clear bit if set.
8	EF2_ECC —EFTPE_2 memory ECC Error. This bit indicates that an ECC has occur in one of the EFTPE_2 internal memories. For details see **Section 26.4.3.3.12, *eFTPE ECC Support***. Read ‘0’ ECC Error event in one of the EFTPE_2 internal memories did not occur Read ‘1’ ECC Error event in one of the EFTPE_2 internal memories occur Write ‘1’ Clear bit if set.
7	EF1_ECC —EFTPE_1 memory ECC Error. This bit indicates that an ECC has occur in one of the EFTPE_1 internal memories. For details see **Section 26.4.3.3.12, *eFTPE ECC Support***. Read ‘0’ ECC Error event in one of the EFTPE_1 internal memories did not occur Read ‘1’ ECC Error event in one of the EFTPE_1 internal memories occur Write ‘1’ Clear bit if set.
6	EF0_ECC —EFTPE_0 memory ECC Error. This bit indicates that an ECC has occur in one of the EFTPE_0 internal memories. For details see **Section 26.4.3.3.12, *eFTPE ECC Support***. Read ‘0’ ECC Error event in one of the EFTPE_0 internal memories did not occur Read ‘1’ ECC Error event in one of the EFTPE_0 internal memories occur Write ‘1’ Clear bit if set.
5	TV_CD_PERR—eTVPE CDL memory Parity Error. This bit indicates that Parity Error has occurred in the eTVPE Channel Data Memory. Read ‘0’ Parity Error in the eTVPE Channel Data memory event did not occur Read ‘1’ Parity Error in the eTVPE Channel Data memory event occur Write ‘1’ Clear bit if set.
4	TV_D_ECC—eTVPE DRE memory ECC Error. This bit indicates that ECC Error has occurred in the eTVPE DRE Memories. Read ‘0’ ECC Error in the eTVPE DRE memories event did not occur Read ‘1’ ECC Error in the eTVPE DRE memories event occur Write ‘1’ Clear bit if set.
3	TV_E_ECC—eTVPE EXT memory ECC Error. This bit indicates that ECC Error has occurred in the eTVPE Extrinsic Memories. Read ‘0’ ECC Error in the eTVPE Extrinsic memories event did not occur Read ‘1’ ECC Error in the eTVPE Extrinsic memories event occur Write ‘1’ Clear bit if set.
2	TV_H_ECC—eTVPE HARQ memory ECC Error. This bit indicates that ECC Error has occurred in the eTVPE HARQ Memories. Read ‘0’ ECC Error in the eTVPE HARQ memories event did not occur Read ‘1’ ECC Error in the eTVPE HARQ memories event occur Write ‘1’ Clear bit if set.
1	DRAM_ECC—DRAM memory ECC Error. This bit indicates that ECC Error has occurred in the DRAM Memories in PSIF2. Read ‘0’ ECC Error in DRAM memories event did not occur Read ‘1’ ECC Error in the event DRAM memories occur Write ‘1’ Clear bit if set.
0	IRAM_ECC—IRAM memory ECC Error. This bit indicates that ECC Error has occurred in the IRAM Memories in PSIF2. Read ‘0’ ECC Error in IRAM memories event did not occur Read ‘1’ ECC Error in the event IRAM memories occur Write ‘1’ Clear bit if set.

26.5.5.1.5 PSIF PIC Edge/Level Register 0 (PSPICELR0)

For each Buffer Descriptors Ring interrupt, this register describes whether it is output as an edge or level interrupt.

Offset 0x0000160C (PSPICELR0) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BDR_DONE_EL[31:16]															
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	0	0	0	0	0	0	0	0	0	0	0

Figure 26-269. PSIF PIC Edge/Level Register 0

Table 26-324. PSIF PIC Edge/Level Register 0 Field Description

Bits	Description
31–0	BDR_DONE_EL[31:0]—BDR_DONE interrupt Edge/Level indication.Each bit in this register fits its event in the PSPICER0[31:0] register. For each of these bits: ‘0’ The interrupt assertion is of ‘level’ type. ’1’ The interrupt assertion is of ‘edge’ type.

26.5.5.1.6 PSIF PIC Mask Register 0 (PSPICMR0)

This register describes the mask bit for each possible BDR_DONE interrupt line. For each of the bits in this register, if disabled (0), the relevant BDR_DONE interrupt line is not asserted in case of event. The relevant bit in the PSIF PIC Event Register 0 (PSPICER0) is still asserted (if the PSIF PIC Edge/Level Register 0 (PSPICELR0) configuration of that bit is 0, that is, level configuration).

Offset 0x00001610 (PSPICMR0) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BDR_DONE_M[31:16]															
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[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-270. PSIF PIC Mask Register 0

Table 26-325. PSIF PIC Mask Register 0 Fields Description

Bits	Description
31–0	BDR_DONE_M[31:0]—Buffer Descriptor Done event Mask indication. For each of the bits: 0 The ipi_bdr_done[x] interrupt line is masked and are not asserted due to BDR_DONE[x] event assertion. 1 The ipi_bdr_done[x] interrupt line is enabled and an interrupt are asserted upon BDR_DONE[x] event assertion

26.5.5.1.7 PSIF PIC Mask Register 1 (PSPICMR1)

This register describes the mask bit for each possible event which trigger the MAPLE-B2s general error interrupt line. For each of the bits in this register, if disabled (0), the relevant event as described in PSIF PIC Event Register 1 (PSPICER1) does not cause the assertion of the MAPLE-B2s general error interrupt. The relevant bit in the PSIF PIC Event Register 1 (PSPICER1) is still asserted regardless of its mask bit.

Offset 0x00001614 (PSPICMR1) 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												CRPE_ ULF_ ERR_M				eMSG_ ERR_M
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-271. PSIF PIC Mask Register 1

Table 26-326. PSIF PIC Mask Register 1 Fields Description

Bits	Description
31–5	Reserved
[4]	CRPE_ULF_ERR_M. CRPE-ULF Error mask indication. 0 — CRPE-ULF error indication is disabled. 1 — CRPE-ULF error indication is enabled.
[3–1]	Reserved.
0	EMSG_ERR_M

26.5.5.1.8 PSIF PIC Mask Register 2 (PSPICMR2)

This register describes the mask bit for each possible event which trigger the MAPLE-B2s ECC general error interrupt line. For each of the bits in this register, if disabled (0), the relevant event as described in PSIF PIC Event Register 2 (PSPICER2) does not cause the assertion of the MAPLEs ECC general error interrupt. The relevant bit in the PSIF PIC Event Register 2 (PSPICER2) is still asserted regardless of its mask bit.

Offset 0x00001618 (PSPICMR2) 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				CRUF_ ECC_M	CRUB_ ECC_M	CRDL_ ECC_M	EQ_E CC_M	EF2_E CC_M	EF1_E CC_M	EF0_E CC_M	TV_CD_ PERR_M	TV_D_ ECC_M	TV_E_ ECC_M	TV_H_ ECC_M	DRAM_ ECC_M	IRAM_ ECC_M
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-272. PSIF PIC Mask Register 2

Table 26-327. PSIF PIC Mask Register 2 Fields Description

Bits	Description
31–13	reserved.
12	CRUF_ECC_M — CRPE-ULF memory ECC Error event Mask indication. 0 The MAPLE-B2's ECC general error interrupt line does not assert due to CRPE-ULF memory ECC error event assertion. 1 Assertion of the CRPE-ULF memory ECC error event causes the assertion of the MAPLE-B2's ECC general error interrupt line.
11	CRUB_ECC_M — CRPE-ULB memory ECC Error event Mask indication. 0 The MAPLE-B2's ECC general error interrupt line does not assert due to CRPE-ULB memory ECC error event assertion. 1 Assertion of the CRPE-ULB memory ECC error event causes the assertion of the MAPLE-B2's ECC general error interrupt line.
10	CRDL_ECC_M — CRPE-DL memory ECC Error event Mask indication. 0 The MAPLE-B2's ECC general error interrupt line does not assert due to CRPE-DL memory ECC error event assertion. 1 Assertion of the CRPE-DL memory ECC error event causes the assertion of the MAPLE-B2's ECC general error interrupt line.
9	EQ_ECC_M — EQPE memory ECC Error event Mask indication. 0 The MAPLE-B2's ECC general error interrupt line does not assert due to EQPE memory ECC error event assertion. 1 Assertion of the EQPE memory ECC error event causes the assertion of the MAPLE-B2's ECC general error interrupt line.

Table 26-327. PSIF PIC Mask Register 2 Fields Description (Continued)

Bits	Description
8	EF2_ECC_M — EFTPE_2 memory ECC Error event Mask indication. 0 The MAPLE-B2's ECC general error interrupt line does not assert due to EFTPE_2 memory ECC error event assertion. 1 Assertion of the EFTPE_2 memory ECC error event causes the assertion of the MAPLE-B2's ECC general error interrupt line.
7	EF1_ECC_M — EFTPE_1 memory ECC Error event Mask indication. 0 The MAPLE-B2's ECC general error interrupt line does not assert due to EFTPE_1 memory ECC error event assertion. 1 Assertion of the EFTPE_1 memory ECC error event causes the assertion of the MAPLE-B2's ECC general error interrupt line.
6	EF0_ECC_M — EFTPE_0 memory ECC Error event Mask indication. 0 The MAPLE-B2's ECC general error interrupt line does not assert due to EFTPE_0 memory ECC error event assertion. 1 Assertion of the EFTPE_0 memory ECC error event causes the assertion of the MAPLE-B2's ECC general error interrupt line.
5	TV_CD_PERR_M—eTVPE CDL memory Parity Error event Mask indication. 0 The MAPLE-B2's ECC general error interrupt line does not assert due to eTVPE CDL memory Parity error event assertion. 1 Assertion of the eTVPE CDL memory Parity error event causes the assertion of the MAPLE-B2's ECC general error interrupt line.
4	TV_D_ECC_M—eTVPE DRE memory ECC Error event Mask indication. 0 The MAPLE-B2's ECC general error interrupt line does not assert due to eTVPE DRE memory ECC error event assertion. 1 Assertion of the eTVPE DRE memory ECC error event causes the assertion of the MAPLE-B2's ECC general error interrupt line.
3	TV_E_ECC_M—eTVPE Extrinsic memory ECC Error event Mask indication. 0 The MAPLE-B2's ECC general error interrupt line does not assert due to eTVPE Extrinsic memory ECC error event assertion. 1 Assertion of the eTVPE Extrinsic memory ECC error event causes the assertion of the MAPLE-B2's ECC general error interrupt line.
2	TV_H_ECC_M—eTVPE HARQ memory ECC Error event Mask indication. 0 The MAPLE-B2's ECC general error interrupt line does not assert due to eTVPE HARQ memory ECC error event assertion. 1 Assertion of the eTVPE HARQ memory ECC error event causes the assertion of the MAPLE-B2's ECC general error interrupt line.
1	DRAM_ECC_M—DRAM memory ECC Error event Mask indication. 0 The MAPLE-B2's ECC general error interrupt line does not assert due to DRAM memory ECC error event assertion. 1 Assertion of the DRAM memory ECC error event causes the assertion of the MAPLE-B2's ECC general error interrupt line.
0	IRAM_ECC_M—IRAM memory ECC Error event Mask indication. 0 The MAPLE-B2's ECC general error interrupt line does not assert due to IRAM memory ECC error event assertion. 1 Assertion of the IRAM memory ECC error event causes the assertion of the MAPLE-B2's ECC general error interrupt line.

26.5.5.1.9 PSIF PIC Interrupts Assertion Clocks Register (PSPICIACR)

For all the BDR_DONE interrupts, this register describes the number of PSIF2 clock cycles the interrupt is asserted in case it is configured as an edge interrupt.

Offset 0x0000161C (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 26-273. PSIF PIC Interrupt Assertion Clocks Register

Table 26-328. PSIF PIC Interrupt Assertion Clocks Register Field Description

Bits	Description
31–3	Reserved
2–0	IAC—Interrupt Assertions Clocks. Describes the number of clock cycles the BDR_DONE interrupts are asserted if they are configured as edge interrupts. 0 The edge interrupt is asserted for 2 PSIF2 clock cycle. 1 The edge interrupt is asserted for 4 PSIF2 clock cycles. 2 The edge interrupt is asserted for 6 PSIF2 clock cycles. 3 The edge interrupt is asserted for 8 PSIF2 clock cycles. 4 The edge interrupt is asserted for 10 PSIF2 clock cycles. 5 The edge interrupt is asserted for 12 PSIF2 clock cycles. 6 The edge interrupt is asserted for 14 PSIF2 clock cycles. 7 The edge interrupt is asserted for 16 PSIF2 clock cycles.

26.5.5.2 eTVPE Registers Description

Table 26-329 describes the eTVPE registers in detail.

Table 26-329. eTVPE Registers Detailed Description

Address	Register	Access	Reset Value	Section
Configuration Registers				
0x0007E000	TVPEC0R. eTVPE Configuration 0 Register	R/W	0x00000000	**Section 26.5.5.2.1**
0x0007E03C	TVAQCR. eTVPE Aposteriori Quality Configuration Register	R/W	0x00000000	**Section 26.5.5.2.2**

26.5.5.2.1 eTVPE Configuration 0 Register (TVPEC0R)

This register contains the general configuration parameters for the eTVPE. All reserved bits must be cleared to 0.

Offset 0x0007E000 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															DOSBY	DOSBI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 26-274. eTVPE Configuration 0 Register

Table 26-330. eTVPE Configuration 0 Fields Description

Bits	Description
31–2	Reserved
1	DOSBY—Data out Byte order 0—Descending order 1—Ascending order
0	DOSBI—Data out Bit order 0—Descending order 1—Ascending order

26.5.5.2.2 eTVPE Aposteriori Quality Configuration Register (TVAQCR)

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.

Offset 0x0007E03C 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 26-275. eTVPE Aposteriori Quality Configuration Register

Table 26-331. eTVPE Aposteriori Quality Configuration Fields Description

Bits	Description
31–9	Reserved
8–0	AQTH—Aposteriori Quality Threshold. Describes the LLR absolute value used as threshold for Aposteriori quality stop criteria. Once all LLR absolute values of iteration N are larger than this field, the eTVPE decoding operation for this job is completed. For details see **Section 26.4.3.2.6.4.4, *Turbo Stopping Criteria Configurations***."

26.5.5.3 eFTPE_x Registers Description

Table 26-332 describes the registers of each of the three eFTPE modules instantiated in MAPLE-B2.

Table 26-332. eFTPE Registers Detailed Description

Address	Register	Access	Reset Value	Section
eFTPE_0 Registers				
0x000BE140–0x000BE14B	EFTPE0DSRx. EFTPE_0 Data Size Registers	R/W	See **Section 26.4.3.3.4.1.1**	**Section 26.5.5.3.1 Section 26.5.5.3.2 Section 26.5.5.3.3**
0x000BE1A0	EFTPE0CR - EFTPE_0 Configuration Register	R/W	0x0000_0000	**Section 26.5.5.3.4**
0x000BE1A8	EFTPE0ECCISR - EFTPE_0 ECC Interrupt Status Register	R/W	0x0000_0000	**Section 26.5.5.3.5**
eFTPE_1 Registers				
0x000FE140–0x000FE14B	EFTPE1DSRx. EFTPE_1 Data Size Registers	R/W	See **Section 26.4.3.3.4.1.1**	**Section 26.5.5.3.1 Section 26.5.5.3.2 Section 26.5.5.3.3**
0x000FE1A0	EFTPE1CR - EFTPE_1 Configuration Register	R/W	0x0000_0000	**Section 26.5.5.3.4**
0x000FE248	EFTPE1ECCISR - EFTPE_1 ECC Interrupt Status Register	R/W	0x0000_0000	**Section 26.5.5.3.5**
eFTPE_2 Registers				
0x0013E140–0x0013E14B	EFTPE2DSRx. EFTPE_2 Data Size Registers	R/W	See **Section 26.4.3.3.4.1.1**	**Section 26.5.5.3.1 Section 26.5.5.3.2 Section 26.5.5.3.3**
0x0013E1A0	EFTPE2CR - EFTPE_2 Configuration Register	R/W	0x0000_0000	**Section 26.5.5.3.4**
0x0013E248	EFTPE2ECCISR - EFTPE_2 ECC Interrupt Status Register	R/W	0x0000_0000	**Section 26.5.5.3.5**

26.5.5.3.1 EFTPE_<x> Data Size Register 0 (FTPE<x>DSR0)

This register, along with FTPE<x>DSR1 and FTPE<x>DSR2, record half the data size for iFFTs (DSx = half data size = [<transform length> - <guards size> - 1 (for DC)] / 2), as described by **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT* on page 26-83.

Offset FTPE_<x>_BASE+0x140 (FTPE<x>DSR0) 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 26-276. EFTPE_<x> Data Size Register 0

Table 26-333. EFTPE_<x> Data Size Register 0 Field Descriptions

Bits	Description
31–26	Reserved
25–16	DS0—Data Size 0. Size of the Right/Left data in an iFFT Transform with the length of 128. The value of the DS0 should not exceed the 4≤DS0 ≤ ((128/2) - 2) values. For details, see "**Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT*" on page 26-83.
15–10	Reserved
9–0	DS1—Data Size 1. Size of the Right/Left data in an iFFT Transform with the length of 256. The value of the DS1 should not exceed the 4≤DS1 ≤ ((256/2) - 2) values. For details, see "**Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT*" on page 26-83.

26.5.5.3.2 EFTPE_<x> Data Size Register 1 (FTPE<x>DSR1)

This register, along with FTPE<x>DSR0 and FTPE<x>DSR1, records half the data size for iFFTs (DSx = half data size = [<transform length> - <guards size> - 1 (for DC)] / 2), as described by **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT* on page 26-83.

Offset FTPE_<x>_BASE+0x144 (FTPE<x>DSR1) 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 26-277. EFTPE_<x> Data Size Register 1

Table 26-334. EFTPE_<x> Data Size Register 1 Field Descriptions

Bits	Description
31–26	Reserved
25–16	DS2—Data Size 2. Size of the Right/Left data in an iFFT Transform with the length of 512. The value of the DS2 should not exceed the 4≤DS2≤((512/2) - 2) values. For details, see "**Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT*" on page 26-83.
15–10	Reserved
9–0	DS3—Data Size 3. Size of the Right/Left data in an iFFT Transform with the length of 1024. The value of the DS3 should not exceed the 4≤DS3≤((1024/2) - 2) values. For details, see "**Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT*" on page 26-83.

26.5.5.3.3 EFTPE_<x> Data Size Register 2 (FTPE<x>DSR2)

This register, along with FTPE<x>DSR0 and FTPE<x>DSR1, records half the data size for iFFTs (DSx = half data size = [<transform length> - <guards size> - 1 (for DC)] / 2), as described by **Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT*" on page 26-83.

Offset FTPE_<x>_BASE+0x148 (FTPE<x>DSR2) 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	DS5									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-278. EFTPE_<x> Data Size Register 2

Table 26-335. EFTPE_<x> Data Size Register 2Field Descriptions

Bits	Description
31–26	Reserved
25–16	DS4—Data Size 4. Size of the Right/Left data in an iFFT Transform with the length of 1536. The value of the DS4 should not exceed the 4≤ DS4 ≤ ((1536/2) - 2) values. For details, see "**Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT*" on page 26-83.
15–10	Reserved
9–0	DS5—Data Size 5. Size of the Right/Left data in an iFFT Transform with the length of 2048. The value of the DS5 should not exceed the 4≤ DS5 ≤ ((2048/2) - 2) values. For details, see "**Section 26.4.3.3.4.1.1**, *Guard Band Insertion for iFFT*" on page 26-83.

26.5.5.3.4 EFTPE_<x> Configuration Register (FTPE<x>CR)

This register contains global configurations of the eFTPE. The configuration should be done once as part of MAPLE initialization. For details see **Section 26.4.3.3.10**, *eFTPE Initialization Configuration*

Offset FTPE_<x>_BASE+0x1A0 (FTPE<x>CR) 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	0									
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						BCIS	BC		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-279. EFTPE_<x> Configuration Register

Table 26-336. EFTPE_<x> Configuration Register Field Descriptions

Bits	Description
31–4	Reserved
3	BCIS — Backward Compatible Input Scale. 1 - The eFTPE scale results are forced to be backward compatible, and also overall scaling mode is calculated with backward compatibility. 0 - The eFTPE scale results are not forced to be backward compatible, and also overall scaling mode is not calculated with backward compatibility. See **Section 26.4.3.3.9.4.6**, *User Defined Input Scaling* for details.
2–0	BC — Buffers Configuration. Configures the pool of buffers according to the following: 0 - one output buffer, one post multiplier buffer and one pre multiplier buffer. 1 - double output buffer, one pre - multiplier buffer 2 - double output buffer, one post multiplier buffer. 3 - Reserved. 4 - one output buffer, double post multiplier buffer. 5 to 7 - Reserved

26.5.5.3.5 EFTPE_<x> ECC Interrupt Status Register (FTPE<x>ECCISR)

This register contains status events of ECC interrupts in the eFTPE. Resetting an ECC status event is done by writing ‘1’ to the relevant bit.

Offset FTPE_<x>_BASE+ 0x248 (FTPE<x>ECCISR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	BP2	BP_1	BP_0	IDBC_3	IDBC_2	IDBC_1	IDBC_0	IDBB_3	IDBB_2	IDBB_1	IDBB_0	IDBA_3	IDBA_2	IDBA_1	IDBA_0
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									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-280. EFTPE_<x> Interrupt Status Register

Table 26-337. EFTPE_<x> Interrupt Status Field Descriptions

Bits	Description
31	Reserved
30	BP_2 - Buffer Pool module 2. ECC event indication for module #2 of the buffer pool. 0 - No ECC error has occur. 1 - ECC error has occur.
29	BP_1 - Buffer Pool module 1.ECC event indication for module #1 of the buffer pool. 0 - No ECC error has occur. 1 - ECC error has occur.
28	BP_0 - Buffer Pool module 0. ECC event indication for module #0 of the buffer pool. 0 - No ECC error has occur. 1 - ECC error has occur.
27	IDBC_3 - Input Data Buffer C module 3. ECC event has occur on module #3 of input data buffer C. 0 - No ECC error has occur. 1 - ECC error has occur.
26	IDBC_2 - Input Data Buffer C module 2. ECC event has occur on module #2 of input data buffer C. 0 - No ECC error has occur. 1 - ECC error has occur.
25	IDBC_1 - Input Data Buffer C module 1. ECC event has occur on module #1 of input data buffer C. 0 - No ECC error has occur. 1 - ECC error has occur.
24	IDBC_0 - Input Data Buffer C module 0. ECC event has occur on module #0 of input data buffer C. 0 - No ECC error has occur. 1 - ECC error has occur.
23	IDBB_3 - Input Data Buffer B module 3. ECC event has occur on module #3 of input data buffer B. 0 - No ECC error has occur. 1 - ECC error has occur.

Table 26-337. EFTPE_<x> Interrupt Status Field Descriptions (Continued)

Bits	Description
22	IDBB_2 - Input Data Buffer B module 2. ECC event has occur on module #2 of input data buffer B. 0 - No ECC error has occur. 1 - ECC error has occur.
21	IDBB_1 - Input Data Buffer B module 1. ECC event has occur on module #1 of input data buffer B. 0 - No ECC error has occur. 1 - ECC error has occur.
20	IDBB_0 - Input Data Buffer B module 0. ECC event has occur on module #0 of input data buffer B. 0 - No ECC error has occur. 1 - ECC error has occur.
19	IDBA_3 - Input Data Buffer A module 3. ECC event has occur on module #3 of input data buffer A. 0 - No ECC error has occur. 1 - ECC error has occur.
18	IDBA_2 - Input Data Buffer A module 2. ECC event has occur on module #2 of input data buffer A. 0 - No ECC error has occur. 1 - ECC error has occur.
17	IDBA_1 - Input Data Buffer A module 1. ECC event has occur on module #1 of input data buffer A. 0 - No ECC error has occur. 1 - ECC error has occur.
16	IDBA_0 - Input Data Buffer A module 0. ECC event has occur on module #0 of input data buffer A. 0 - No ECC error has occur. 1 - ECC error has occur.
15–0	Reserved

26.5.5.4 EQPE Registers Description

Table 26-338 describes the EQPE registers in detail.

Table 26-338. EQPE Registers Detailed Description

Address	Register	Access	Reset Value	Section
Configuration Registers				
0x001BFA08	EQ_THRESH - Threshold Register	R/W	0x0000_0000	**Section 26.5.5.4.1**
Status Registers				
0x001BFB08	EQ_ECCEVENT - ECC Event Register	R/W	0x0000_0000	**Section 26.5.5.4.2**

26.5.5.4.1 EQPE Threshold Register (EQ_THRESH)

This register configures the threshold value for the matrix singularity check. For details see **Section 26.4.3.5.10**, *EQPE Status Indications*.

Offset 0x001BFA08 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									SING_THRESH							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-281. EQPE Threshold Register

Table 26-339. EQPE Threshold Fields Description

Bits	Description
31–8	Reserved
7–0	SING_THRESH—Threshold for the singularity check.This field is in signed 1Q7 format and represents the exponent of the threshold. Any matrix which has one or more exponent of the diagonal values of the R matrix (after QR decomposition) which are smaller than the threshold value is reported as singular (close to singular). The total number of singular matrices is reported in EQPE BD *SING*] field which can be read after the EQPE job is finished.

26.5.5.4.2 EQPE ECC Event Register (EQ_ECCEVENT)

This register records ECC events of the EQPE memories. For details see **Section 26.4.3.5.11**, *EQPE ECC Support*.

Offset 0x001BFB08 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ECC_ERR										OUT3_ECC	OUT2_ECC	OUT1_ECC	OUT0_ECC	OSCL_ECC	SCL_ECC
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	W1_ECC	W0_ECC	Y1_ECC	Y0_ECC	F1_ECC	F0_ECC	S1_ECC	S0_ECC	H7_ECC	H6_ECC	H5_ECC	H4_ECC	H3_ECC	H2_ECC	H1_ECC	H0_ECC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-282. EQPE ECC Event Register

Table 26-340. EQPE ECC Event Fields Description

Bits	Description
31	ECC_ERR — ECC Error - when set this bit indicates that a multiple-bit error has occurred in one of the EQPE memories. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - No ECC event has occur in any of EQPE memories. 1 - ECC event has occur in one of EQPE memories.
30–22	Reserved
21	OUT3_ECC — Output Buffer 3 multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
20	OUT2_ECC — Output Buffer 2 multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
19	OUT1_ECC — Output Buffer 1 multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
18	OUT0_ECC — Output Buffer 0 multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
17	OSCL_ECC — Output Scale buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
16	SCL_ECC — Input Scale buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
15	W1_ECC — Interpolation Weights 1 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
14	W0_ECC — Interpolation Weights 0 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
13	Y1_ECC — Y 1 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
12	Y0_ECC — Y 0 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
11	F1_ECC — Feedback 1 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.

Table 26-340. EQPE ECC Event Fields Description

Bits	Description
10	F0_ECC — Feedback 0 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
9	S1_ECC — S1 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
8	S0_ECC — S0 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
7	H7_ECC — H7 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
6	H6_ECC — H6 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
5	H5_ECC — H5 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
4	H4_ECC — H4 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
3	H3_ECC — H3 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
2	H2_ECC — H2 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
1	H1_ECC — H1 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.
0	H0_ECC — H0 buffer multiple bits ECC error. This bit can be cleared by the host, by writing 1 to this bit, writing 0 to this bit, has no effect. 0 - ECC event hasn't occurred. 1 - ECC multiple bits error has occurred.

26.5.5.5 CRPE-ULB Registers Description

Table 26-341 describes the CRPE-ULB registers in detail.

Table 26-341. CRPE-ULB Registers Detailed Description

Address	Register	Access	Size	Reset Value	Section
Configuration Registers					
0x0025F00C	Reserved				
0x0025F010–0x0025F02F	CRUBIW1SxCR - CRPE-ULB Interpolation Weights 1 Sample x Configuration Register (x = 0...7)	R/W	4B	0x0000_0000	**Section 26.5.5.5.1**
0x0025F030–0x0025F04F	CRUBIW2SxCR - CRPE-ULB Interpolation Weights 2 Sample x Configuration Register (x = 0...7)	R/W	4B	0x0000_0000	**Section 26.5.5.5.2**
0x0025F050–0x0025F067	CRUBGFAxCR - CRPE-ULB Group First Antenna x Configuration Register (x = 0...5)	R/W	4B	0x0000_0000	**Section 26.5.5.5.3**
0x0025F068–0x0025F07F	CRUBGNOAxCR - CRPE-ULB Group Number of Antenna x Configuration Register (x = 0...5)	R/W	4B	0x0000_0000	**Section 26.5.5.5.4**
Status Registers					
0x0025F800	CRUBESR - CRPE Uplink Batch Event Status Register	R	4B	0x0000_0000	**Section 26.5.5.5.5**
0x0025F820	CRUBOSCSR - CRPE-ULB Output Sat Counter Status Register	R	4B	0x0000_0000	**Section 26.5.5.5.6**
0x0025F824	CRUBISCSR - CRPE-ULB Interpolation Sat Counter Status Register	R	4B	0x0000_0000	**Section 26.5.5.5.7**

26.5.5.5.1 CRPE-ULB Interpolation Weights 1 Sample <x> Configuration Register (CRUBIW1SxCR)

This register contains the interpolation weights for CRPE-ULB Interpolation operation.

Offset 0x0025F010 offset 4 x = 0... 7 — Access: User read/write

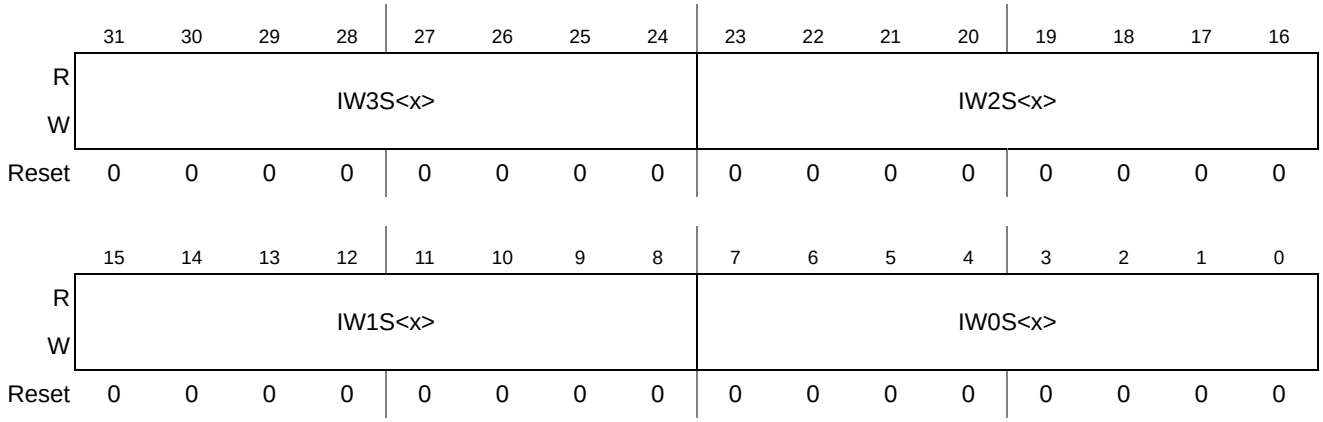


Figure 26-283. CRPE-ULB Interpolation Weights 1 Sample <x> Configuration Register

Table 26-342. CRPE-ULB Interpolation Weights 1 Sample <x> Configuration Fields Description

Bits	Description
31–24	IW3S<x> — Signed interpolated weight 3, phase x Valid values: -128 to 127
23–16	IW2S<x> — Signed interpolated weight 2, phase x Valid values: -128 to 127
15–8	IW1S<x> — Signed interpolated weight 1, phase x Valid values: -128 to 127
7–0	IW0S<x> — Signed interpolated weight 0, phase x Valid values: -128 to 127

26.5.5.5.2 CRPE-ULB Interpolation Weights 2 Sample <x> Configuration Register (CRUBIW2SxCR)

This register contains the interpolation weights for CRPE-ULB Interpolation operation.

Offset 0x0025F030 Access: User read/write
offset 4
x = 0... 7

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IW7S<x>								IW6S<x>							
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	IW5S<x>								IW4S<x>							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-284. CRPE-ULB Interpolation Weights 2 Sample <x> Configuration Register

Table 26-343. CRPE-ULB Interpolation Weights 2 Sample <x> Configuration Fields Description

Bits	Description
31–24	IW7S<x> — Signed interpolated weight 7, phase x Valid values: -128 to 127
23–16	IW6S<x> — Signed interpolated weight 6, phase x Valid values: -128 to 127
15–8	IW5S<x> — Signed interpolated weight 5, phase x Valid values: -128 to 127
7–0	IW4S<x> — Signed interpolated weight 4, phase x Valid values: -128 to 127

26.5.5.5.3 CRPE-ULB Group First Antenna <x> Configuration Register (CRUBGFAxCR)

This register contains the first antenna for each group.

Offset 0x0025F050 Access: User read/write
offset 4
x = 0... 5

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	G<4x>_FANT								G<4x+1>_FANT							
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	G<4x+2>_FANT								G<4x+3>_FANT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-285. CRPE-ULB Group First Antenna x Configuration Register

Table 26-344. CRPE-ULB Group First Antenna x Configuration Fields Description

Bits	Description
31–30	Reserved
29–24	G<4x>FANT — First Antenna For Group <4x> Valid values are 0 to 47
23–22	Reserved
21–16	G<4x+1>FANT — First Antenna For Group <4x+1> Valid values are 0 to 47
15–14	Reserved
13–8	G<4x+2>FANT — First Antenna For Group <4x+2> Valid values are 0 to 47
7–6	Reserved
5–0	G<4x+3>FANT — First Antenna For Group <4x+3> Valid values are 0 to 47

26.5.5.5.4 CRPE-ULB Group Number Of Antenna <x> Configuration Register (CRUBGNOAxCR)

This register contains the number of antennas for each group.

Offset 0x0025F068
offset 4
x = 0... 5

Access: User read/write

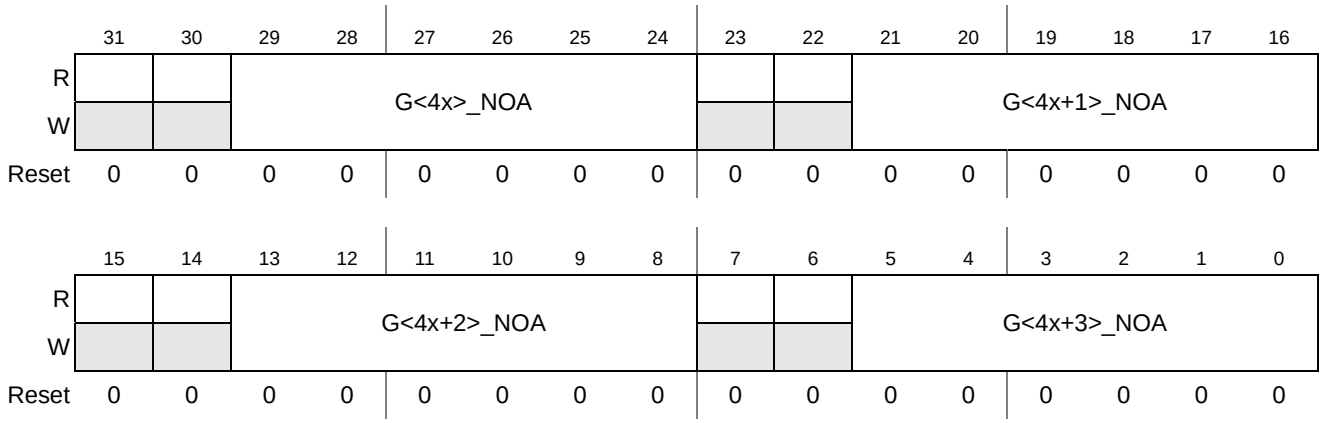


Figure 26-286. CRPE-ULB Group number Of Antenna x Configuration Register

Table 26-345. CRPE-ULB Group Number Of Antenna x Configuration Fields Description

Bits	Description
31–30	Reserved
29–24	G<4x>NOA — Number of Antenna For Group <4x> Valid values are 0 to 47
23–22	Reserved
21–16	G<4x+1>NOA — Number of Antenna For Group <4x+1> Valid values are 0 to 47
15–14	Reserved
13–8	G<4x+2>NOA — Number of Antenna For Group <4x+2> Valid values are 0 to 47
7–6	Reserved
5–0	G<4x+3>NOA — Number of Antenna For Group <4x+3> Valid values are 0 to 47

26.5.5.5.5 CRPE-ULB Event Status Register (CRUBESR)

This register includes status bits indication for all multiple ECC error events.

Offset 0x0025F800 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			OBPMME_ES		PCHMME_ES		PCQMME_ES		FCQMME_ES		FILEME_ES		OBME_ES		IBME_ES	
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																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-287. CRPE-ULB Event Status Register

Table 26-346. CRPE-ULB Event Status Fields Description

Bits	Description
31–30	Reserved
29	OBPMME_EM — Output Params memory multiple ECC error indication. 0 - ECC event has not occur since last time it was reset 1 - ECC event has occur since last time it was reset
28	Reserved
27	PCHMME_EM — PCH Params memory multiple ECC error indication. 0 - ECC event has not occur since last time it was reset 1 - ECC event has occur since last time it was reset
26	Reserved
25	PCQMME_EM — PCH Command Queue memory multiple ECC error indication. 0 - ECC event has not occur since last time it was reset 1 - ECC event has occur since last time it was reset
24	Reserved
23	FCQMME_EM — Fingers Command Queue memory multiple ECC error indication. 0 - ECC event has not occur since last time it was reset 1 - ECC event has occur since last time it was reset
22	Reserved
21	FILEME_EM — Linked List Entries memory multiple ECC error indication. 0 - ECC event has not occur since last time it was reset 1 - ECC event has occur since last time it was reset
20	Reserved

Table 26-346. CRPE-ULB Event Status Fields Description

Bits	Description
19	OBME_EM — Output Buffer memory multiple ECC error indication. 0 - ECC event has not occur since last time it was reset 1 - ECC event has occur since last time it was reset
18	Reserved
17	IBME_EM — Input Buffer memory multiple ECC error indication. 0 - ECC event has not occur since last time it was reset 1 - ECC event has occur since last time it was reset
16–0	Reserved

26.5.5.5.6 CRPE-ULB Output Saturation Counter Status Register (CRUBOSCSR)

This register indicates the number of times the Output result was saturate. Writing to this registers resets it - the data is ignored.

Offset 0x0025F820 Access: User read/write

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

Figure 26-288. CRPE-ULB Output Saturation Counter Status Register

Table 26-347. CRPE-ULB Output Saturation Counter Status Fields Description

Bits	Description
31–0	OS_COUNT — Output Saturation Counter. Indicates the number of times an Output symbol was saturated

26.5.5.5.7 CRPE-ULB Interpolation Saturation Counter Status Register (CRUBISCSR)

This register indicates the number of times the interpolation result was saturated after shift by *ISHA*, when truncating back to 8I,8Q. Writing to this registers resets it; the written data is ignored.

Offset 0x0025F824 Access: User read/write

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

Figure 26-289. CRPE-ULB Interpolation Saturation Counter Status Register

Table 26-348. CRPE-ULB Interpolation Saturation Counter Status Fields Description

Bits	Description
31–0	IS_COUNT — Interpolation Saturation Counter. Indicates the number of times an interpolated chip was saturated.

26.5.5.6 CRPE-ULF Registers Description

Table 26-349 lists the CRPE-ULF registers.

Table 26-349. CRPE-ULF Registers Detailed Description

Address	Register	Access	Reset Value	Section
Configuration Registers				
0x0023E000	ULFGCR - UL FAST General Configuration Register	R/W	0x0000_0000	**Section 26.5.5.6.1**
0x0023E004– 0x0023E00B	Reserved			
0x0023E00C	ULFSCR - UL FAST Secondary Configuration Register	R/W	0x00000000	**Section 26.5.5.6.2**
0x0023E010– 0x0023E07F	Reserved			
0x0023E080– 0x0023E0BF	ULFICRx - UL FAST Interpolation Configuration Register <x> (x = 0...15)	R/W	0x0000_0000	**Section 26.5.5.6.3**
0x0023E0C0– 0x0023E0FF	Reserved			

Table 26-349. CRPE-ULF Registers Detailed Description (Continued)

Address	Register	Access	Reset Value	Section
0x0023E100–0x0023E18F	ULFOBxBCR - UL FAST Output Buffer <x> Base Configuration Register (x = 0...17)	R/W	0x0000_0000	**Section 26.5.5.6.4**
	ULFOBxACR - UL FAST Output Buffer <x> Attributes Configuration Register (x = 0...17)	R/W	0x0000_0000	**Section 26.5.5.6.5**
0x0023E190–0x0023E27FF	Reserved			
Status Registers				
0x0023E280	ULFESR - UL FAST Event Status Register	R/W	0x0000_0000	**Section 26.5.5.6.6**
0x0023E290	ULFCFSR - UL FAST Commands FIFO Status Register	R/W	0x0000_0000	**Section 26.5.5.6.7**
0x0023E294	ULFIBSR - UL FAST IB Status Register	R/W	0x0000_0000	**Section 26.5.5.6.8**
0x0023E298	ULFTSR - UL FAST Time Status Register	R/W	0x0000_0000	**Section 26.5.5.6.9**
0x0023E29C	ULFECCSR - UL FAST ECC Status Register	R/W	0x0000_0000	**Section 26.5.5.6.10**

26.5.5.6.1 ULF General Configuration Register (ULFGCR)

This register (along with *ULFSCG*) defines the general configuration parameters of the CRPE-ULF.

Offset 0x0023E000 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MNOF								OBUR		DDS	INB	MDS	PCE	MP	
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	OVS		IBM	NOA					ELD3	ELD2	ELD1	ELD0	IS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-290. ULF General Configuration Register

Table 26-350. ULF General Configuration Fields Description

Bits	Description
31–24	MNOF — Maximum Number Of Fingers. This field should be set to 200 (0xC8), which is the maximum number of fingers (in resolution of 16 fingers) representing 3200 supported fingers.
23–22	OBUR — Output Buffer Update Rate. This field set the rate the status of the output buffer is updated by ULF 0 - The status of the output buffer is updated up-to one time in 32 chips processing period. 1 - The status of the output buffer is updated up-to two times in 32 chips processing period. 2 - The status of the output buffer is updated up-to four times in 32 chips processing period. 3 - The status of the output buffer is updated up-to eight times in 32 chips processing period.
21	DDS — DD Shift. The intermediate DD result is shift DDS bits right. 0 - No shift to intermediate DD results. 1 - Intermediate DD results are right shifted by 1.
20	INB — Interpolation Bypass. 0 - Interpolation logic is enabled 1 - Interpolation logic is bypassed
19	MDS — Maximum Delay Spread. This field set the Maximum Delay Spread of the fingers 0 - The maximum delay spread is 256 chips 1 - The maximum delay spread is 512 chips
18	PCE — Pilot Correlation Enable. 0 - ULF doesn’t correlate the PILOT Soft Symbol with the corresponding PILOT bit 1 - ULF correlates the PILOT Soft Symbol with the corresponding PILOT bit
17–16	MP — MBus Priority. This field set the priority of the MBus Initiator requests. Range: 0 to 3. (0 is the highest priority).
15–14	OVS — Over Sampling. This field selects the Over Sampling rate (OVS) of the input data: 0 - Input data is with OVS_2 1 - Input data is with OVS_4 2 - Input data is with OVS_8 3 - Input data is with OVS_{16}
13	IBM — Input Buffer Mode. Should be set to 1 when working with CPRI (streaming mode) or if interpolation bypass is enabled. 0 - The input buffer is accessed directly. 1 - The input buffer is accesses through 32 chips windows.
12–8	NOA — Number of Antennas. This field indicates the number of active antennas. Range: 1 to 24.
7	ELD3— Early Late Disable #3. 0 - Early Late Processing is enabled for DPCCH TPC Soft Symbol. 1 - Early Late Processing is disabled for DPCCH TPC Soft Symbol.
6	ELD2 — Early Late Disable #2. 0 - Early Late Processing is enabled for DPCCH FBI Soft Symbol. 1 - Early Late Processing is disabled for DPCCH FBI Soft Symbol.
5	ELD1 — Early Late Disable #1. 0 - Early Late Processing is enabled for DPCCH/PRACH TFCI Soft Symbol. 1 - Early Late Processing is disabled for DPCCH/PRACH TFCI Soft Symbol.
4	ELD0 — Early Late Disable #0. 0 - Early Late Processing is enabled for DPCCH/PRACH PILOT Soft Symbol. 1 - Early Late Processing is disabled for DPCCH/PRACH PILOT Soft Symbol.
3–0	IS — Interpolation Shift. The intermediate interpolated data is shift IS bits right. 0 - Intermediate results are not shifted. ... 15 - Intermediate results are right shifted by 15.

26.5.5.6.2 ULF Secondary Configuration Register (ULFSCR)

This register (along with *ULFGCR*) defines the general configuration parameters of the CRPE-ULF.

Offset 0x0023E00C Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R								ELD7	ELD6	ELD5	ELD4	MBS				
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				MNOC												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-291. ULF Secondary Configuration Register

Table 26-351. ULF Secondary Configuration Fields Description

Bits	Description
31–25	Reserved
24	ELD7— Early Late Disable #7. 0 - Early Late Processing is enabled for General Purpose #1 Soft Symbol 1 - Early Late Processing is disabled for General Purpose #1 Soft Symbol
23	ELD6— Early Late Disable #6. 0 - Early Late Processing is enabled for General Purpose #0 Soft Symbol 1 - Early Late Processing is disabled for General Purpose #0 Soft Symbol
22	ELD5— Early Late Disable #5. 0 - Early Late Processing is enabled for HS-DPCCH Soft Symbol 1 - Early Late Processing is disabled for HS-DPCCH Soft Symbol
21	ELD4— Early Late Disable #4. 0 - Early Late Processing is enabled for E-DPCCH Soft Symbol 1 - Early Late Processing is disabled for E-DPCCH Soft Symbol
20–16	MBS—Maximum Burst Size. This field defines the maximum burst size accesses (in 16 bytes granularity) for the CRPE-ULF MBus output results. Recommended value is 16, that is, bursts of 256 bytes. 0 - Reserved. 1 - CRPE-ULF MBus output accesses maximum burst size is 16 bytes bursts. 2 - CRPE-ULF MBus output accesses maximum burst size is 32 bytes bursts. ... 24 - CRPE-ULF MBus output accesses maximum burst size is 384 bytes bursts. 25 to 31 - Reserved.
15–13	Reserved.
12–0	MNOC - Minimum Number of Cycles in Processing Period. Should be set to 4144 (0x1030)

26.5.5.6.3 ULF Interpolation Configuration Register <x> (ULFICRx)

This set of registers define the interpolation weights of the CRPE-ULF.

Offset 0x0023E080 Access: User read/write
offset 4
x = 0...15

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	W(4x + 3)								W(4x + 2)							
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	W(4x + 1)								W(4x + 0)							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-292. ULF Interpolation Configuration Register

Table 26-352. ULF Interpolation Configuration Fields Description

Bits	Description
31–24	W(4x + 3)— Interpolation Weight #(4x + 3).
23–16	W(4x + 2)— Interpolation Weight #(4x + 2).
15–8	W(4x + 1)— Interpolation Weight #(4x + 1).
7–0	W(4x + 0)— Interpolation Weight #(4x + 0).

26.5.5.6.4 ULF Output Buffer <x> Base Configuration Register (ULFOBxBCR)

This set of registers define the base address of the output buffers of the CRPE-ULF.

Offset 0x0023E100 Access: User read/write
offset 8
x = 0...17

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

Figure 26-293. ULF Output Buffer <x> Base Configuration Register

Table 26-353. ULF Output Buffer <x> Base Configuration Fields Description

Bits	Description
31–4	BASEx — The base address of the cyclic output buffer #x. The base address must be 16 bytes aligned.
3–0	reserved

26.5.5.6.5 ULF Output Buffer <x> Attributes Configuration Register (ULFOBxACR)

This set of registers define the attributes of the output buffers of the CRPE-ULF.

Offset 0x0023E104
offset 8
x = 0...17

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R						GID			TE7	TE6	TE5	TE4	TE3	TE2	TE1	TE0
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	SIZEx															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-294. ULF Output Buffer <x> Attributes Configuration Register

Table 26-354. ULF Output Buffer <x> Attributes Configuration Fields Description

Bits	Description
31–27	Reserved
26–24	GID — Group ID. This field sets the output buffer group ID. A packet is directed to an output buffer only if the packet's GID is identical to the output buffer GID.
23	TE7 — Type Enable #7 0 - General Purpose #1 Packets is discarded by this buffer 1 - General Purpose #1 Packets is accepted by this buffer
22	TE7 — Type Enable #6 0 - General Purpose #0 Packets is discarded by this buffer 1 - General Purpose #0 Packets is accepted by this buffer
21	TE5 — Type Enable #5 0 - HS-DPCCH Packets is discarded by this buffer 1 - HS-DPCCH Packets is accepted by this buffer
20	TE4 — Type Enable #4 0 - E-DPCCH Packets is discarded by this buffer 1 - E-DPCCH Packets is accepted by this buffer
19	TE3 — Type Enable #3 0 - DPCCH TPC Packets is discarded by this buffer 1 - DPCCH TPC Packets is accepted by this buffer

Table 26-354. ULF Output Buffer <x> Attributes Configuration Fields Description

Bits	Description
18	TE2 — Type Enable #2 0 - DPCCH FBI Packets is discarded by this buffer 1 - DPCCH FBI Packets is accepted by this buffer
17	TE1 — Type Enable #1 0 - DPCCH/RACH TFCI Packets is discarded by this buffer. 1 - DPCCH/RACH TFCI Packets is accepted by this buffer.
16	TE0 — Type Enable #0 0 - DPCCH/RACH PILOT Packets is discarded by this buffer. 1 - DPCCH/RACH PILOT Packets is accepted by this buffer.
15–0	SIZEx — This field set the size of the output buffer with 16 bytes granularity. The output buffer is enabled if this field is great than 0.

26.5.5.6.6 ULF Event Status Register (ULFESR)

This register includes bits which are set due to internal CRPE-ULF events. Clearing the bits is done by writing ‘1’ to relevant bit.

Offset 0x0023E280 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												ECC	IBO	CE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-295. ULF Event Status Register

Table 26-355. ULF Event Status Fields Description

Bits	Description
31–5	Reserved
4	ECC — ECC error Indication. If set an ECC error has occur in one of the CRPE-ULF internal memories.
3 IBO	Input Buffer Overflow error event. This bit is set if ULFEMLR[IBOE] is set and IB is overflowed.
2 CE	Command Error This bit is set if ULFEMLR[CEE] is set and an internal Command Error occurs.
1–0	Reserved.

26.5.5.6.7 ULF Command FIFO Status Register (ULFCFSR)

This register describe the error attributes in case of command error indication.

Offset 0x0023E290 Access: User Read

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

Figure 26-296. ULF Command FIFO Status Register

Table 26-356. ULF Command FIFO Status Fields Description

Bits	Description
31–29	Reserved
28	CT — Command Type. This field indicates the type of the command which initiated a Command Error Event 0 - The command Type is FIC. 1 - The command Type is PCHC.
27–16	CID — Command ID. if *CT* = 0, this field indicates the FID of the FIC which initiated a Command Error Event if *CT* = 1, this field indicates the PCHID of the PCHC which initiated a Command Error Event
15–0	Reserved

26.5.5.6.8 ULF Input Buffer Status Register (ULFIBSR)

This register describe the input buffer address in case of input buffer overflow.

Offset 0x0023E294 Access: User Read

	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			ADDR													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-297. ULF Input Buffer Status Register

Table 26-357. ULF Input Buffer Status Fields Description

Bits	Description
31–14	Reserved
13–0	ADDR — This field indicates the address of the transaction which caused an Input Buffer overflow error event.

26.5.5.6.9 ULF Time Status Register (ULFTSR)

This register maintain time indexes for processing periods and sub-slot.

Offset 0x0023E298 Access: User Read

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

Figure 26-298. ULF Time Status Register

Table 26-358. ULF Time Status Fields Description

Bits	Description
31–25	Reserved
24–16	SSI — Sub Slot Index. This field indicates the index of the sub-slot currently being processed by ULF.
15–3	Reserved
2–0	PPI — Processing Period Index. This field status the index of the processing period currently being processed by ULF.

26.5.5.6.10 ULF ECC Status Register (ULFECCSR)

The status bits in the ULFECCSR register are set due to internal ECC Error in one of its internal memory modules. Clearing a bit is done by writing ‘1’ to it.

Offset 0x0023E29C Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		M14	M13		M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
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																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-299. ULF Time Status Register

Table 26-359. ULF Time Status Fields Description

Bits	Description
31	Reserved
30	M14 — Multiple ECC Error #14. Indicates ECC Error in MIB memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
29	M13 — Multiple ECC Error #13. Indicates ECC Error in PHF memory.Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
28	Reserved
27	M11 — Multiple ECC Error #11. Indicates ECC Error in PDF memory.Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
26	M10 — Multiple ECC Error #10. Indicates ECC Error in PGT memory.Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
25	M9 — Multiple ECC Error #9. Indicates ECC Error in FIT1 memory.Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
24	M8 — Multiple ECC Error #8. Indicates ECC Error in FIT0 memory.Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
23	M7 — Multiple ECC Error #7. Indicates ECC Error in FLIST1 memory.Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
22	M6 — Multiple ECC Error #6. Indicates ECC Error in FLIST0 memory.Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur

Table 26-359. ULF Time Status Fields Description

Bits	Description
21	M5 — Multiple ECC Error #5. Indicates ECC Error in FICT memory.Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
20	M4 — Multiple ECC Error #4. Indicates ECC Error in PCHT1 memory.Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
19	M3 — Multiple ECC Error #3. Indicates ECC Error in PCHT0 memory.Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
18	M2 — Multiple ECC Error #2. Indicates ECC Error in CF1 memory.Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
17	M1 — Multiple ECC Error #1. Indicates ECC Error in CF0 memory.Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
16	M0 — Multiple ECC Error #0. Indicates ECC Error in IB memory.Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
15–0	Reserved

26.5.5.7 CRPE-DL Registers Description

Table 26-360 describes the CRPE-DL registers in detail.

Table 26-360. CRPE-DL Registers Detailed Description

Address	Register	Access	Size	Reset Value	Section
CRPE-DL Chips Output Data Table					
0x00228000–0x002287FF	CDCODT - CRPE-DL Chips Output Data Table x (x=0...127)	R/W	16B	0x0000_0000	**Section 26.5.5.7.1**
Slot Format Look Up Table					
0x00228800–0x00228D9F	SFLUTx - Slot Format Look-up Table x (x=0...89)	R/W	16B	0x0000_0000	**Section 26.5.5.7.2**
Scrambling Initialization Look Up Table					
0x0023F800–0x0023F9FF	SCRILUTx - Scrambling Initialization Look-up Table x (x=0...31)	R/W	16B	0x0000_0000	**Section 26.5.5.7.3**
Configuration Registers					
0x0023F148	CDNVR - CRPE Downlink Normalization Value Register	R/W	4B	0x0000_0000	**Section 26.5.5.7.4**
Control Registers					
0x00220000–0x00221FFF	CDVAGLR0_x - CRPE Downlink Virtual Antenna Gains Control Register 0_x (x = 0...511)	R/W	8B	0x0000_0000	**Section 26.5.5.7.5**
0x00220000–0x00221FFF	CDVAGLR1_x - CRPE Downlink Virtual Antenna Gains Control Register 1_x (x = 0...511)	R/W	8B	0x0000_0000	**Section 26.5.5.7.6**

Table 26-360. CRPE-DL Registers Detailed Description (Continued)

Address	Register	Access	Size	Reset Value	Section
0x0023F000	CDBFCVLR0 - CRPE Downlink Beam Forming Coefficients Values Control Register 0	R/W	16B	0x0000_0000	**Section 26.5.5.7.7**
0x0023F010	CDBFCVLR1 - CRPE Downlink Beam Forming Coefficients Values Control Register 1	R/W	16B	0x0000_0000	**Section 26.5.5.7.8**
0x0023F020	CDBFCVLR2 - CRPE Downlink Beam Forming Coefficients Values Control Register 2	R/W	16B	0x0000_0000	**Section 26.5.5.7.9**
0x0023F030	CDBFCVLR3 - CRPE Downlink Beam Forming Coefficients Values Control Register 3	R/W	16B	0x0000_0000	**Section 26.5.5.7.10**
0x0023F040	CDBFCVLR4 - CRPE Downlink Beam Forming Coefficients Values Control Register 4	R/W	16B	0x0000_0000	**Section 26.5.5.7.11**
0x0023F050	CDBFCVLR5 - CRPE Downlink Beam Forming Coefficients Values Control Register 5	R/W	16B	0x0000_0000	**Section 26.5.5.7.12**
0x0023F060	CDBFCVLR6 - CRPE Downlink Beam Forming Coefficients Values Control Register 6	R/W	16B	0x0000_0000	**Section 26.5.5.7.13**
0x0023F070	CDBFCVLR7 - CRPE Downlink Beam Forming Coefficients Values Control Register 7	R/W	16B	0x0000_0000	**Section 26.5.5.7.14**
0x0023F080	CDSLR - CRPE Downlink Start Control Register	R/W	4B	0x0000_0000	**Section 26.5.5.7.15**
0x0023F084	CDTCLR - CRPE Downlink TPC Command Control Register	W	4B	0x0000_0000	**Section 26.5.5.7.16**
0x0023F088–0x0023F08F	Reserved				
0x0023F090	CDVAGCLR - CRPE Downlink Virtual Antennas Gain Command Control Register	R/W	4B	0x0000_0000	**Section 26.5.5.7.17**
0x0023F094	Reserved				
0x0023F098	CDGCLR - CRPE Downlink General Command Control Register	R/W	8B	0x0000_0000	**Section 26.5.5.7.18**
0x0023F0A0–0x0023F0DF	CDIPLRx - CRPE Downlink Idle Period Control Register X (x = 0...15)	R/W	4B	0x0000_0000	**Section 26.5.5.7.19**
0x0023F0E0–0x0023F0EF	CDBFCTCLRx - CRPE Downlink Beam Forming Coefficients Timing Command Control Register X (x = 0...3)	R/W	4B	0x0000_0000	**Section 26.5.5.7.20**
0x0023F0F0–0x0023F0FF	CDCCSCLRx - CRPE Downlink Combined Chips Shift Command Control Registers X (x = 0...3)	R/W	4B	0x0000_0000	**Section 26.5.5.7.21**
0x0023F10C	CDRLR - CRPE Downlink Rate Control Register	R/W	4B	0x0000_0000	**Section 26.5.5.7.22**
Status Registers					
0x0023F100	CDESR - CRPE Downlink Event Status Register	R	4B	0x0000_0000	**Section 26.5.5.7.23**
0x0023F104	CDPSSR - CRPE Downlink Processing Stage Status Register	R	4B	0x0000_0000	**Section 26.5.5.7.24**
0x0023F108	CDECCSR - CRPE Downlink ECC Status Register	R	4B	0x0000_0000	**Section 26.5.5.7.25**

26.5.5.7.1 CRPE-DL Chips Output Data Table (CDCODT)

The CRPE-DL Chips Output Data Table memory contains 4 bytes for each chip for each output antenna. There are 2 entries for each antenna at the output buffer. Each entry contains 16 chips of a single antenna. The memory consists of 16 bytes words, each word contains 4 complex output chips.

Following is the addressing order of the chips in the output buffer, starting from base address of 0x00228000:

- Base: chips 3-0, antenna 0, entry 0
- Base+16: chips 7-4, antenna 0, entry 0
- Base+2*16: chips 11-8, antenna 0, entry 0
- Base+3*16: chips 15-12, antenna 0, entry 0
- Base+4*16: chips 3-0, antenna 0, entry 1
- ...
- Base+63*16: chips 15-12, antenna 7, entry 1
- Base+64*16: chips 3-0, antenna 8, entry 0
- ...
- Base+127*16: chips 15-12, antenna 15, entry 1

Figure 26-300 describe each Output buffer entry data structure:

Offset 0x00228000 (CDCODT) Access: User read/write
offset 0x10, size 16
x = 0...127

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
R W	$CDATAI_{x,y}$															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
R W	$CDATAQ_{x,y}$															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
R W	$CDATAI_{x+1,y}$															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
R W	$CDATAQ_{x+1,y}$															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-300. CRPE-DL Chip Output Data Table

	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
R W	$CDATAI_{x+2,y}$															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
R W	$CDATAQ_{x+2,y}$															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	$CDATAI_{x+3,y}$															
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	$CDATAQ_{x+3,y}$															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-300. CRPE-DL Chip Output Data Table (Continued)

The following table describe the *CDCODT* field description:

Table 26-361. CRPE-DL Chip Output Data Table Fields Description

Bits	Description
127-112	$CDATAI_{x,y}$ - Chip Data Real part of complex chip x of output antenna y.
111-96	$CDATAQ_{x,y}$ - Chip Data Imaginary part of complex chip x of output antenna y.
95-80	$CDATAI_{x+1,y}$ - Chip Data Real part of complex chip x+1 of output antenna y.
79-64	$CDATAQ_{x+1,y}$ - Chip Data Imaginary part of complex chip x+1 of output antenna y.
63-48	$CDATAI_{x+2,y}$ - Chip Data Real part of complex chip x+2 of output antenna y.
47-32	$CDATAQ_{x+2,y}$ - Chip Data Imaginary part of complex chip x+2 of output antenna y.
31-16	$CDATAI_{x+3,y}$ - Chip Data Real part of complex chip x+3 of output antenna y.
15-0	$CDATAQ_{x+3,y}$ - Chip Data Imaginary part of complex chip x+3 of output antenna y.

26.5.5.7.2 CRPE-DL Slot Format Look-Up Table (SFLUT)

This memory table contains 90 entries, each specifying a unique slot format. The information in this table defines the distribution of input symbols to up to 5 different fields in the slot. In addition, one of the fields is identified as TPC field. Each field is associated with one of 4

programable gain factors. Gain operation and TPC override require the knowledge of slot configuration.

Supported slot formats that are associated with Spreading Factor 4 include number of symbols that is divisible by 4 at each field of the slot. **Figure 26-301** describe each SFLUT entry data structure:

Offset 0x00228800 (SFLUT) Access: User read/write
offset 0x10, size 16
x = 0...89

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
R							GUF_x		$FFTGF_x$		$FRTGF_x$		$TRDGF_x$		SND GF_x	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
R	SND GF_x		$FSTGF_x$		$TPCFN_x$				$FRTFS_x$							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 26-301. CRPE-DL Slot Format Look-up Table

The following table describe the SFLUT field description:

Table 26-362. CRPE-DL Slot Format Look-up Table Fields Description

Bits	Description
127–58	Reserved.
57–55	GUF_x - Gain Update Field. This field defines at which field in the slot gain factors should be updated. Gain update field must contain at least one symbol. 0 - The updated gain factors take affect at the beginning of the first field in the slot. 1 - The updated gain factors take affect at the beginning of the second field in the slot. 2 - The updated gain factors take affect at the beginning of the third field in the slot. 3 - The updated gain factors take affect at the beginning of fourth field in the slot. 4 - The updated gain factors take affect at the beginning of fifth field in the slot. 5-7 - reserved.
54–53	$FFTGF_x$ - Fifth Field Gain Factor. This field defines which gain factor is associated with fifth field. The gain factors values are programmed at Virtual Antenna Gains Control Registers (*CDVAGLR0*). 0 - fifth field is associated with *CDVAGLR0[FSTRG]* 1 - fifth field is associated with *CDVAGLR0[SNDRG]* 2 - fifth field is associated with *CDVAGLR0[TRDRG]* 3 - fifth field is associated with *CDVAGLR0[FRTRG]*
52–51	$FRTGF_x$ - Fourth Field Gain Factor. This field defines which gain factor is associated with fourth field. The gain factors values are programmed at Virtual Antenna Gains Control Registers (*CDVAGLR0*). 0 - fourth field is associated with *CDVAGLR0[FSTRG]* 1 - fourth field is associated with *CDVAGLR0[SNDRG]* 2 - fourth field is associated with *CDVAGLR0[TRDRG]* 3 - fourth field is associated with *CDVAGLR0[FRTRG]*
50–49	$TRDGF_x$ - Third Field Gain Factor. This field defines which gain factor is associated with third field. The gain factors values are programmed at Virtual Antenna Gains Control Registers (*CDVAGLR0*). 0 - third field is associated with *CDVAGLR0[FSTRG]* 1 - third field is associated with *CDVAGLR0[SNDRG]* 2 - third field is associated with *CDVAGLR0[TRDRG]* 3 - third field is associated with *CDVAGLR0[FRTRG]*
48–47	$SNDGF_x$ - Second Field Gain Factor. This field defines which gain factor is associated with second field. The gain factors values are programmed at Virtual Antenna Gains Control Registers (*CDVAGLR0*). 0 - second field is associated with *CDVAGLR0[FSTRG]* 1 - second field is associated with *CDVAGLR0[SNDRG]* 2 - second field is associated with *CDVAGLR0[TRDRG]* 3 - second field is associated with *CDVAGLR0[FRTRG]*
46–45	$FSTGF_x$ - First Field Gain Factor. This field defines which gain factor is associated with first field. The gain factors values are programmed at Virtual Antenna Gains Control Registers (*CDVAGLR0*). 0 - first field is associated with *CDVAGLR0[FSTRG]* 1 - first field is associated with *CDVAGLR0[SNDRG]* 2 - first field is associated with *CDVAGLR0[TRDRG]* 3 - first field is associated with *CDVAGLR0[FRTRG]*

Table 26-362. CRPE-DL Slot Format Look-up Table Fields Description

Bits	Description
44–42	$TPCFN_x$ - TPC Field Number. 0 - TPC is the first field. 1 - TPC is the second field. 2 - TPC is the third field. 3 - TPC is the fourth field. 4 - TPC is the fifth field. 5 - TPC field is not present. Range: 0-5.
41–32	$FRTFS_x$ - Fourth Field Separator This bit indicates the offset of the first symbol of fifth field in the slot. Input symbols with offset higher or equal to this offset belong to the fifth field. Input symbols with offset lower than this offset belong to one of the following fields: first, second, third or fourth field. Range: 0-640. 640 is out of the slot range, used for non existent field separator.
31–30	Reserved
29–20	$TRDFS_x$ - Third Field Separator This bit indicates the offset of the first symbol of fourth field in the slot. Input symbols with offset higher or equal to this offset belong to one of the following fields: fourth or fifth field. Input symbols with offset lower than this offset belong to one of the following fields: first, second or third field. Range: 0-640. 640 is out of the slot range, used for non existent field separator.
19–10	$SNDFS_x$ - Second Field Separator This bit indicates the offset of the first symbol of third field in the slot. Input symbols with offset higher or equal to this offset belong to one of the following fields: third, fourth or fifth field Input symbols with offset lower than this offset belong to one of the following fields: first or second field. Range: 0-640. 640 is out of the slot range, used for non existent field separator.
9–0	$FSTFS_x$ - First Field Separator This bit indicates the offset of the first symbol of second field in the slot. Input symbols with offset higher or equal to this offset belong to one of the following fields: second, third, fourth or fifth field. Input symbols with offset lower than this offset belong to the first field. Range: 0-640. 640 is out of the slot range, used for non existent field separator.

26.5.5.7.3 CRPE-DL Scrambling Initialization Look-Up Table (SCRILUT)

This memory table contains 32 entries, each specifying a unique scrambling code (ranging from 0 to 8191). **Figure 26-302** describe each *SCRILUT* entry data structure:

Offset 0x0023F800 (SCRILUT)
offset 0x10, size 16
x = 0...31

Access: User read/write

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	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				SN_x												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-302. CRPE-DL Scrambling Initialization Look-up Table

The following table describe the *SCRILUT* field description:

Table 26-363. CRPE-DL Scrambling Initialization Look-up Table Fields Description

Bits	Description
127–13	Reserved.
12–0	SN_x - Scrambling Number Selection n selection for scrambling code $S_{dl,n}$, correlating with channel's non-compressed mode. Range: 0-8191.

26.5.5.7.4 CRPE-DL Normalization Value Configuration Register (CDNVCR)

This is the register defines the value of additional real gain multiplication factor. This additional gain operation is performed only if *CDNVCR[NVEN]*=1 and only if both: real part of the complex gain and imaginary part of the complex gain (see *CDVAGCLR1*) are non-zero values.

Offset 0x0023F148 Access: User read/write

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

Figure 26-303. CRPE-DL Normalization Value Configuration Register

Table 26-364. CRPE-DL Normalization Value Fields Description

Bits	Description
31–17	Reserved.
16	NVEN — Normalization Value Enable 0 - Normalization value is ignored. 1 - Normalization value is used if I Gain and Q Gain are both non-zero values.
15–0	NV — Normalization Value. Signed (1q15) normalization value used as conditional multiplication factor. Reset Value: 16'h5A82.

26.5.5.7.5 CRPE-DL Virtual Antenna Gains Control Register 0 (CDVAGLR0)

This 8 bytes register is used for updating virtual antennas real gain values for the channel and the timing specified in the *CDVAGCLR* register.

Offset 0x00220000
offset 0x10, size 8
x = 0...511

Access: User read/write

	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
R W	FRTRG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
R W	TRDRG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 26-304. CRPE-DL Virtual Antenna Gains Control Register 0

Table 26-365. CRPE-DL Virtual Antenna Gains 0 Fields Description

Bits	Description
63–48	FRTRG — Fourth Real Gain for virtual antennas. 16 bit field specifying the value of the **fourth** real gain used for virtual antennas. This field is a shadow field. It is becoming active at the beginning of a selected field as pointed by GUF at slot format LUT, and at a processing slot as according to GUS and GUN fields in CDVAGCLR. In addition, this field is becoming active at the beginning of the first slot of the channel.
47–32	TRDRG — Third Real Gain for virtual antennas. 16 bit field specifying the value of the **third** real gain used for virtual antennas. This field is a shadow field. It is becoming active at the beginning of a selected field as pointed by GUF at slot format LUT, and at a processing slot as according to GUS and GUN fields in CDVAGCLR. In addition, this field is becoming active at the beginning of the first slot of the channel.
31–16	SNDRG — Second Real Gain for virtual antennas. 16 bit field specifying the value of the **second** real gain used for virtual antennas. This field is a shadow field. It is becoming active at the beginning of a selected field as pointed by GUF at slot format LUT, and at a processing slot as according to GUS and GUN fields in CDVAGCLR. In addition, this field is becoming active at the beginning of the first slot of the channel.
15–0	FSTRG — First Real Gain for virtual antennas. 16 bit field specifying the value of the **first** real gain used for virtual antennas. This field is a shadow field. It is becoming active at the beginning of a selected field as pointed by GUF at slot format LUT, and at a processing slot as according to GUS and GUN fields in CDVAGCLR. In addition, this field is becoming active at the beginning of the first slot of the channel.

26.5.5.7.6 CRPE-DL Virtual Antenna Gains Control Register 1 (CDVAGLR1)

This 8 bytes register is used for updating virtual antennas complex gain values for the channel and the timing specified in the *CDVAGCLR* register.

Offset 0x00220008
offset 0x10, size 8
x = 0...511

Access: User read/write

	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	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									IGB		QGB		IGA		QGA	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-305. CRPE-DL Virtual Antenna Gains Control Register 1

Table 26-366. CRPE-DL Virtual Antenna Gains 1 Fields Description

Bits	Description
63–8	Reserved.
7–6	IGB — I Gain for virtual antenna B. 2 bit field specifying the real part of the complex gain used for virtual antenna B. This field is a shadow field. It is becoming active at the beginning of a selected field as pointed by *GUF* at slot format LUT, and at a processing slot as according to *GUS* and *GUN* fields in *CDVAGCLR*. In addition, this field is becoming active at the beginning of the first slot of the channel. 00 - signed 0 01 - signed 1 11 - signed -1 10 - reserved
5–4	QGB — Q Gain for virtual antenna B. 2 bit field specifying the imaginary part of the complex gain used for virtual antenna B. This field is a shadow field. It is becoming active at the beginning of a selected field as pointed by *GUF* at slot format *LUT*, and at a processing slot as according to *GUS* and *GUN* fields in *CDVAGCLR*. In addition, this field is becoming active at the beginning of the first slot of the channel. 00 - signed 0 01 - signed 1 11 - signed -1 10 - reserved

Table 26-366. CRPE-DL Virtual Antenna Gains 1 Fields Description (Continued)

Bits	Description
3–2	IGA — I Gain for virtual antenna A. 2 bit field specifying the real part of the complex gain used for virtual antenna A. This field is a shadow field. It is becoming active at the beginning of a selected field as pointed by *GUF* at slot format LUT, and at a processing slot as according to *GUS* and *GUN* fields in *CDVAGCLR*. In addition, this field is becoming active at the beginning of the first slot of the channel. 00 - signed 0 01 - signed 1 11 - signed -1 10 - reserved
1–0	QGA — Q Gain for virtual antenna A. 2 bit field specifying the imaginary part of the complex gain used for virtual antenna A. This field is a shadow field. It is becoming active at the beginning of a selected field as pointed by GUF at slot format LUT, and at a processing slot as according to *GUS* and *GUN* fields in *CDVAGCLR*. In addition, this field is becoming active at the beginning of the first slot of the channel. 00 - signed 0 01 - signed 1 11 - signed -1 10 - reserved

26.5.5.7.7 CRPE-DL Beam Forming Coefficients Values Control Register 0 (CDBFCVLR0)

This register is used for configuring the beam forming matrix elements of following virtual antennas and beam antennas:

virtual antenna 1, beam antennas 0-3; virtual antenna 0, beam antennas 0-3.

Offset 0x0023F000 Access: User read/write
size 16

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
R W	$BF_{1,3}I$								$BF_{1,3}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
R W	$BF_{1,2}I$								$BF_{1,2}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
R W	$BF_{1,1}I$								$BF_{1,1}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
R W	$BF_{1,0}I$								$BF_{1,0}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-306. CRPE-DL Beam Forming 0 Control Register

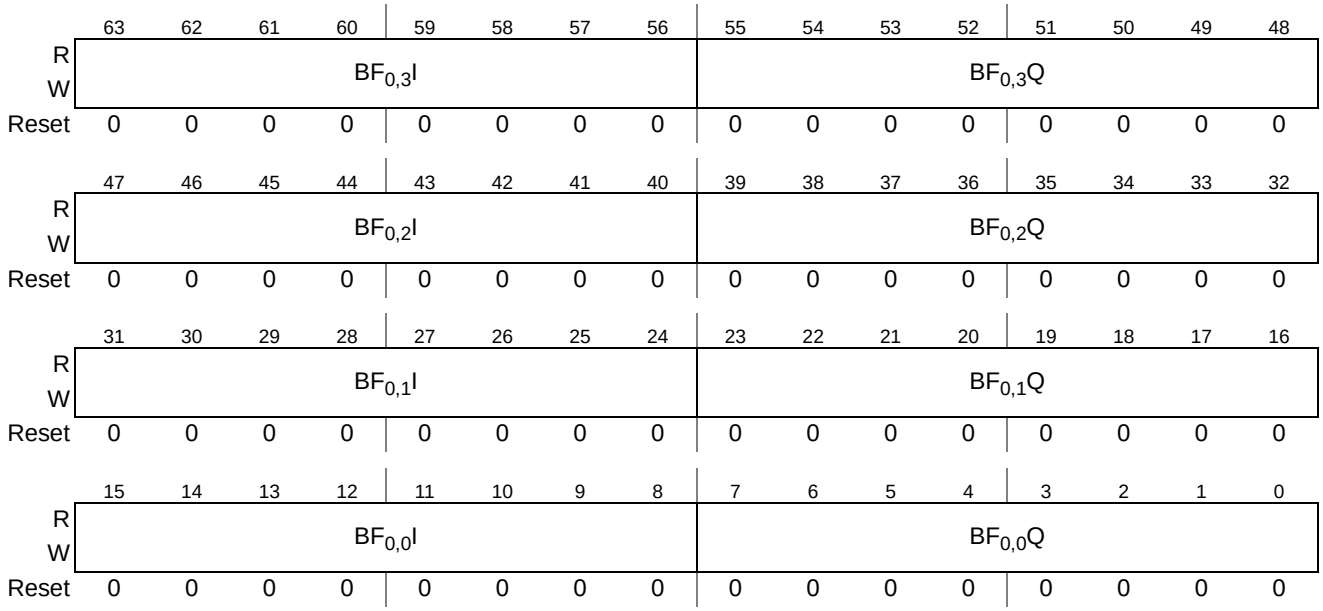


Figure 26-306. CRPE-DL Beam Forming 0 Control Register (Continued)

Table 26-367. CRPE-DL Beam Forming 0 Fields Description

Bits	Description
127–120	$BF_{1,3}I$ — Beam Forming 1,3 matrix I. This field is the real multiplication factor of 1,3 element of the Beam Forming Matrix, correlating with virtual antenna 1 and beam antenna 3.
119–112	$BF_{1,3}Q$ — Beam Forming 1,3 matrix Q. This field is the imaginary multiplication factor of 1,3 element of the Beam Forming Matrix, correlating with virtual antenna 1 and beam antenna 3.
111–104	$BF_{1,2}I$ — Beam Forming 1,2 matrix I. This field is the real multiplication factor of 1,2 element of the Beam Forming Matrix, correlating with virtual antenna 1 and beam antenna 2.
103–96	$BF_{1,2}Q$ — Beam Forming 1,2 matrix Q. This field is the imaginary multiplication factor of 1,2 element of the Beam Forming Matrix, correlating with virtual antenna 1 and beam antenna 2.
95–88	$BF_{1,1}I$ — Beam Forming 1,1 matrix I. This field is the real multiplication factor of 1,1 element of the Beam Forming Matrix, correlating with virtual antenna 1 and beam antenna 1.
87–80	$BF_{1,1}Q$ — Beam Forming 1,1 matrix Q. This field is the imaginary multiplication factor of 1,1 element of the Beam Forming Matrix, correlating with virtual antenna 1 and beam antenna 1.
79–72	$BF_{1,0}I$ — Beam Forming 1,0 matrix I. This field is the real multiplication factor of 1,0 element of the Beam Forming Matrix, correlating with virtual antenna 1 and beam antenna 0.
71–64	$BF_{1,0}Q$ — Beam Forming 1,0 matrix Q. This field is the imaginary multiplication factor of 1,0 element of the Beam Forming Matrix, correlating with virtual antenna 1 and beam antenna 0.
63–56	$BF_{0,3}I$ — Beam Forming 0,3 matrix I. This field is the real multiplication factor of 0,3 element of the Beam Forming Matrix, correlating with virtual antenna 0 and beam antenna 3.
55–48	$BF_{0,3}Q$ — Beam Forming 0,3 matrix Q. This field is the imaginary multiplication factor of 0,3 element of the Beam Forming Matrix, correlating with virtual antenna 0 and beam antenna 3.
47–40	$BF_{0,2}I$ — Beam Forming 0,2 matrix I. This field is the real multiplication factor of 0,2 element of the Beam Forming Matrix, correlating with virtual antenna 0 and beam antenna 2.
39–32	$BF_{0,2}Q$ — Beam Forming 0,2 matrix Q. This field is the imaginary multiplication factor of 0,2 element of the Beam Forming Matrix, correlating with virtual antenna 0 and beam antenna 2.

Table 26-367. CRPE-DL Beam Forming 0 Fields Description

Bits	Description
31–24	$BF_{0,1}I$ — Beam Forming 0,1 matrix I. This field is the real multiplication factor of 0,1 element of the Beam Forming Matrix, correlating with virtual antenna 0 and beam antenna 1.
23–16	$BF_{0,1}Q$ — Beam Forming 0,1 matrix Q. This field is the imaginary multiplication factor of 0,1 element of the Beam Forming Matrix, correlating with virtual antenna 0 and beam antenna 1.
15–8	$BF_{0,0}I$ — Beam Forming 0,0 matrix I. This field is the real multiplication factor of 0,0 element of the Beam Forming Matrix, correlating with virtual antenna 0 and beam antenna 0.
7–0	$BF_{0,0}Q$ — Beam Forming 0,0 matrix Q. This field is the imaginary multiplication factor of 0,0 element of the Beam Forming Matrix, correlating with virtual antenna 0 and beam antenna 0.

26.5.5.7.8 CRPE-DL Beam Forming Coefficients Values Control Register 1 (CDBFCVLR1)

This register is used for configuring the beam forming matrix elements of following virtual antennas and beam antennas:

virtual antenna 3, beam antennas 0-3; virtual antenna 2, beam antennas 0-3.

Offset 0x0023F010 size 16 — Access: User read/write

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
R W	$BF_{3,3}I$								$BF_{3,3}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
R W	$BF_{3,2}I$								$BF_{3,2}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
R W	$BF_{3,1}I$								$BF_{3,1}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
R W	$BF_{3,0}I$								$BF_{3,0}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-307. CRPE-DL Beam Forming 1 Control Register

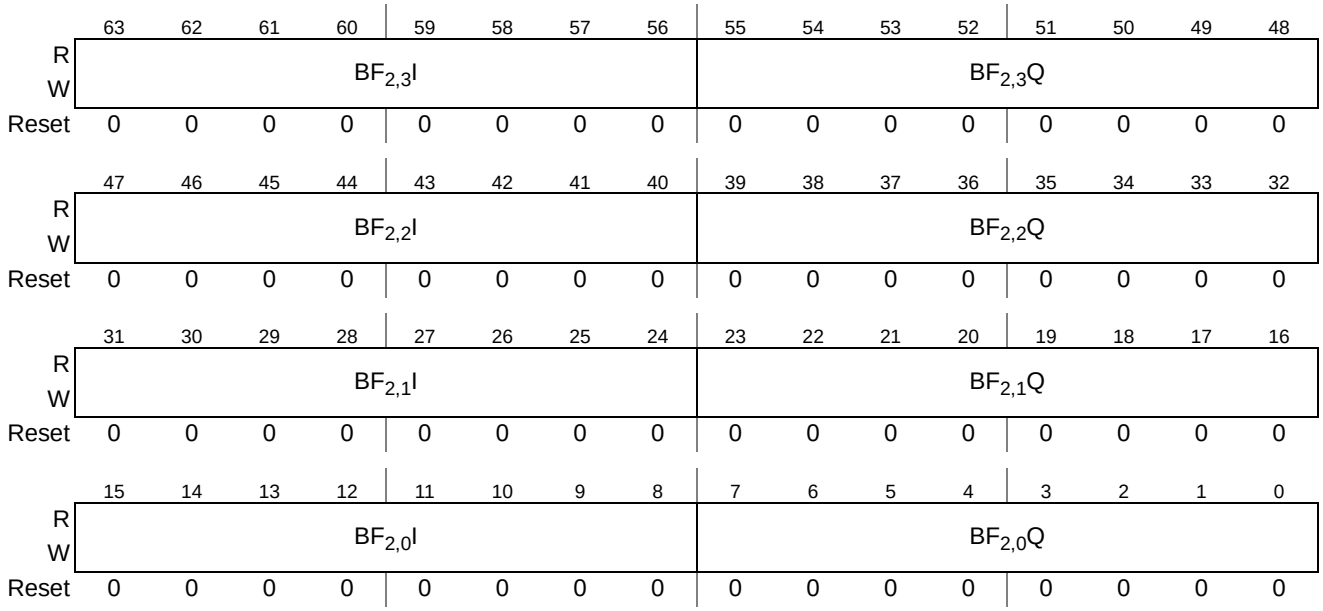


Figure 26-307. CRPE-DL Beam Forming 1 Control Register (Continued)

Table 26-368. CRPE-DL Beam Forming 1 Fields Description

Bits	Description
127–120	$BF_{3,3}I$ — Beam Forming 3,3 matrix I. This field is the real multiplication factor of 3,3 element of the Beam Forming Matrix, correlating with virtual antenna 3 and beam antenna 3.
119–112	$BF_{3,3}Q$ — Beam Forming 3,3 matrix Q. This field is the imaginary multiplication factor of 3,3 element of the Beam Forming Matrix, correlating with virtual antenna 3 and beam antenna 3.
111–104	$BF_{3,2}I$ — Beam Forming 3,2 matrix I. This field is the real multiplication factor of 3,2 element of the Beam Forming Matrix, correlating with virtual antenna 3 and beam antenna 2.
103–96	$BF_{3,2}Q$ — Beam Forming 3,2 matrix Q. This field is the imaginary multiplication factor of 3,2 element of the Beam Forming Matrix, correlating with virtual antenna 3 and beam antenna 2.
95–88	$BF_{3,1}I$ — Beam Forming 3,1 matrix I. This field is the real multiplication factor of 3,1 element of the Beam Forming Matrix, correlating with virtual antenna 3 and beam antenna 1.
87–80	$BF_{3,1}Q$ — Beam Forming 3,1 matrix Q. This field is the imaginary multiplication factor of 3,1 element of the Beam Forming Matrix, correlating with virtual antenna 3 and beam antenna 1.
79–72	$BF_{3,0}I$ — Beam Forming 3,0 matrix I. This field is the real multiplication factor of 3,0 element of the Beam Forming Matrix, correlating with virtual antenna 3 and beam antenna 0.
71–64	$BF_{3,0}Q$ — Beam Forming 3,0 matrix Q. This field is the imaginary multiplication factor of 3,0 element of the Beam Forming Matrix, correlating with virtual antenna 3 and beam antenna 0.
63–56	$BF_{2,3}I$ — Beam Forming 2,3 matrix I. This field is the real multiplication factor of 2,3 element of the Beam Forming Matrix, correlating with virtual antenna 2 and beam antenna 3.
55–48	$BF_{2,3}Q$ — Beam Forming 2,3 matrix Q. This field is the imaginary multiplication factor of 2,3 element of the Beam Forming Matrix, correlating with virtual antenna 2 and beam antenna 3.
47–40	$BF_{2,2}I$ — Beam Forming 2,2 matrix I. This field is the real multiplication factor of 2,2 element of the Beam Forming Matrix, correlating with virtual antenna 2 and beam antenna 2.
39–32	$BF_{2,2}Q$ — Beam Forming 2,2 matrix Q. This field is the imaginary multiplication factor of 2,2 element of the Beam Forming Matrix, correlating with virtual antenna 2 and beam antenna 2.

Table 26-368. CRPE-DL Beam Forming 1 Fields Description (Continued)

Bits	Description
31–24	$BF_{2,1}I$ — Beam Forming 2,1 matrix I. This field is the real multiplication factor of 2,1 element of the Beam Forming Matrix, correlating with virtual antenna 2 and beam antenna 1.
23–16	$BF_{2,1}Q$ — Beam Forming 2,1 matrix Q. This field is the imaginary multiplication factor of 2,1 element of the Beam Forming Matrix, correlating with virtual antenna 2 and beam antenna 1.
15–8	$BF_{2,0}I$ — Beam Forming 2,0 matrix I. This field is the real multiplication factor of 2,0 element of the Beam Forming Matrix, correlating with virtual antenna 2 and beam antenna 0.
7–0	$BF_{2,0}Q$ — Beam Forming 2,0 matrix Q. This field is the imaginary multiplication factor of 2,0 element of the Beam Forming Matrix, correlating with virtual antenna 2 and beam antenna 0.

26.5.5.7.9 CRPE-DL Beam Forming Coefficients Values Control Register 2 (CDBFCVLR2)

This register is used for configuring the beam forming matrix elements of following virtual antennas and beam antennas:

virtual antenna 5, beam antennas 4–7; virtual antenna 4, beam antennas 4–7.

Offset 0x0023F020 Access: User read/write
size 16

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
R W	$BF_{5,7}I$								$BF_{5,7}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
R W	$BF_{5,6}I$								$BF_{5,6}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
R W	$BF_{5,5}I$								$BF_{5,5}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
R W	$BF_{5,4}I$								$BF_{5,4}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-308. CRPE-DL Beam Forming 2 Control Register

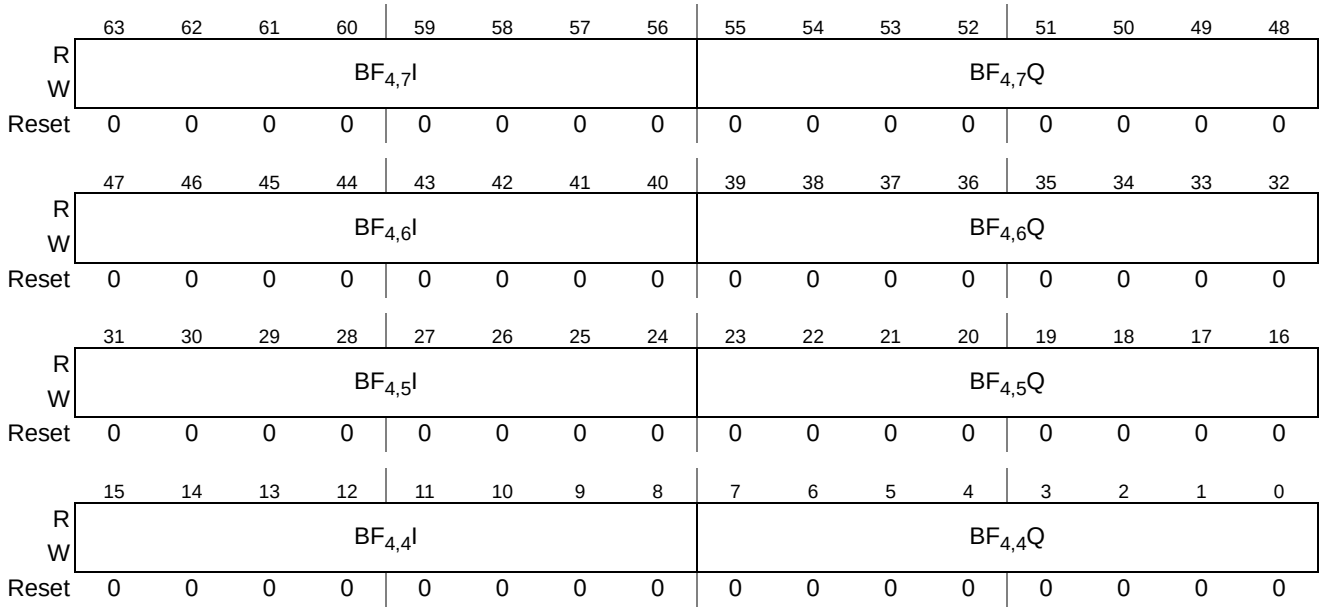


Figure 26-308. CRPE-DL Beam Forming 2 Control Register (Continued)

Table 26-369. CRPE-DL Beam Forming 2 Fields Description

Bits	Description
127–120	$BF_{5,7}I$ — Beam Forming 5,7 matrix I. This field is the real multiplication factor of 5,7 element of the Beam Forming Matrix, correlating with virtual antenna 5 and beam antenna 7.
119–112	$BF_{5,7}Q$ — Beam Forming 5,7 matrix Q. This field is the imaginary multiplication factor of 5,7 element of the Beam Forming Matrix, correlating with virtual antenna 5 and beam antenna 7.
111–104	$BF_{5,6}I$ — Beam Forming 5,6 matrix I. This field is the real multiplication factor of 5,6 element of the Beam Forming Matrix, correlating with virtual antenna 5 and beam antenna 6.
103–96	$BF_{5,6}Q$ — Beam Forming 5,6 matrix Q. This field is the imaginary multiplication factor of 5,6 element of the Beam Forming Matrix, correlating with virtual antenna 5 and beam antenna 6.
95–88	$BF_{5,5}I$ — Beam Forming 5,5 matrix I. This field is the real multiplication factor of 5,5 element of the Beam Forming Matrix, correlating with virtual antenna 5 and beam antenna 5.
87–80	$BF_{5,5}Q$ — Beam Forming 5,5 matrix Q. This field is the imaginary multiplication factor of 5,5 element of the Beam Forming Matrix, correlating with virtual antenna 5 and beam antenna 5.
79–72	$BF_{5,4}I$ — Beam Forming 5,4 matrix I. This field is the real multiplication factor of 5,4 element of the Beam Forming Matrix, correlating with virtual antenna 5 and beam antenna 4.
71–64	$BF_{5,4}Q$ — Beam Forming 5,4 matrix Q. This field is the imaginary multiplication factor of 5,4 element of the Beam Forming Matrix, correlating with virtual antenna 5 and beam antenna 4.
63–56	$BF_{4,7}I$ — Beam Forming 4,7 matrix I. This field is the real multiplication factor of 4,7 element of the Beam Forming Matrix, correlating with virtual antenna 4 and beam antenna 7.
55–48	$BF_{4,7}Q$ — Beam Forming 4,7 matrix Q. This field is the imaginary multiplication factor of 4,7 element of the Beam Forming Matrix, correlating with virtual antenna 4 and beam antenna 7.
47–40	$BF_{4,6}I$ — Beam Forming 4,6 matrix I. This field is the real multiplication factor of 4,6 element of the Beam Forming Matrix, correlating with virtual antenna 4 and beam antenna 6.
39–32	$BF_{4,6}Q$ — Beam Forming 4,6 matrix Q. This field is the imaginary multiplication factor of 4,6 element of the Beam Forming Matrix, correlating with virtual antenna 4 and beam antenna 6.

Table 26-369. CRPE-DL Beam Forming 2 Fields Description (Continued)

Bits	Description
31–24	$BF_{4,5}I$ — Beam Forming 4,5 matrix I. This field is the real multiplication factor of 4,5 element of the Beam Forming Matrix, correlating with virtual antenna 4 and beam antenna 5.
23–16	$BF_{4,5}Q$ — Beam Forming 4,5 matrix Q. This field is the imaginary multiplication factor of 4,5 element of the Beam Forming Matrix, correlating with virtual antenna 4 and beam antenna 5.
15–8	$BF_{4,4}I$ — Beam Forming 4,4 matrix I. This field is the real multiplication factor of 4,4 element of the Beam Forming Matrix, correlating with virtual antenna 4 and beam antenna 4.
7–0	$BF_{4,4}Q$ — Beam Forming 4,4 matrix Q. This field is the imaginary multiplication factor of 4,4 element of the Beam Forming Matrix, correlating with virtual antenna 4 and beam antenna 4.

26.5.5.7.10 CRPE-DL Beam Forming Coefficients Values Control Register 3 (CDBFCVLR3)

This register is used for configuring the beam forming matrix elements of following virtual antennas and beam antennas:

virtual antenna 7, beam antennas 4–7; virtual antenna 6, beam antennas 4–7.

Offset 0x0023F030 size 16 — Access: User read/write

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
R W	$BF_{7,7}I$								$BF_{7,7}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
R W	$BF_{7,6}I$								$BF_{7,6}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
R W	$BF_{7,5}I$								$BF_{7,5}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
R W	$BF_{7,4}I$								$BF_{7,4}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-309. CRPE-DL Beam Forming 3 Control Register

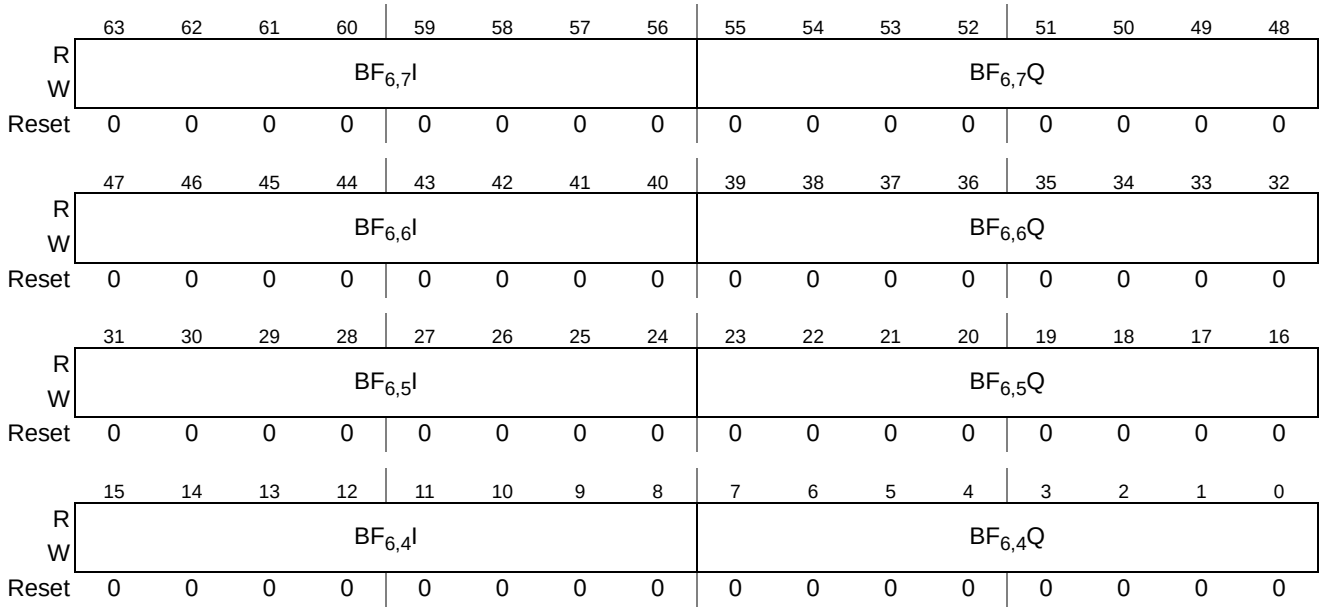


Figure 26-309. CRPE-DL Beam Forming 3 Control Register (Continued)

Table 26-370. CRPE-DL Beam Forming 3 Fields Description

Bits	Description
127–120	$BF_{7,7}I$ — Beam Forming 7,7 matrix I. This field is the real multiplication factor of 7,7 element of the Beam Forming Matrix, correlating with virtual antenna 7 and beam antenna 7.
119–112	$BF_{7,7}Q$ — Beam Forming 7,7 matrix Q. This field is the imaginary multiplication factor of 7,7 element of the Beam Forming Matrix, correlating with virtual antenna 7 and beam antenna 7.
111–104	$BF_{7,6}I$ — Beam Forming 7,6 matrix I. This field is the real multiplication factor of 7,6 element of the Beam Forming Matrix, correlating with virtual antenna 7 and beam antenna 6.
103–96	$BF_{7,6}Q$ — Beam Forming 7,6 matrix Q. This field is the imaginary multiplication factor of 7,6 element of the Beam Forming Matrix, correlating with virtual antenna 7 and beam antenna 6.
95–88	$BF_{7,5}I$ — Beam Forming 7,5 matrix I. This field is the real multiplication factor of 7,5 element of the Beam Forming Matrix, correlating with virtual antenna 7 and beam antenna 5.
87–80	$BF_{7,5}Q$ — Beam Forming 7,5 matrix Q. This field is the imaginary multiplication factor of 7,5 element of the Beam Forming Matrix, correlating with virtual antenna 7 and beam antenna 5.
79–72	$BF_{7,4}I$ — Beam Forming 7,4 matrix I. This field is the real multiplication factor of 7,4 element of the Beam Forming Matrix, correlating with virtual antenna 7 and beam antenna 4.
71–64	$BF_{7,4}Q$ — Beam Forming 7,4 matrix Q. This field is the imaginary multiplication factor of 7,4 element of the Beam Forming Matrix, correlating with virtual antenna 7 and beam antenna 4.
63–56	$BF_{6,7}I$ — Beam Forming 6,7 matrix I. This field is the real multiplication factor of 6,7 element of the Beam Forming Matrix, correlating with virtual antenna 6 and beam antenna 7.
55–48	$BF_{6,7}Q$ — Beam Forming 6,7 matrix Q. This field is the imaginary multiplication factor of 6,7 element of the Beam Forming Matrix, correlating with virtual antenna 6 and beam antenna 7.
47–40	$BF_{6,6}I$ — Beam Forming 6,6 matrix I. This field is the real multiplication factor of 6,6 element of the Beam Forming Matrix, correlating with virtual antenna 6 and beam antenna 6.
39–32	$BF_{6,6}Q$ — Beam Forming 6,6 matrix Q. This field is the imaginary multiplication factor of 6,6 element of the Beam Forming Matrix, correlating with virtual antenna 6 and beam antenna 6.

Table 26-370. CRPE-DL Beam Forming 3 Fields Description (Continued)

Bits	Description
31–24	$BF_{6,5}I$ — Beam Forming 6,5 matrix I. This field is the real multiplication factor of 6,5 element of the Beam Forming Matrix, correlating with virtual antenna 6 and beam antenna 5.
23–16	$BF_{6,5}Q$ — Beam Forming 6,5 matrix Q. This field is the imaginary multiplication factor of 6,5 element of the Beam Forming Matrix, correlating with virtual antenna 6 and beam antenna 5.
15–8	$BF_{6,4}I$ — Beam Forming 6,4 matrix I. This field is the real multiplication factor of 6,4 element of the Beam Forming Matrix, correlating with virtual antenna 6 and beam antenna 4.
7–0	$BF_{6,4}Q$ — Beam Forming 6,4 matrix Q. This field is the imaginary multiplication factor of 6,4 element of the Beam Forming Matrix, correlating with virtual antenna 6 and beam antenna 4.

26.5.5.7.11 CRPE-DL Beam Forming Coefficients Values Control Register 4 (CDBFCVLR4)

This register is used for configuring the beam forming matrix elements of following virtual antennas and beam antennas:

virtual antenna 9, beam antennas 8-11; virtual antenna 8, beam antennas 8-11.

Offset 0x0023F040 size 16 — Access: User read/write

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
R W	$BF_{9,11}I$								$BF_{9,11}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
R W	$BF_{9,10}I$								$BF_{9,10}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
R W	$BF_{9,9}I$								$BF_{9,9}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
R W	$BF_{9,8}I$								$BF_{9,8}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-310. CRPE-DL Beam Forming 4 Control Register

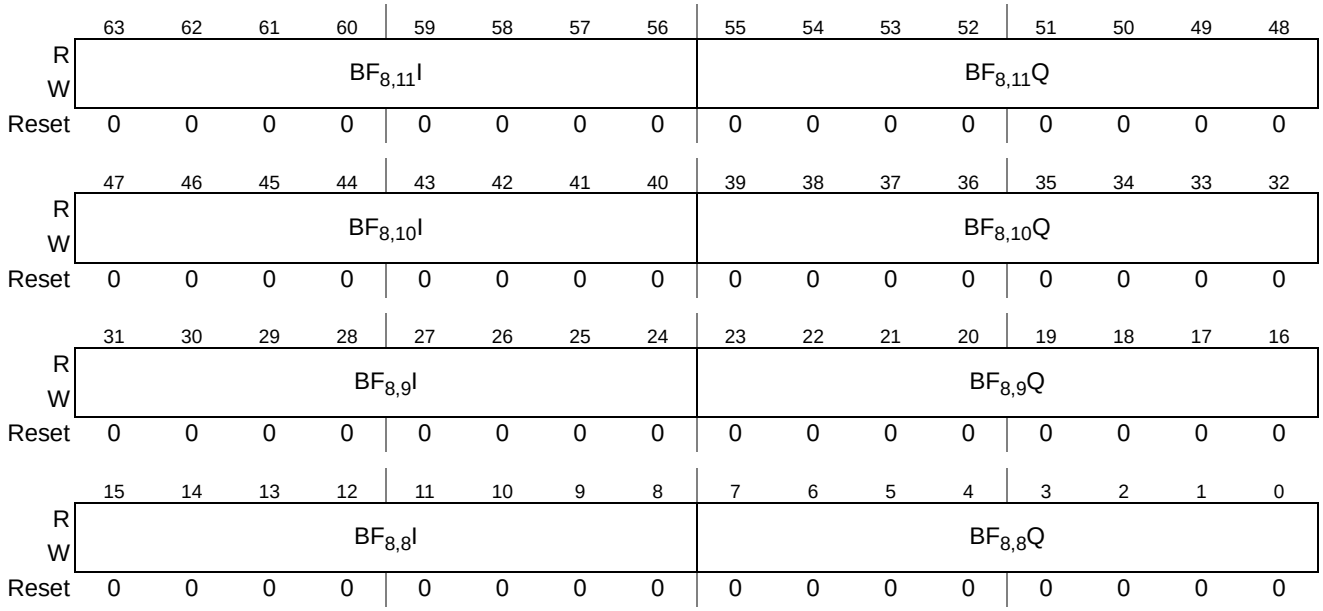


Figure 26-310. CRPE-DL Beam Forming 4 Control Register (Continued)

Table 26-371. CRPE-DL Beam Forming 4 Fields Description

Bits	Description
127–120	$BF_{9,11}I$ — Beam Forming 9,11 matrix I. This field is the real multiplication factor of 9,11 element of the Beam Forming Matrix, correlating with virtual antenna 9 and beam antenna 11.
119–112	$BF_{9,11}Q$ — Beam Forming 9,11 matrix Q. This field is the imaginary multiplication factor of 9,11 element of the Beam Forming Matrix, correlating with virtual antenna 9 and beam antenna 11.
111–104	$BF_{9,10}I$ — Beam Forming 9,10 matrix I. This field is the real multiplication factor of 9,10 element of the Beam Forming Matrix, correlating with virtual antenna 9 and beam antenna 10.
103–96	$BF_{9,10}Q$ — Beam Forming 9,10 matrix Q. This field is the imaginary multiplication factor of 9,10 element of the Beam Forming Matrix, correlating with virtual antenna 9 and beam antenna 10.
95–88	$BF_{9,9}I$ — Beam Forming 9,9 matrix I. This field is the real multiplication factor of 9,9 element of the Beam Forming Matrix, correlating with virtual antenna 9 and beam antenna 9.
87–80	$BF_{9,9}Q$ — Beam Forming 9,9 matrix Q. This field is the imaginary multiplication factor of 9,9 element of the Beam Forming Matrix, correlating with virtual antenna 9 and beam antenna 9.
79–72	$BF_{9,8}I$ — Beam Forming 9,8 matrix I. This field is the real multiplication factor of 9,8 element of the Beam Forming Matrix, correlating with virtual antenna 9 and beam antenna 8.
71–64	$BF_{9,8}Q$ — Beam Forming 9,8 matrix Q. This field is the imaginary multiplication factor of 9,8 element of the Beam Forming Matrix, correlating with virtual antenna 9 and beam antenna 8.
63–56	$BF_{8,11}I$ — Beam Forming 8,11 matrix I. This field is the real multiplication factor of 8,11 element of the Beam Forming Matrix, correlating with virtual antenna 8 and beam antenna 11.
55–48	$BF_{8,11}Q$ — Beam Forming 8,11 matrix Q. This field is the imaginary multiplication factor of 8,11 element of the Beam Forming Matrix, correlating with virtual antenna 8 and beam antenna 11.
47–40	$BF_{8,10}I$ — Beam Forming 8,10 matrix I. This field is the real multiplication factor of 8,10 element of the Beam Forming Matrix, correlating with virtual antenna 8 and beam antenna 10.
39–32	$BF_{8,10}Q$ — Beam Forming 8,10 matrix Q. This field is the imaginary multiplication factor of 8,10 element of the Beam Forming Matrix, correlating with virtual antenna 8 and beam antenna 10.

Table 26-371. CRPE-DL Beam Forming 4 Fields Description (Continued)

Bits	Description
31–24	$BF_{8,9}I$ — Beam Forming 8,9 matrix I. This field is the real multiplication factor of 8,9 element of the Beam Forming Matrix, correlating with virtual antenna 8 and beam antenna 9.
23–16	$BF_{8,9}Q$ — Beam Forming 8,9 matrix Q. This field is the imaginary multiplication factor of 8,9 element of the Beam Forming Matrix, correlating with virtual antenna 8 and beam antenna 9.
15–8	$BF_{8,8}I$ — Beam Forming 8,8 matrix I. This field is the real multiplication factor of 8,8 element of the Beam Forming Matrix, correlating with virtual antenna 8 and beam antenna 8.
7–0	$BF_{8,8}Q$ — Beam Forming 8,8 matrix Q. This field is the imaginary multiplication factor of 8,8 element of the Beam Forming Matrix, correlating with virtual antenna 8 and beam antenna 8.

26.5.5.7.12 CRPE-DL Beam Forming Coefficients Values Control Register 5 (CDBFCVLR5)

This register is used for configuring the beam forming matrix elements of following virtual antennas and beam antennas:

virtual antenna 11, beam antennas 8-11; virtual antenna 10, beam antennas 8-11.

Offset 0x0023F050 size 16 — Access: User read/write

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
R W	$BF_{11,11}I$								$BF_{11,11}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
R W	$BF_{11,10}I$								$BF_{11,10}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
R W	$BF_{11,9}I$								$BF_{11,9}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
R W	$BF_{11,8}I$								$BF_{11,8}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-311. CRPE-DL Beam Forming 5 Control Register

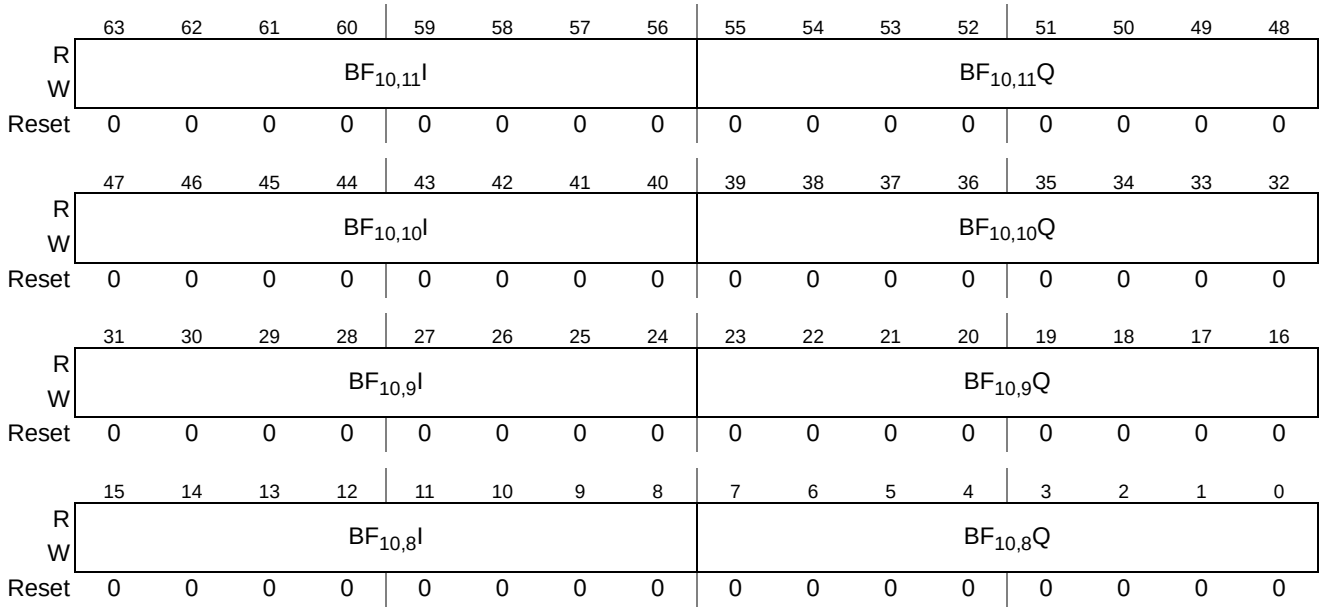


Figure 26-311. CRPE-DL Beam Forming 5 Control Register

Table 26-372. CRPE-DL Beam Forming 5 Fields Description

Bits	Description
127–120	$BF_{11,11}I$ — Beam Forming 11,11 matrix I. This field is the real multiplication factor of 11,11 element of the Beam Forming Matrix, correlating with virtual antenna 11 and beam antenna 11.
119–112	$BF_{11,11}Q$ — Beam Forming 11,11 matrix Q. This field is the imaginary multiplication factor of 11,11 element of the Beam Forming Matrix, correlating with virtual antenna 11 and beam antenna 11.
111–104	$BF_{11,10}I$ — Beam Forming 11,10 matrix I. This field is the real multiplication factor of 11,10 element of the Beam Forming Matrix, correlating with virtual antenna 11 and beam antenna 10.
103–96	$BF_{11,10}Q$ — Beam Forming 11,10 matrix Q. This field is the imaginary multiplication factor of 11,10 element of the Beam Forming Matrix, correlating with virtual antenna 11 and beam antenna 10.
95–88	$BF_{11,9}I$ — Beam Forming 11,9 matrix I. This field is the real multiplication factor of 11,9 element of the Beam Forming Matrix, correlating with virtual antenna 11 and beam antenna 9.
87–80	$BF_{11,9}Q$ — Beam Forming 11,9 matrix Q. This field is the imaginary multiplication factor of 11,9 element of the Beam Forming Matrix, correlating with virtual antenna 11 and beam antenna 9.
79–72	$BF_{11,8}I$ — Beam Forming 11,8 matrix I. This field is the real multiplication factor of 11,8 element of the Beam Forming Matrix, correlating with virtual antenna 11 and beam antenna 8.
71–64	$BF_{11,8}Q$ — Beam Forming 11,8 matrix Q. This field is the imaginary multiplication factor of 11,8 element of the Beam Forming Matrix, correlating with virtual antenna 11 and beam antenna 8.
63–56	$BF_{10,11}I$ — Beam Forming 10,11 matrix I. This field is the real multiplication factor of 10,11 element of the Beam Forming Matrix, correlating with virtual antenna 10 and beam antenna 11.
55–48	$BF_{10,11}Q$ — Beam Forming 10,11 matrix Q. This field is the imaginary multiplication factor of 10,11 element of the Beam Forming Matrix, correlating with virtual antenna 10 and beam antenna 11.
47–40	$BF_{10,10}I$ — Beam Forming 10,10 matrix I. This field is the real multiplication factor of 10,10 element of the Beam Forming Matrix, correlating with virtual antenna 10 and beam antenna 10.
39–32	$BF_{10,10}Q$ — Beam Forming 10,10 matrix Q. This field is the imaginary multiplication factor of 10,10 element of the Beam Forming Matrix, correlating with virtual antenna 10 and beam antenna 10.

Table 26-372. CRPE-DL Beam Forming 5 Fields Description

Bits	Description
31–24	$BF_{10,9}I$ — Beam Forming 10,9 matrix I. This field is the real multiplication factor of 10,9 element of the Beam Forming Matrix, correlating with virtual antenna 10 and beam antenna 9.
23–16	$BF_{10,9}Q$ — Beam Forming 10,9 matrix Q. This field is the imaginary multiplication factor of 10,9 element of the Beam Forming Matrix, correlating with virtual antenna 10 and beam antenna 9.
15–8	$BF_{10,8}I$ — Beam Forming 10,8 matrix I. This field is the real multiplication factor of 10,8 element of the Beam Forming Matrix, correlating with virtual antenna 10 and beam antenna 8.
7–0	$BF_{10,8}Q$ — Beam Forming 10,8 matrix Q. This field is the imaginary multiplication factor of 10,8 element of the Beam Forming Matrix, correlating with virtual antenna 10 and beam antenna 8.

26.5.5.7.13 CRPE-DL Beam Forming Coefficients Values Control Register 6 (CDBFCVLR6)

This register is used for configuring the beam forming matrix elements of following virtual antennas and beam antennas:

virtual antenna 13, beam antennas 12–15; virtual antenna 12, beam antennas 12–15.

Offset 0x0023F060 size 16 — Access: User read/write

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
R W	$BF_{13,15}I$								$BF_{13,15}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
R W	$BF_{13,14}I$								$BF_{13,14}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
R W	$BF_{13,13}I$								$BF_{13,13}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
R W	$BF_{13,12}I$								$BF_{13,12}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-312. CRPE-DL Beam Forming 6 Control Register

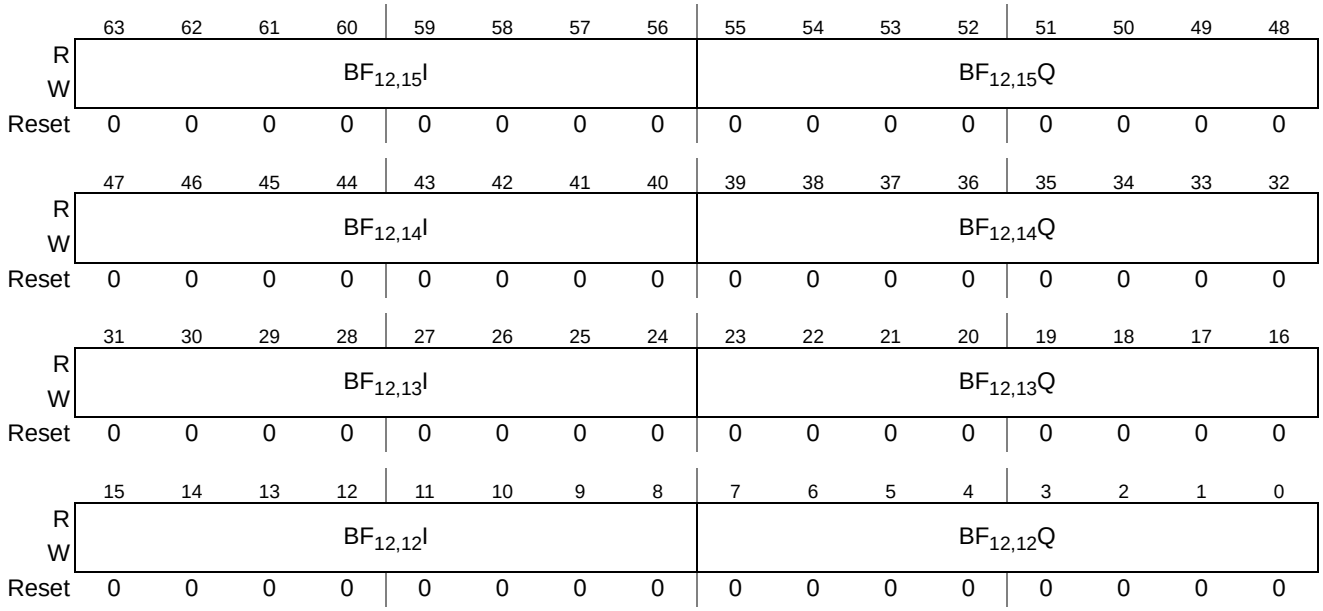


Figure 26-312. CRPE-DL Beam Forming 6 Control Register (Continued)

Table 26-373. CRPE-DL Beam Forming 6 Fields Description

Bits	Description
127–120	$BF_{13,15}I$ — Beam Forming 13,15 matrix I. This field is the real multiplication factor of 13,15 element of the Beam Forming Matrix, correlating with virtual antenna 13 and beam antenna 15.
119–112	$BF_{13,15}Q$ — Beam Forming 13,15 matrix Q. This field is the imaginary multiplication factor of 13,15 element of the Beam Forming Matrix, correlating with virtual antenna 13 and beam antenna 15.
111–104	$BF_{13,14}I$ — Beam Forming 13,14 matrix I. This field is the real multiplication factor of 13,14 element of the Beam Forming Matrix, correlating with virtual antenna 13 and beam antenna 14.
103–96	$BF_{13,14}Q$ — Beam Forming 13,14 matrix Q. This field is the imaginary multiplication factor of 13,14 element of the Beam Forming Matrix, correlating with virtual antenna 13 and beam antenna 14.
95–88	$BF_{13,13}I$ — Beam Forming 13,13 matrix I. This field is the real multiplication factor of 13,13 element of the Beam Forming Matrix, correlating with virtual antenna 13 and beam antenna 13.
87–80	$BF_{13,13}Q$ — Beam Forming 13,13 matrix Q. This field is the imaginary multiplication factor of 13,13 element of the Beam Forming Matrix, correlating with virtual antenna 13 and beam antenna 13.
79–72	$BF_{13,12}I$ — Beam Forming 13,12 matrix I. This field is the real multiplication factor of 13,12 element of the Beam Forming Matrix, correlating with virtual antenna 13 and beam antenna 12.
71–64	$BF_{13,12}Q$ — Beam Forming 13,12 matrix Q. This field is the imaginary multiplication factor of 13,12 element of the Beam Forming Matrix, correlating with virtual antenna 13 and beam antenna 12.
63–56	$BF_{12,15}I$ — Beam Forming 12,15 matrix I. This field is the real multiplication factor of 12,15 element of the Beam Forming Matrix, correlating with virtual antenna 12 and beam antenna 15.
55–48	$BF_{12,15}Q$ — Beam Forming 12,15 matrix Q. This field is the imaginary multiplication factor of 12,15 element of the Beam Forming Matrix, correlating with virtual antenna 12 and beam antenna 15.
47–40	$BF_{12,14}I$ — Beam Forming 12,14 matrix I. This field is the real multiplication factor of 12,14 element of the Beam Forming Matrix, correlating with virtual antenna 12 and beam antenna 14.
39–32	$BF_{12,14}Q$ — Beam Forming 12,14 matrix Q. This field is the imaginary multiplication factor of 12,14 element of the Beam Forming Matrix, correlating with virtual antenna 12 and beam antenna 14.

Table 26-373. CRPE-DL Beam Forming 6 Fields Description (Continued)

Bits	Description
31–24	$BF_{12,13}I$ — Beam Forming 12,13 matrix I. This field is the real multiplication factor of 12,13 element of the Beam Forming Matrix, correlating with virtual antenna 12 and beam antenna 13.
23–16	$BF_{12,13}Q$ — Beam Forming 12,13 matrix Q. This field is the imaginary multiplication factor of 12,13 element of the Beam Forming Matrix, correlating with virtual antenna 12 and beam antenna 13.
15–8	$BF_{12,12}I$ — Beam Forming 12,12 matrix I. This field is the real multiplication factor of 12,12 element of the Beam Forming Matrix, correlating with virtual antenna 12 and beam antenna 12.
7–0	$BF_{12,12}Q$ — Beam Forming 12,12 matrix Q. This field is the imaginary multiplication factor of 12,12 element of the Beam Forming Matrix, correlating with virtual antenna 12 and beam antenna 12.

26.5.5.7.14 CRPE-DL Beam Forming Coefficients Values Control Register 7 (CDBFCVLR7)

This register is used for configuring the beam forming matrix elements of following virtual antennas and beam antennas:

virtual antenna 15, beam antennas 12–15; virtual antenna 14, beam antennas 12–15.

Offset 0x0023F070 size 16 — Access: User read/write

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
R W	$BF_{15,15}I$								$BF_{15,15}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
R W	$BF_{15,14}I$								$BF_{15,14}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
R W	$BF_{15,13}I$								$BF_{15,13}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
R W	$BF_{15,12}I$								$BF_{15,12}Q$							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-313. CRPE-DL Beam Forming 7 Control Register

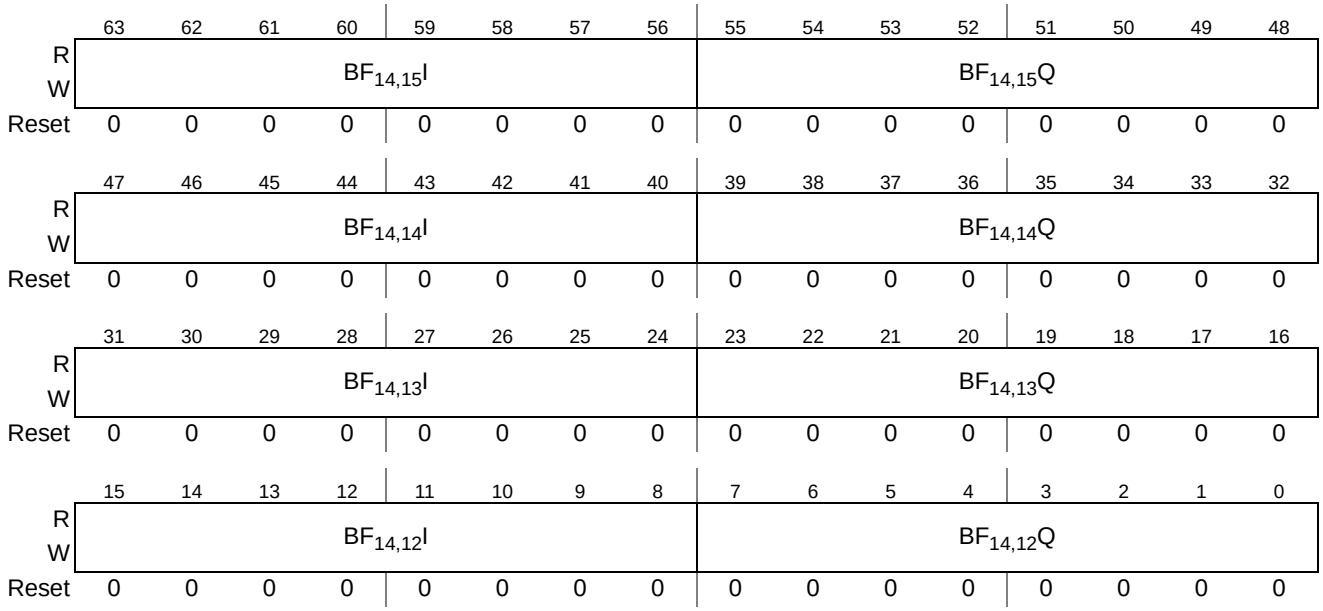


Figure 26-313. CRPE-DL Beam Forming 7 Control Register (Continued)

Table 26-374. CRPE-DL Beam Forming 7 Fields Description

Bits	Description
127–120	$BF_{15,15}I$ — Beam Forming 15,15 matrix I. This field is the real multiplication factor of 15,15 element of the Beam Forming Matrix, correlating with virtual antenna 15 and beam antenna 15.
119–112	$BF_{15,15}Q$ — Beam Forming 15,15 matrix Q. This field is the imaginary multiplication factor of 15,15 element of the Beam Forming Matrix, correlating with virtual antenna 15 and beam antenna 15.
111–104	$BF_{15,14}I$ — Beam Forming 15,14 matrix I. This field is the real multiplication factor of 15,14 element of the Beam Forming Matrix, correlating with virtual antenna 15 and beam antenna 14.
103–96	$BF_{15,14}Q$ — Beam Forming 15,14 matrix Q. This field is the imaginary multiplication factor of 15,14 element of the Beam Forming Matrix, correlating with virtual antenna 15 and beam antenna 14.
95–88	$BF_{15,13}I$ — Beam Forming 15,13 matrix I. This field is the real multiplication factor of 15,13 element of the Beam Forming Matrix, correlating with virtual antenna 15 and beam antenna 13.
87–80	$BF_{15,13}Q$ — Beam Forming 15,13 matrix Q. This field is the imaginary multiplication factor of 15,13 element of the Beam Forming Matrix, correlating with virtual antenna 15 and beam antenna 13.
79–72	$BF_{15,12}I$ — Beam Forming 15,12 matrix I. This field is the real multiplication factor of 15,12 element of the Beam Forming Matrix, correlating with virtual antenna 15 and beam antenna 12.
71–64	$BF_{15,12}Q$ — Beam Forming 15,12 matrix Q. This field is the imaginary multiplication factor of 15,12 element of the Beam Forming Matrix, correlating with virtual antenna 15 and beam antenna 12.
63–56	$BF_{14,15}I$ — Beam Forming 14,15 matrix I. This field is the real multiplication factor of 14,15 element of the Beam Forming Matrix, correlating with virtual antenna 14 and beam antenna 15.
55–48	$BF_{14,15}Q$ — Beam Forming 14,15 matrix Q. This field is the imaginary multiplication factor of 14,15 element of the Beam Forming Matrix, correlating with virtual antenna 14 and beam antenna 15.
47–40	$BF_{14,14}I$ — Beam Forming 14,14 matrix I. This field is the real multiplication factor of 14,14 element of the Beam Forming Matrix, correlating with virtual antenna 14 and beam antenna 14.
39–32	$BF_{14,14}Q$ — Beam Forming 14,14 matrix Q. This field is the imaginary multiplication factor of 14,14 element of the Beam Forming Matrix, correlating with virtual antenna 14 and beam antenna 14.

Table 26-374. CRPE-DL Beam Forming 7 Fields Description (Continued)

Bits	Description
31–24	$BF_{14,13}I$ — Beam Forming 14,13 matrix I. This field is the real multiplication factor of 14,13 element of the Beam Forming Matrix, correlating with virtual antenna 14 and beam antenna 13.
23–16	$BF_{14,13}Q$ — Beam Forming 14,13 matrix Q. This field is the imaginary multiplication factor of 14,13 element of the Beam Forming Matrix, correlating with virtual antenna 14 and beam antenna 13.
15–8	$BF_{14,12}I$ — Beam Forming 14,12 matrix I. This field is the real multiplication factor of 14,12 element of the Beam Forming Matrix, correlating with virtual antenna 14 and beam antenna 12.
7–0	$BF_{14,12}Q$ — Beam Forming 14,12 matrix Q. This field is the imaginary multiplication factor of 14,12 element of the Beam Forming Matrix, correlating with virtual antenna 14 and beam antenna 12.

26.5.5.7.15 CRPE-DL Start Control Register (CDSLR)

Writing to this register resets the frame, slot, sub slot and chunk count of the CRPE-DL (in *CDPSSR* register) and starts writing the output buffers. The data is ignored.

Offset 0x0023F080 Access: User 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																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-314. CRPE-DL Start Control Register

Table 26-375. CRPE-DL Start Control Fields Description

Bits	Description
31–0	Reserved.

26.5.5.7.16 CRPE-DL TPC Command Control Register (CDTCLR)

This register is used for updating the TPC field of a selected channel. The programming to this register shall take effect in the next processing of first input symbols associated with TPC field.

Offset 0x0023F084 Access: User read/write

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

Figure 26-315. CRPE-DL TPC Command Control Register

Table 26-376. CRPE-DL TPC Command Control Fields Description

Bits	Description
31–29	Reserved.
28	TPCOV — TPC Override. This bit indicates that TPC overriding symbol should be used instead of associated input stream. 0 - TPC override is turned off. 1 - TPC override is turned on.
27–26	Reserved.
25–16	TPCID — TPC Channel ID. This field indicates the channel whose TPC is being updated. The fields correlate with channels numbered #TPCID and #TPCID+512.
15–8	TPCSI — TPC Symbol Real Part. 8 bit field specifying the real part of the complex TPC overriding symbol. Value relates to symbol without STTD operation.
7–0	TPCSQ — TPC Symbol Imaginary Part. 8 bit field specifying the imaginary part of the complex TPC overriding symbol. Value relates to symbol without STTD operation.

26.5.5.7.17 CRPE-DL Virtual Antennas Gain Command Control Register (CDVAGCLR)

This register defines the slot at which virtual antenna gains programmed in *CDVAGCLR0* and *CDVAGCLR1* start to affect. Until this slot, previously programmed gains are used. The use of the updated gains is synchronized to the beginning of the field pointed by associated *GUF* field from slot format LUT, and to the beginning of the first slot of the channel.

Offset 0x0023F090 Access: User read/write

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

Figure 26-316. CRPE-DL Virtual Antennas Gain Command Control Register

Table 26-377. CRPE-DL Virtual Antennas Gain Command Control Fields Description

Bits	Description
31–26	Reserved.
25–16	VCID — Virtual Antenna Channel ID. This field specifies which physical channel gain parameters should be updated with the data written in the *CDVAGCLR0* and *CDVAGCLR1* registers. The fields correlate with channels numbered #VCID and #VCID+512.
15–5	Reserved.
4	GUN — Gain Update Next. This field is indicates whether gain update should take effect at slot pointed by GUS or at the next earliest update time. 0 - The gain should be updated at slot pointed by *GUS*. 1 - The gain should be updated at the next earliest update time.
3–0	GUS — Gain Update Slot. This field is specifying the slot number in frame to start using the new virtual antenna parameters. This field takes affect only while *GUN* field is cleared. In addition, new virtual antenna parameters are used at the beginning of the first slot of the channel. Range: 0-14.

26.5.5.7.18 CRPE-DL General Command Control Register (CDGCLR)

This register is used for updating the channel general parameters. An active channel should not override parameters as long as CRPE-DL is still processing the channel.

Offset 0x0023F098 size 8 — Access: User read/write

	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
R	CMSFIX							CMSCR				SCRIX				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
R	SFIX								VANTB				VANTA			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TSTD EN	STTD		VANT ABEN	SCH		GCCID									
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	CMOVSF								OVSF							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-317. CRPE-DL General Command Control Register

Table 26-378. CRPE-DL General Command Control Fields Description

Bits	Description
63–57	CMSFIX — Compressed Mode Slot Format Index. A pointer to entry of Slot Format LUT, indicating slot format of compressed frames. Range: 0 to 89.
56–55	CMSCR — Compressed Mode Scrambling Code Index. This field indicates which scrambling code is used at compressed mode. 0 - Same as the one used in non-compressed mode. 1 - Left alternative scrambling code. 2 - Right alternative scrambling code. 3 - Reserved
54–53	Reserved
52–48	SCRIX — Scrambling Code Index. A pointer to entry of Scrambling Initialization LUT Memory, indicating SN of channel's scrambling code. Range: 0 to 31.
47–41	SFIX — Slot Format Index. A pointer to entry of Slot Format LUT, indicating slot format of non-compressed frames. Range: 0 to 89.
40	Reserved
39–36	VANTB — Virtual Antenna B. Destination Virtual Antenna B. Range: 0 to 15. **Note**: $VANTB_X$ must not equal $VANTA_X$.
35–32	VANTA — Virtual Antenna A. Destination Virtual Antenna A. Range: 0 to 15. **Note**: $VANTA_X$ must not equal $VANTB_X$.

Table 26-378. CRPE-DL General Command Control Fields Description

Bits	Description
31	TSTDEN — TSTD Enable. This field indicates whether CRPE-DL should perform TSTD operation. When TSTD operation is enabled, in even numbered slot channel output stream is targeted to Virtual Antenna A; while in odd numbered slot channel output stream is targeted to Virtual Antenna B. When TSTD operation is disabled, channel output streams are generated for both antennas. *TSTDEN* may be enabled only for synchronization channels. 0 - TSTD operation is disabled. 1 - TSTD operation is enabled.
30–29	STTD — STTD Operation Field. This field indicates how STTD operation relates to the channel. 00 - STTD operation should not be performed on the channel's input stream and there is no coupled channel on which the STTD operation is performed externally. This option can be associated with channels 0-511. This option must be selected when TSTDEN is enabled. 01 - STTD operation should not be performed on the channel's input stream and there is a coupled channel on which the STTD operation is performed externally. This option can be associated with channels 0-511. 10 - STTD operation should be performed on the channel's input stream in addition to the non-STTD operation. CRPE-DL should process the channel twice: one time without performing STTD operation and second time with STTD operation. The processed output stream without STTD operation is targeted to Virtual Antenna A, while the processed output stream with STTD operation is targeted to Virtual Antenna B. This option can be associated with channels 0-511. This option is not supported for channels with Spreading Factor 256. 11 - STTD operation was performed externally on the channel's input stream and there is a coupled channel on which the STTD operation is not performed externally. This option can be associated with channels 512–1023.
28	VANTABEN — Virtual Antenna B Enable. This field indicates whether channel's processing should be targeted to Virtual Antenna B in addition to Virtual Antenna A. VANTABEN must be set when STTD equals 2 and when TSTDEN is set. VANTABEN must be cleared when STTD equals 1 or 3. 0 - Channel is not targeted to Virtual Antenna B. 1 - Channel is targeted to Virtual Antenna B in addition to Virtual Antenna A.
27	SCH — Synchronization Channel. This field indicates whether the channel is a synchronization channel. When synchronization channel is processed, each slot produces only 256 chips, starting from offset 0. This option can be associated with channels 0-511. 0 - channel is not a synchronization channel. 1 - channel is a synchronization channel.
26	Reserved
25–16	GCCID — General Command Channel ID. This field indicates the channel whose general parameters are being updated. Range: 0 to 1023.
15–8	CMOVSF — Compressed Mode OVSF. This field indicates the Cch, SF, OVSF code used at compressed mode, used as specified in the standard. Range: 0 to 255.
7–0	OVSF — Orthogonal Variable Spreading Factor. This field indicates the Cch, SF, OVSF code, used as specified in the standard. Range: 0 to 255.

26.5.5.7.19 CRPE-DL Idle Period Control Register <x> (CDIPLRx)

These registers define the idle period of each virtual antenna. There are 16 such registers ($0 \leq x \leq 15$), where each register index refer to antenna index.

Offset 0x0023F0A0 Access: User read/write
offset 4
x = 0...15

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IPV_i				$IPESL_i$								$IPESBS_i$			
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					$IPSSL_i$								$IPSSBS_i$			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-318. CRPE-DL Idle Period Control Register

Table 26-379. CRPE-DL Idle Period Fields Description

Bits	Description
31	IPV_i — Idle Period Valid VA_i. This field is specifying if idle period should be performed for virtual antenna i. This bit is set by the user and cleared by CRPE-DL once idle period is performed from start to end. 0 - idle period should not be performed. 1 - idle period should be performed from start to end.
30–28	Reserved.
27–24	$IPESL_i$ — Idle Period End Slot VA_i. This field is specifying the end slot in frame of idle period for virtual antenna i. Range: 0 to 14
23–20	Reserved.
19–16	$IPESBS_i$ — Idle Period End Sub-Slot VA_i. This field is specifying the end sub-slot in slot of idle period for virtual antenna i. Range: 0 to 9
15–12	Reserved.
11–8	$IPSSL_i$ — Idle Period Start Slot VA_i This field is specifying the start slot in frame of idle period for virtual antenna i. Range: 0 to 14
7–4	Reserved.
3–0	$IPSSBS_i$ — Idle Period Start Sub-Slot VA_i. This field is specifying the start sub-slot in slot of idle period for virtual antenna i. Range: 0 to 9

26.5.5.7.20 CRPE-DL Beam Forming Coefficients Timing Command Control Register <x> (CDBFCTCLRx)

These registers define the time at which beam forming coefficients programmed in *CDBFCVLR* registers start to affect. Until this time previously programmed coefficients are used.

In addition, these registers also define the shift right operation that is performed after multiplication by beam-forming coefficients.

CDBFCTCLR0 correlates with *CDBFCVLR(0-1)* and virtual antennas 0-3, *CDBFCTCLR1* correlates with *CDBFCVLR(2–3)* and virtual antennas 4–7, *CDBFCTCLR2* correlates with

CDBFCVLR(4–5) and virtual antennas 8–11, and *CDBFCTCLR3* correlates with *CDBFCVLR(6–7)* and virtual antennas 12–15.

Offset 0x0023F0E0 + offset 4 x = 0...3 — Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BFSUN			BFSNUM					BFSUSL				BFSUSS			
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								BFUN	BFUSL				BFUSS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-319. CRPE-DL Beam Forming Coefficients Timing Command Control Register

Table 26-380. CRPE-DL Beam Forming Coefficients Timing Command Fields Description

Bits	Description
31	BFSUN — Beam Forming Shift Update Next. This field is indicates whether beam forming shift number update should take effect at slot and sub-slot as pointed by *BFSUSL* and *BFSUSS* or at the next earliest sub-slot time. 0 - The shift number should be updated at slot and sub-slot as pointed by *BFSUSL* and *BFSUSS*. 1 - The gain should be updated at the next earliest sub-slot time.
30–29	Reserved.
28–24	BFSNUM — Beam Forming Shift Number. This field is directly specifying the number of shift right operations to perform after multiplication by beam-forming coefficients. Range: 0 to 16
23–20	BFSUSL — Beam Forming Shift Update Slot. This field is specifying the slot number in frame to start using the new beam forming shift number. Range: 0 to 14
19–16	BFSUSS — Beam Forming Shift Update Sub-Slot. This field is specifying the sub-slot number in slot to start using the new beam forming shift number. Range: 0 to 9
15–9	Reserved.
8	BFUN — Beam Forming Update Next. This field is indicates whether beam forming coefficients update should take effect at slot and sub-slot as pointed by *BFUSL* and *BFUSS* or at the next earliest sub-slot time. 0 - The gain should be updated at slot and sub-slot as pointed by *BFUSL* and *BFUSS*. 1 - The gain should be updated at the next earliest update time.
7–4	BFUSL — Beam Forming Update Slot. This field is specifying the slot number in frame to start using the new beam forming coefficients. Range: 0 to 14
3–0	BFUSS — Beam Forming Update Sub-Slot. This field is specifying the sub-slot number in slot to start using the new beam forming coefficients. Range: 0 to 9

26.5.5.7.21 CRPE-DL Combined Chips Shift Command Control Register <x> (CDCCSCLRx)

These registers define the shift right operation that is performed after chips accumulation.

CDCCSCLR0 correlates with virtual antennas 0-3, *CDCCSCLR1* correlates with virtual antennas 4–7, *CDCCSCLR2* correlates with virtual antennas 8–11, and *CDCCSCLR3* correlates with virtual antennas 12–15.

Offset 0x0023F0F0 Access: User read/write
offset 4
x = 0...3

	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	CCSUN			CCSNUM					CCSUSL				CCSUSS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-320. CRPE-DL Combined Chips Shift Command Control Register

Table 26-381. CRPE-DL Combined Chips Shift Command Fields Description

Bits	Description
31–16	Reserved.
15	CCSUN — Combined Chips Shift Update Next. This field is indicates whether combined chips shift number update should take effect at slot and sub-slot as pointed by *CCSUSL* and *CCSUSS* or at the next earliest sub-slot time. 0 - The shift number should be updated at slot and sub-slot as pointed by *CCSUSL* and *CCSUSS*. 1 - The gain should be updated at the next earliest sub-slot time.
14–13	Reserved.
12–8	CCSNUM — Combined Chips Shift Number. This field is directly specifying the number of shift right operations to perform for the combined chips. Range: 0 to 16
7–4	CCSUSL — Combined Chips Shift Update Slot. This field is specifying the slot number in frame to start using the new combined chips shift number. Range: 0 to 14
3–0	CCSUSS — Combined Chips Shift Update Sub-Slot. This field is specifying the sub-slot number in slot to start using the new combined chips shift number. Range: 0 to 9

26.5.5.7.22 CRPE-DL Rate Control Register (CDRLR)

This register takes affect when Rate Control Source *RCS* field in *CDOMCP* initialization parameter (see **Section 26.5.2.4**, *CRPE-DL Output Mode Configuration Parameter (CDOMCP)*) is set. The register provide fields to indicate rate control events of new 16 chips chunk and new sub-slot. Only one rate control event takes affect, according to Rate Selector *RSEL* field value in *CDOMCP* initialization parameter.

The fields at this register are set by the host and cleared by the CRPE-DL.

Offset 0x0023F10C 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								BSE								CSE
W								1								1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-321. CRPE-DL Rate Control Register

Table 26-382. CRPE-DL Rate Fields Description

Bits	Description
31–9	Reserved.
8	BSE — Sub-Slot Start Event. Writing 1 to this field indicates that new 256 chips sub-slot started. This field is cleared by the CRPE-DL. This field takes affect when *RCS* and *RSEL* fields are set at *CDOMCP* initialization parameter.
7–1	Reserved.
0	CSE — 16 Chips Chunk Start Event. Writing 1 to this field indicates that new 16 chips chunk started. This field is cleared by the CRPE-DL. This field takes affect when *RCS* field is set and *RSEL* field is cleared at *CDOMCP* initialization parameter.

26.5.5.7.23 CRPE-DL Event Status Register (CDESR)

A bit in this register is set according to events and are reset when writing ‘1’ to the location of the set bit.

Offset 0x0023F100 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				MI				OBS								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-322. CRPE-DL Event Status Register

Table 26-383. CRPE-DL Event Status Fields Description

Bits	Description
31–13	Reserved.
12	MI — Memory Initialized. 0 - Memory initialization is not complete. 1 - Memory initialization is complete.
11–9	Reserved.
8	OBS — Output Buffer Status. 0 - Output buffer’s first entry is not ready to be fetched since last time CDSLR was written. 1 - Output buffer’s first entry is ready to be fetched for the first time since last time CDSLR was written.
7–0	Reserved.

26.5.5.7.24 CRPE-DL Processing Stage Status Register (CDPSSR)

The fields in this register indicate the state of CRPE-DL in terms of frame, slot, sub-slot and chunk count. The register is cleared at *CDSLR* initialization.

Offset 0x0023F104 Access: User read/write

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

Figure 26-323. CRPE-DL Processing Stage Status Register

Table 26-384. CRPE-DL Processing Stage Status Fields Description

Bits	Description
31–28	Reserved.
27–16	FI — Frame Index. This field is set to 0 when *CDSLR* is set and increments every time CRPE-DL starts processing the first 16 chips of a new frame. Range: 0 to 4096.
15–12	Reserved.
11–8	SI — Slot Index. This field is set to 0 when *CDSLR* is set and increments every time CRPE-DL starts processing the first 16 chips of a new slot. Range: 0 to 14.
7–4	BI — Sub Slot Index. This field is set to 0 when *CDSLR* is set and increments every time CRPE-DL starts processing the first 16 chips of a new sub-slot. Range: 0 to 9.
3–0	CI — Chunk Index. This field is set to 0 when *CDSLR* is set and increments every time CRPE-DL starts processing new 16 chips processing chunk. Range: 0 to 15.

26.5.5.7.25 CRPE-DL ECC Status Register (CDECCSR)

The fields in this register indicate on multi ECC error events in the CRPE-DL internal memories. Clearing a set bit is done by writing ‘1’ to it.

Offset 0x0023F108 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		ME14	ME13	ME12	ME11	ME10	ME9	ME8	ME7	ME6	ME5	ME4	ME3	ME2	ME1	ME0
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																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-324. CRPE-DL ECC Status Register

Table 26-385. CRPE-DL ECC Status Fields Description

Bits	Description
31	Reserved.
30	ME14 — Multiple ECC Error #14. Indicates ECC Error in OUTBUF1 memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
29	ME13 — Multiple ECC Error #13. Indicates ECC Error in OUTBUF0 memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
28	ME12 — Multiple ECC Error #12. Indicates ECC Error in VAGS memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
27	ME11 — Multiple ECC Error #11. Indicates ECC Error in VAGA memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
26	ME10 — Multiple ECC Error #10. Indicates ECC Error in VAT memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
25	ME9 — Multiple ECC Error #9. Indicates ECC Error in TPCA memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
24	ME8 — Multiple ECC Error #8. Indicates ECC Error in TPCS memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
23	ME7 — Multiple ECC Error #7. Indicates ECC Error in INBUF1 memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur

Table 26-385. CRPE-DL ECC Status Fields Description

Bits	Description
22	ME6 — Multiple ECC Error #6. Indicates ECC Error in INBUF0 memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
21	ME5 — Multiple ECC Error #5. Indicates ECC Error in GC memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
20	ME4 — Multiple ECC Error #4. Indicates ECC Error in INWP1 memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
19	ME3 — Multiple ECC Error #3. Indicates ECC Error in INWP0 memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
18	ME2 — Multiple ECC Error #2. Indicates ECC Error in SF_LUT memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
17	ME1 — Multiple ECC Error #1. Indicates ECC Error in SLC1 memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
16	ME0 — Multiple ECC Error #0. Indicates ECC Error in SLC0 memory. Clearing this bit is by write ‘1’. 1 - ECC error has occur 0 - No ECC error has occur
15–0	Reserved

27 Security Engine (SEC)

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.

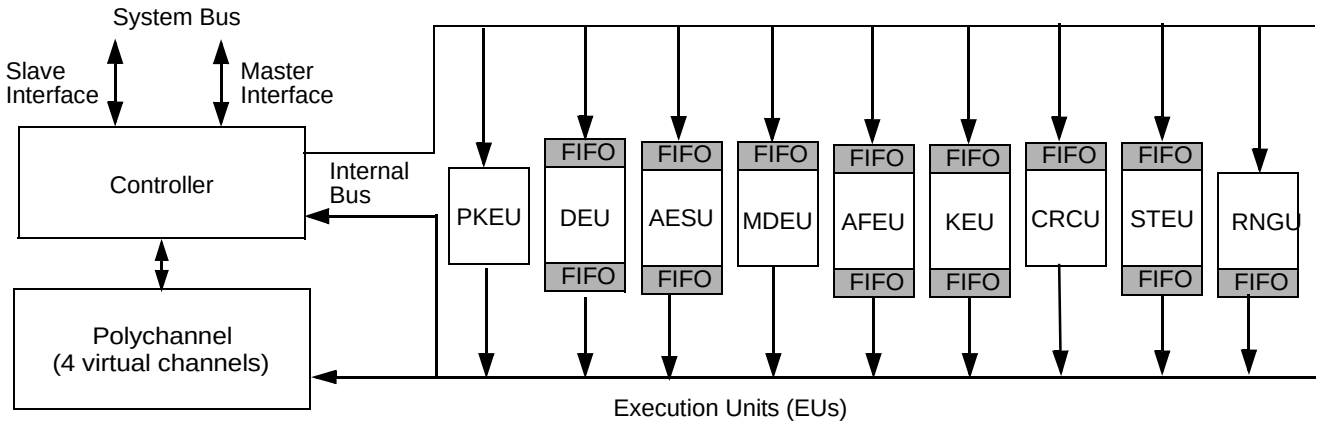


Figure 27-1. SEC Functional Diagram

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 signalling, 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 27-100.

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 signalling, see **Section 27.7.6.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4])*, on page 27-199. For detail on the writeback fields, see **Section 27.7.1.2**, *Descriptor Header*, on page 27-102.

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.

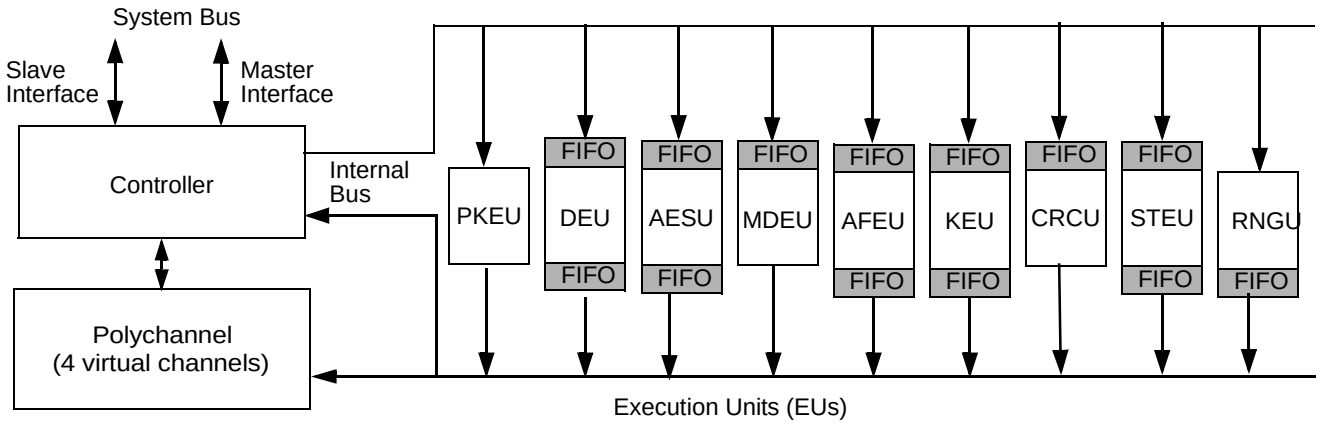


Figure 27-2. SEC Functional Diagram

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 27-179). 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 27-179), 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 27-179.
- 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 27-182.

- 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 27-182.
- 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 27-183.
- 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 27-185.
- 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 27-189.
- 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 27-192.

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 27-199.
- 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 27-202.
- 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 27-207.
- 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 27-209.
- 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 27-210.

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 27-189 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 27-185 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 27-199). 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 27-202 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 27-100 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.

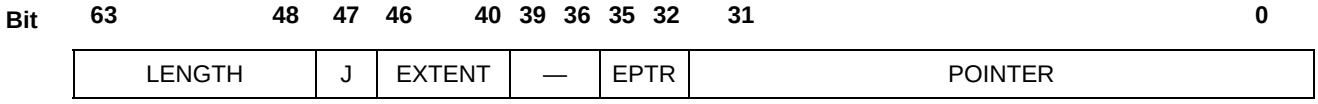


Figure 27-3. Descriptor Pointer Layout

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 27-177 for details on the Link Table programming model. **Figure 27-4** shows the layout of a link table entry.

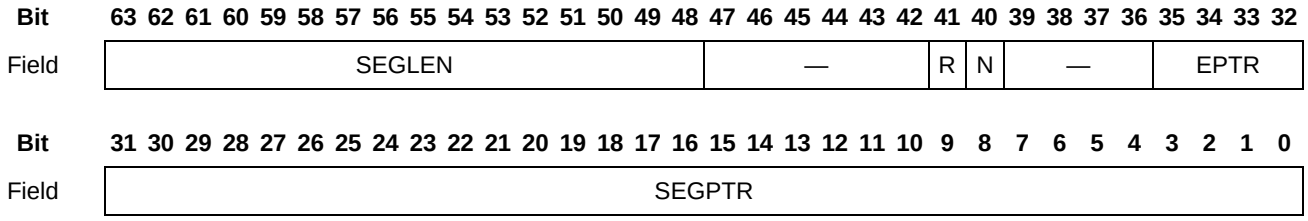


Figure 27-4. Link Table Entry

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 27-202).

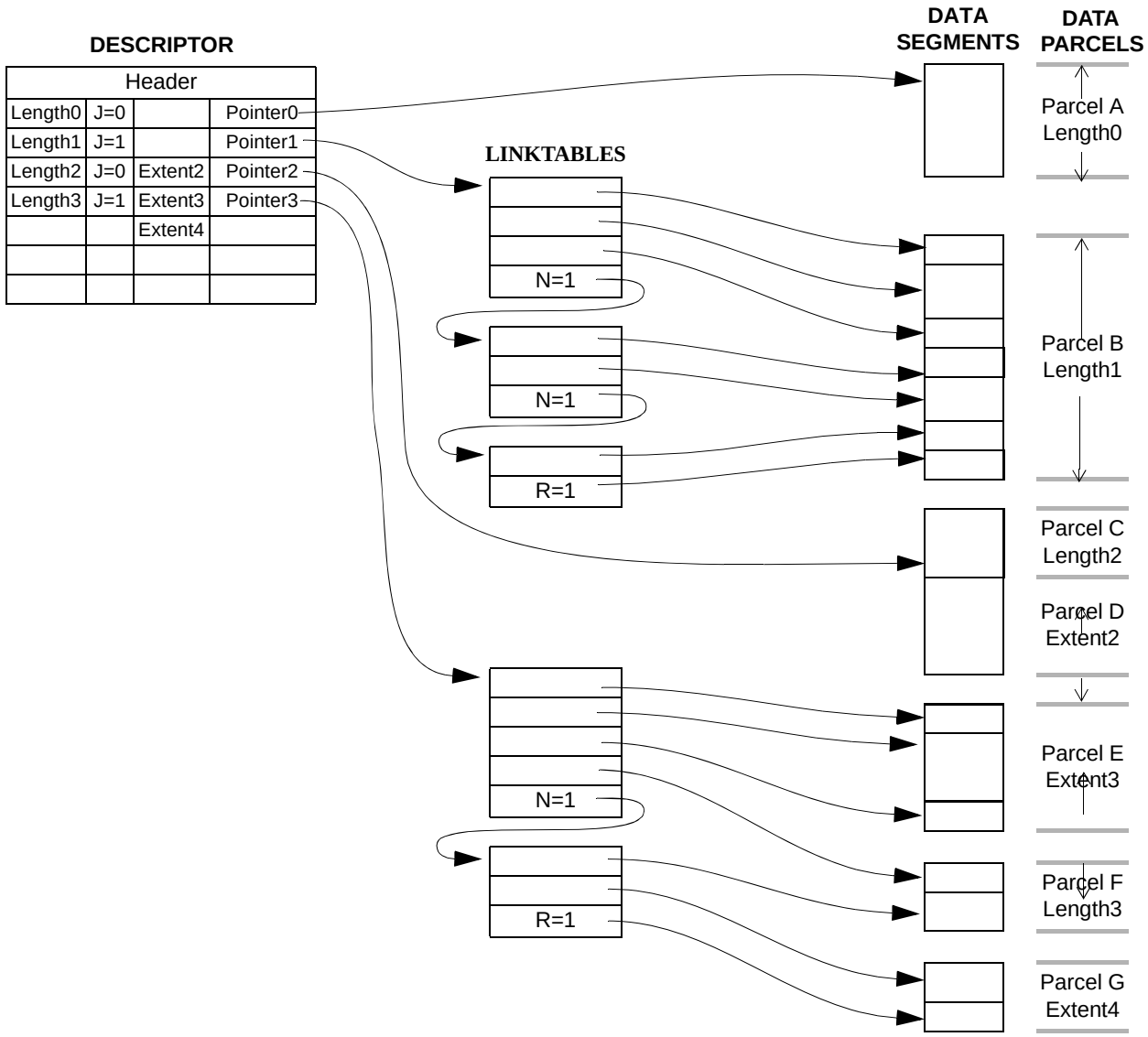


Figure 27-5. Descriptors, Link Tables, and Parcels

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 27-96). 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 27-96 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 27-96 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 27-202) 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 27-211*** 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: $B = \underline{A}^E \bmod N$, 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, $\underline{A} = AR \pmod N$, 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 27-96 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 27-202) 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 27-239).

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 27-199), 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 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 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 27-96 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 27-202) 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 27-241), 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 that is, 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, that is, 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 27-96 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 27-199), 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 27-102 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 27-252 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 27-100.

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 27-260).

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 27-189). 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 27-202) 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 27-258), 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 27-252). 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 27-179) 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 27-96 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 27-274).

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 27-189). 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 27-202) 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 27-272), 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 27-269). 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 27-199), and mask the ICE bit in the Interrupt Mask Register (**Section 27.7.12.7**, *KEU Interrupt Mask Register (KEUIMR)*, on page 27-286). 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 27-96 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 27-278). 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 27-286).

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 27-189). 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 27-202) 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 27-284), 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 SEC EU Assignment Status Register shows which channels own which EUs at a particular moment in time. However, there is a missing value in the EU Assignment Register (EUASR) such that the STEU (Snow3G Execution Unit) field in the EUASR is not updated properly, and always reads back as 0xF. As a result, it is not possible to know which (if any) channel is using the STEU. When operating the SEC in descriptor mode, the user does not have any reason to care which channel owns the STEU at a particular moment in time. STEU descriptors can be dispatched to any channel, and the SEC controller ensures that all channels get access to the STEU.

The only conditions under which the EU Assignment Register is useful is when mixing descriptor based operations with direct access mode (slave mode) operations. Direct access mode operations involve software writing directly to EU registers, bypassing the channels entirely. This is not a recommended mode of operation, but if there is a compelling reason to use direct access mode for STEU operations, only use direct access mode. If not, software must ensure that all STEU descriptors have completed before performing an STEU direct access. The recommendation is to use descriptors for all STEU operations. If the alternative is required, use direct access mode for all STEU operations. Do not switch between the two modes.

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 27-323.

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.

In the MSC8157E, any hardware comparison of a received MAC to a MAC-1 value fails. Therefore, never set CICV = 1 for a SNOW F9 operation. To make the comparison using software with CICV = 0, the user constructs a SNOW F9 descriptor to instruct the SEC to output the calculated F9 MAC so that the software can compare the calculated MAC with the received MAC-1 value. Refer to the *SEC 2/3x Descriptor Programmer's Guide* for details on constructing the descriptor.

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 27-96 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 27-335 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 27-336 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 27-337).

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 27-189). 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 27-202) 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 27-337), 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

Note: To prevent stalling and enable the detection of stalled channels, ensure the following conditions:

- Ensure that software does not create SEC descriptors that request encryption or decryption of zero length data.
- Use the SEC crypto-channel watchdog timer to detect stalled channels. Enable the individual channel watchdog timer via the Channel Configuration Register CCRx[WGN] field. This is a one-time configuration. The timer stops when the channel completes a descriptor and restarts at zero each time the channel fetches a new descriptor. The default setting for the watchdog timer is 227 SEC clock cycles. If the timer expires, the channel generates an interrupt and the Channel Status Register shows the cause as a Watchdog Timeout (WDT). The channel stall is cleared by resetting the channel via CCR[R]. The offending EU is reset automatically by the channel.

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 27-100**
  - Descriptor header, **page 27-102**
  - Descriptor Pointers, **page 27-175**
  - Link tables, **page 27-177**
- SEC Controller
  - Master Control Register (MCR), **page 27-179**
  - Controller Identification Register (CIDR), **page 27-182**
  - Controller IP Block Revision Register (CIPBRR), **page 27-182**
  - EU Assignment Status Register (EUASR), **page 27-183**
  - Controller Interrupt Enable Register (CIER), **page 27-185**
  - Controller Interrupt Status Register (CISR), **page 27-189**
  - Controller Interrupt Clear Register (CICR), **page 27-192**
- Polychannel
  - Fetch FIFO Enqueue Counter (FFEC), **page 27-195**
  - Descriptor Finished Counter (DFC), **page 27-196**
  - Data Bytes In Counter (DBIC), **page 27-197**
  - Data Bytes Out Counter (DBOC), **page 27-198**
- Channels
  - Channel 1–4 Configuration Registers (CCR[1–4], **page 27-199**
  - Channel 1–4 Pointer Registers (CSR[1–4]), **page 27-202**
  - Channel 1–4 Current Descriptor Pointer Registers (CDPR[1–4]), **page 27-207**
  - Channel 1–4 Fetch FIFOs (CFF[1–4], **page 27-209**

- MDEU Data Size Register (MDEUDSR), **page 27-255**
- MDEU Reset Control Register (MDEURCR), **page 27-256**
- MDEU Status Register (MDEUSR), **page 27-257**
- MDEU Interrupt Status Register (MDEUISR), **page 27-258**
- MDEU Interrupt Mask Register (MDEUIMR), **page 27-260**
- MDEU ICV Size Register (MDEUICVSR), **page 27-262**
- MDEU End_of_Message Register (MDEUEOMR), **page 27-263**
- MDEU Context Registers (MDEUCR), **page 27-264**
- MDEU Key Registers (MDEUKR[1–8]), **page 27-266**
- MDEU Input FIFO, **page 27-266**

■ AFEU

- AFEU Mode Register (AFEUMR), **page 27-267**
- AFEU Key Size Register (AFEUKSR), **page 27-268**
- AFEU Content/Data Size Register (AFEUCDSR), **page 27-269**
- AFEU Reset Control Register (AFEURCR), **page 27-270**
- AFEU Status Register (AFEUSR), **page 27-271**
- AFEU Interrupt Status Register (AFEUISR), **page 27-272**
- AFEU Interrupt Mask Register (AFEUIMR), **page 27-274**
- AFEU End_of_Message Register (AFEUEOMR), **page 27-276**
- AFEU Context Memory, **page 27-276**
- AFEU Context Memory Pointer Register (AFEUCMPR), **page 27-277**
- AFEU Key Registers (AFEUKR[1–2]), **page 27-277**
- AFEU Input FIFO / Output FIFO, **page 27-277**

■ KEU

- KEU Mode Register (KEUMR), **page 27-278**
- KEU Key Size Register (KEUKSR), **page 27-280**
- KEU Data Size Register (KEUDSR), **page 27-281**
- KEU Reset Control Register (KEURCR), **page 27-282**
- KEU Status Register (KEUSR), **page 27-283**
- KEU Interrupt Status Register (KEUISR), **page 27-284**
- KEU Interrupt Mask Register (KEUIMR), **page 27-286**
- KEU Data Out Register (KEUDOR), **page 27-288**
- KEU End_of_Message Register (KEUEOMR), **page 27-289**
- KEU IV1 Register (KEUIV1R), **page 27-290**
- KEU ICV_In Register (KEUICVIR), **page 27-291**
- KEU IV2 Register (KEUIV2R), **page 27-292**
- KEU Context Registers 1-6 (KEUCR[1–6]), **page 27-293**
- KEU Key Data Registers 1–2 (KEUKDR[1–2]), **page 27-294**
- KEU Key Data Registers 3–4 (KEUKDR[3–4], **page 27-295**
- KEU Input FIFO/Output FIFO, **page 27-295**

■ CRCU

- CRCU Mode Register (CRCUMR), **page 27-296**
- CRCU Key Size Register (CRCUKSR), **page 27-298**
- CRCU Data Size Register (CRCUDSR), **page 27-299**
- CRCU Reset Control Register (CRCURCR), **page 27-300**
- CRCU Control Register (CRCUCR), **page 27-301**
- CRCU Status Register (CRCUSR), **page 27-302**
- CRCU Interrupt/Error Status Register (CRCUISR), **page 27-303**
- CRCU Interrupt/Error Mask Register (CRCUIMR), **page 27-305**
- CRCU ICV Size Register (CRCUICVSR), **page 27-307**
- CRCU End_of_Message Register (CRCUEOMR), **page 27-308**
- CRCU Context Register (CRCUCXR), **page 27-309**
- CRCU Key Register (CRCUKR), **page 27-310**
- CRCU Input FIFO, **page 27-310**

■ STEU

- STEU Mode Register (STEUMR), **page 27-311**
- STEU Key Size Register (STEUKSR), **page 27-312**
- STEU Data Size Register (STEUDSR), **page 27-313**
- STEU Reset Control Register (STEURCR), **page 27-314**
- STEU Status Register (STEUSR), **page 27-315**
- STEU Interrupt Status Register (STEUISR), **page 27-316**
- STEU Interrupt Mask Register (STEUIMR), **page 27-318**
- STEU Data Out Register (STEUDOR), **page 27-320**
- STEU End_of_Message Register (STEUEOMR), **page 27-321**
- STEU IV1 Register (STEUIV1R), **page 27-322**
- STEU ICV_In Register (STEUICVIR), **page 27-323**
- STEU IV2 Register (STEUIV2R), **page 27-324**
- STEU Context Registers 1 (STEUCR1), **page 27-325**
- STEU Context Registers 2 (STEUCR2), **page 27-326**
- STEU Context Registers 3 (STEUCR3), **page 27-327**
- STEU Context Registers 4 (STEUCR4), **page 27-328**
- STEU LFSR State Register 0–7 (STEULFSRSR[0–7]), **page 27-329**
- STEU FSM State Register 1 (STEUFSMSR1), **page 27-330**
- STEU FSM State Register 2 (STEUFSMSR2), **page 27-331**
- STEU Key Data Registers 1–2 (STEUKDR[1–2]), **page 27-332**
- STEU Input FIFO/Output FIFO, **page 27-332**

■ Random Number Generator (RNGU)

- RNGU Mode Register (RNGMR), **page 27-333**
- RNGU Data Size Register (RNGDSR), **page 27-334**
- RNGU Reset Control Register (RNGRCR), **page 27-335**
- RNGU Status Register (RNGSR), **page 27-336**
- RNGU Interrupt Status Register (RNGISR), **page 27-337**

— RNGU Interrupt Mask Register (RNGIMR), **page 27-338**
— RNGU End_of_Message Register (RNGEOMR), **page 27-340**
— RNGU Entropy Registers 0–7 (RNGER[0–7], **page 27-340**
— RNGU Output FIFO, **page 27-341**

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 27-199).	
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.

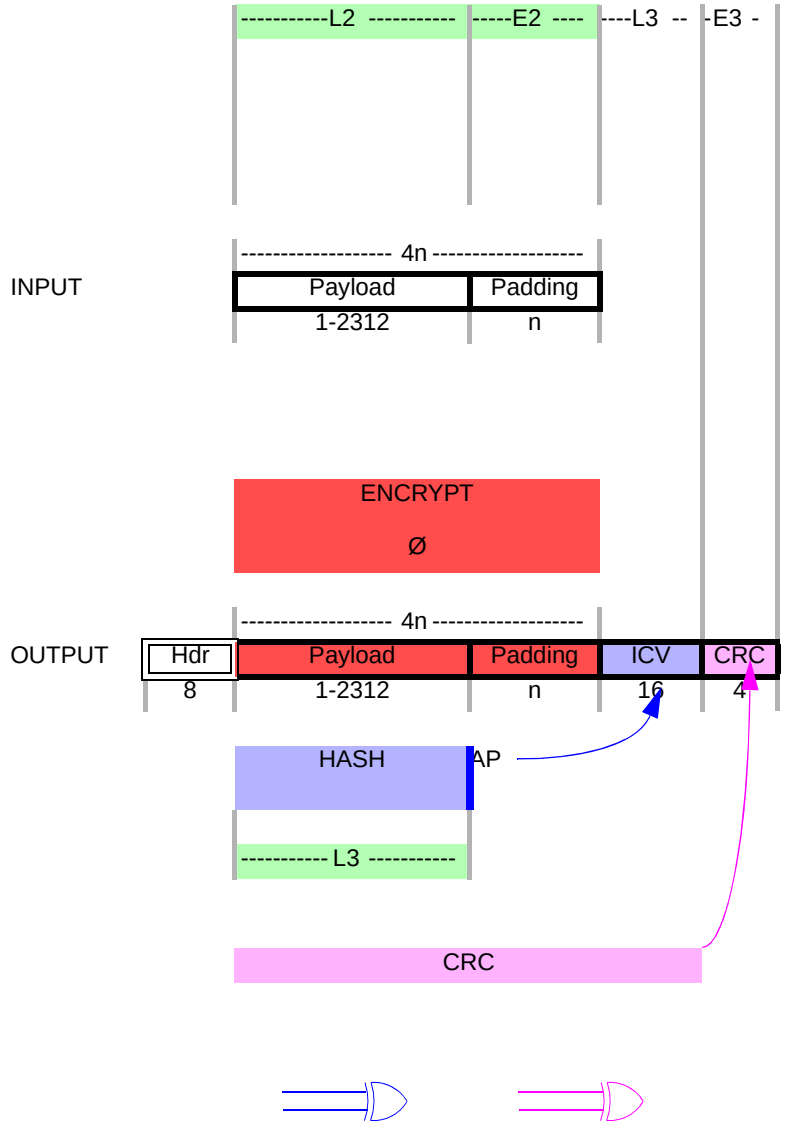
Relevant input data fields are shown with lengths labelled below them. Any additional restrictions on field sizes are shown above the fields. The following size designators are used:
B - bytes (the default for all lengths)
b - bits
n - any positive integer
4n - integer multiple of 4

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 161.

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

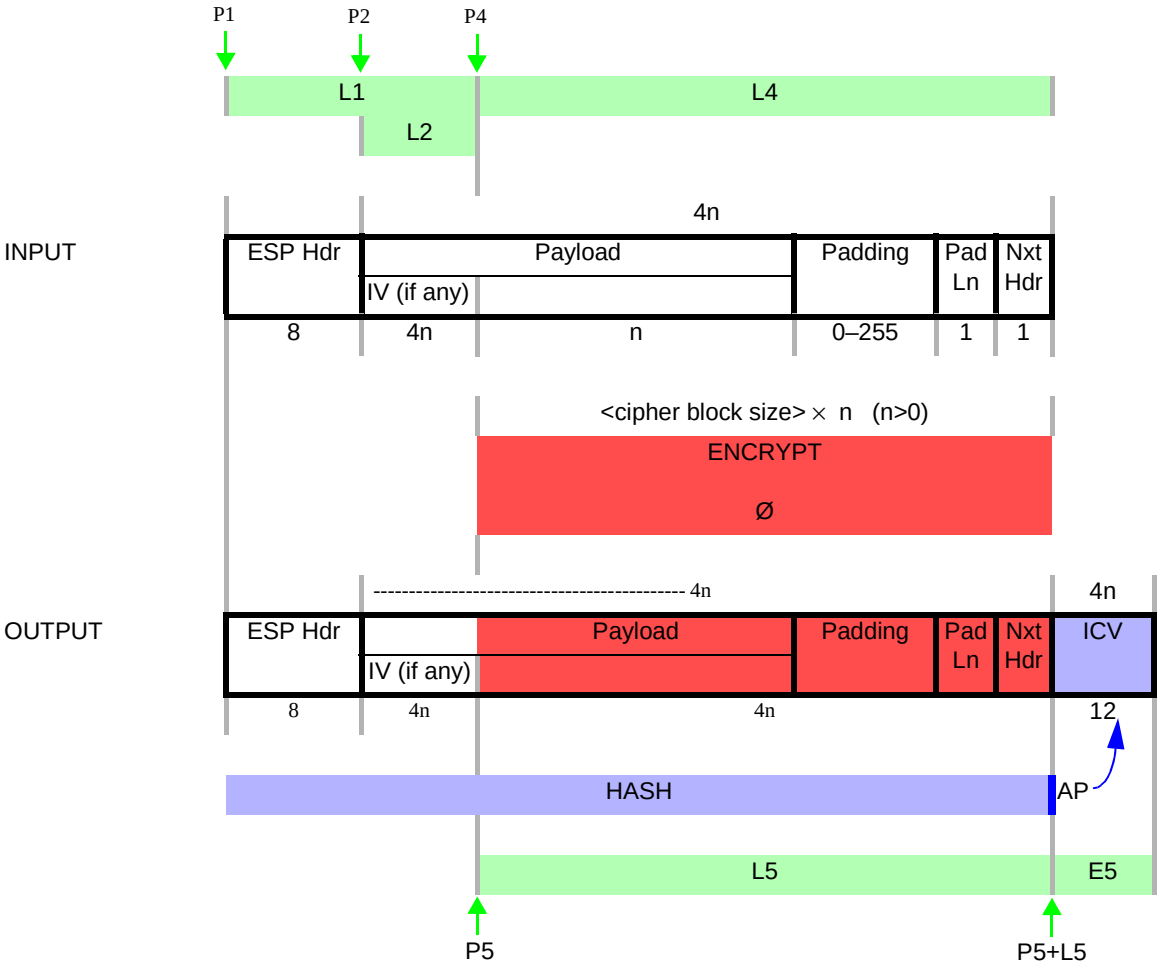


Figure 27-17. IPsec ESP Outbound Packet Pointer Diagram

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

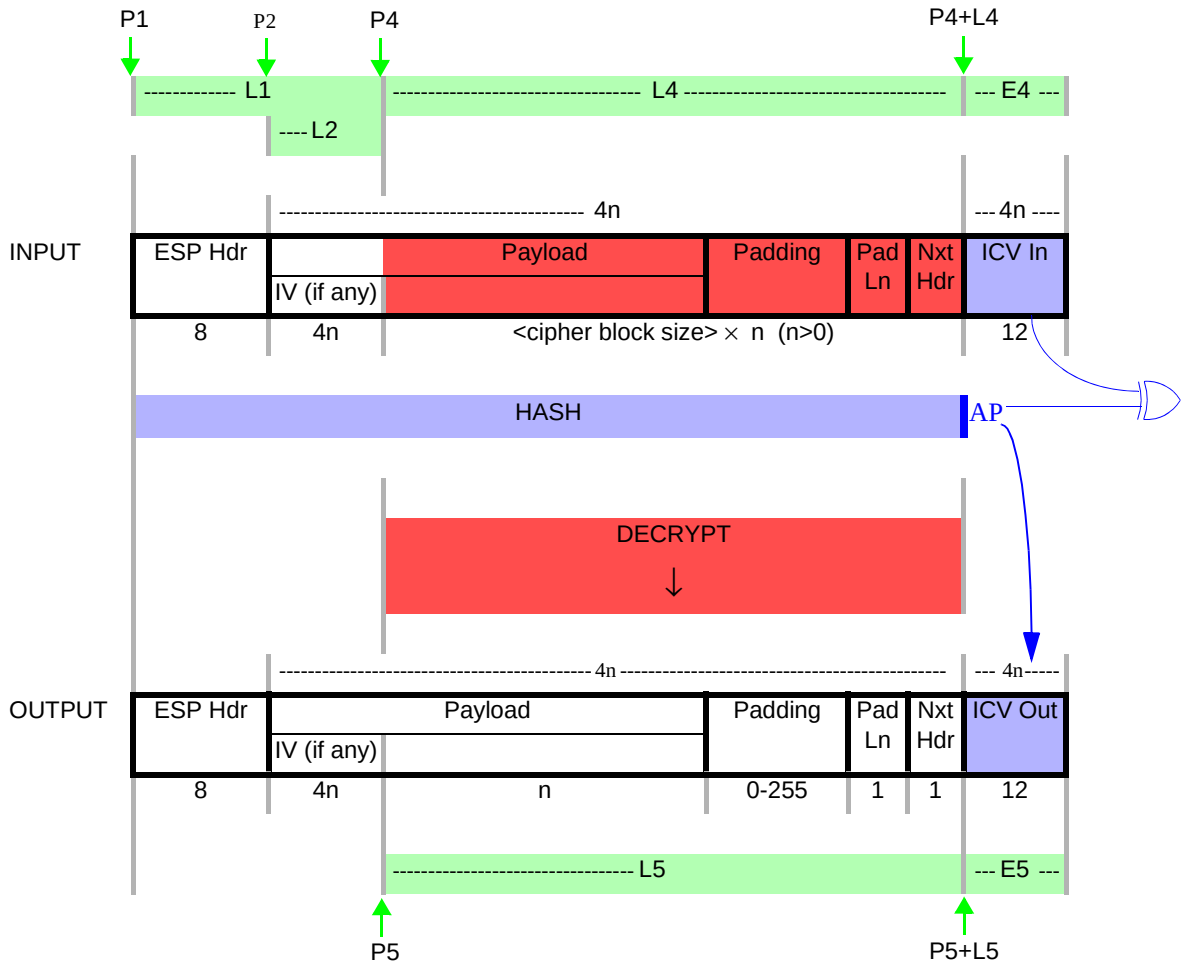


Figure 27-19. IPsec ESP Inbound Packet Pointer Diagram

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 (that is, 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

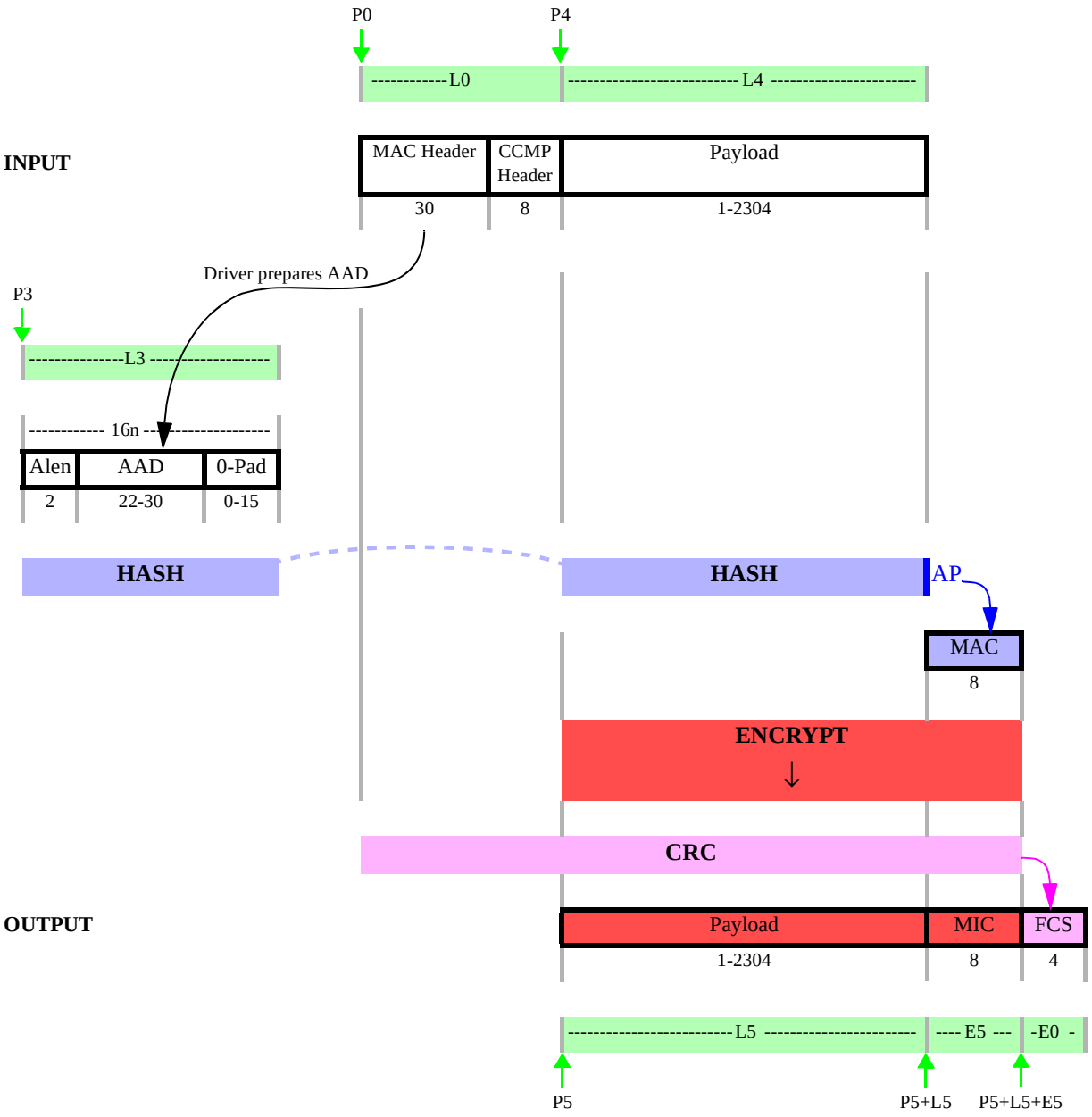


Figure 27-21. IEEE 802.11i (AES-CCM) Outbound Packet Pointer Diagram

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

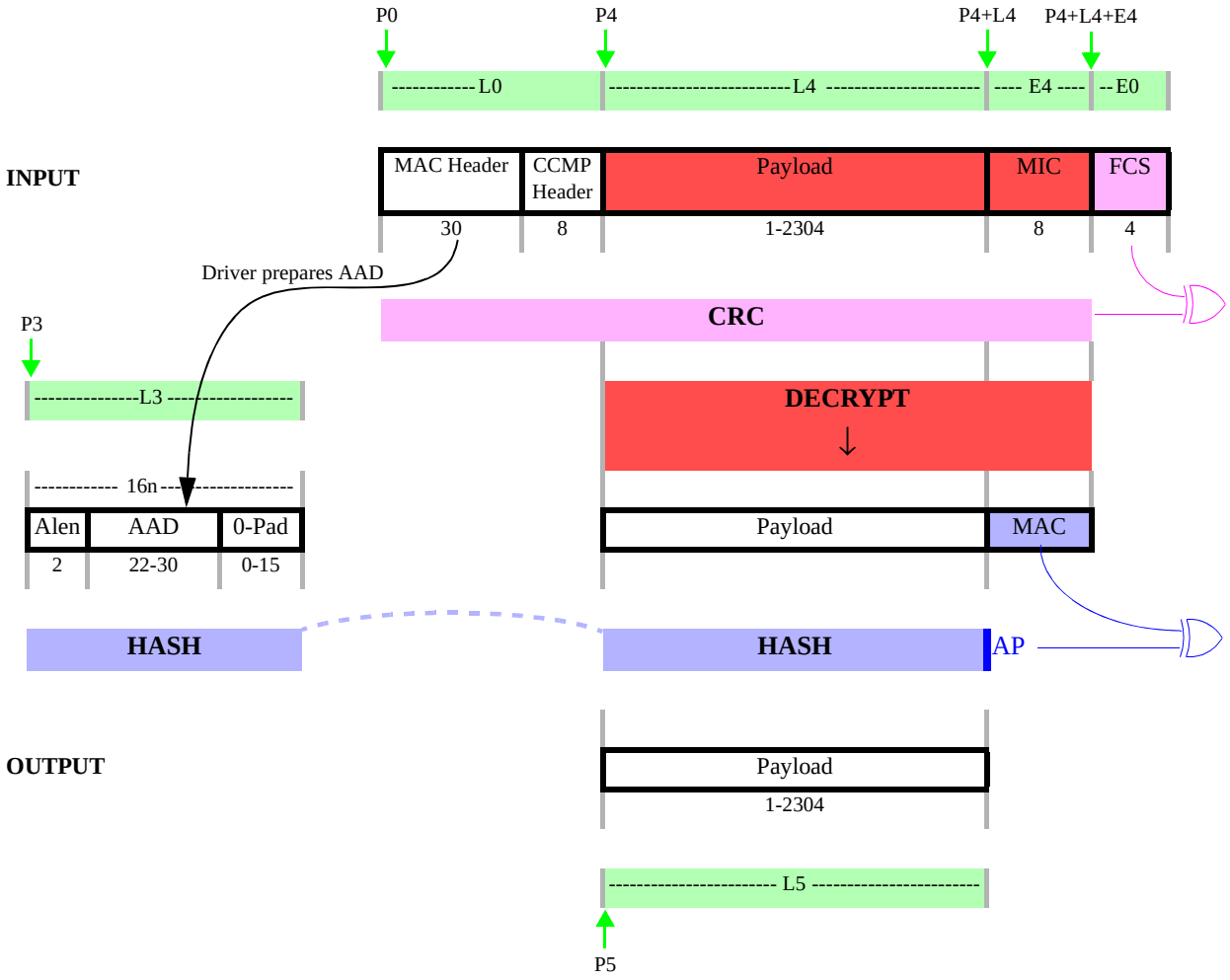


Figure 27-23. IEEE 802.11i (AES-CCM) Inbound Packet Pointer Diagram

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

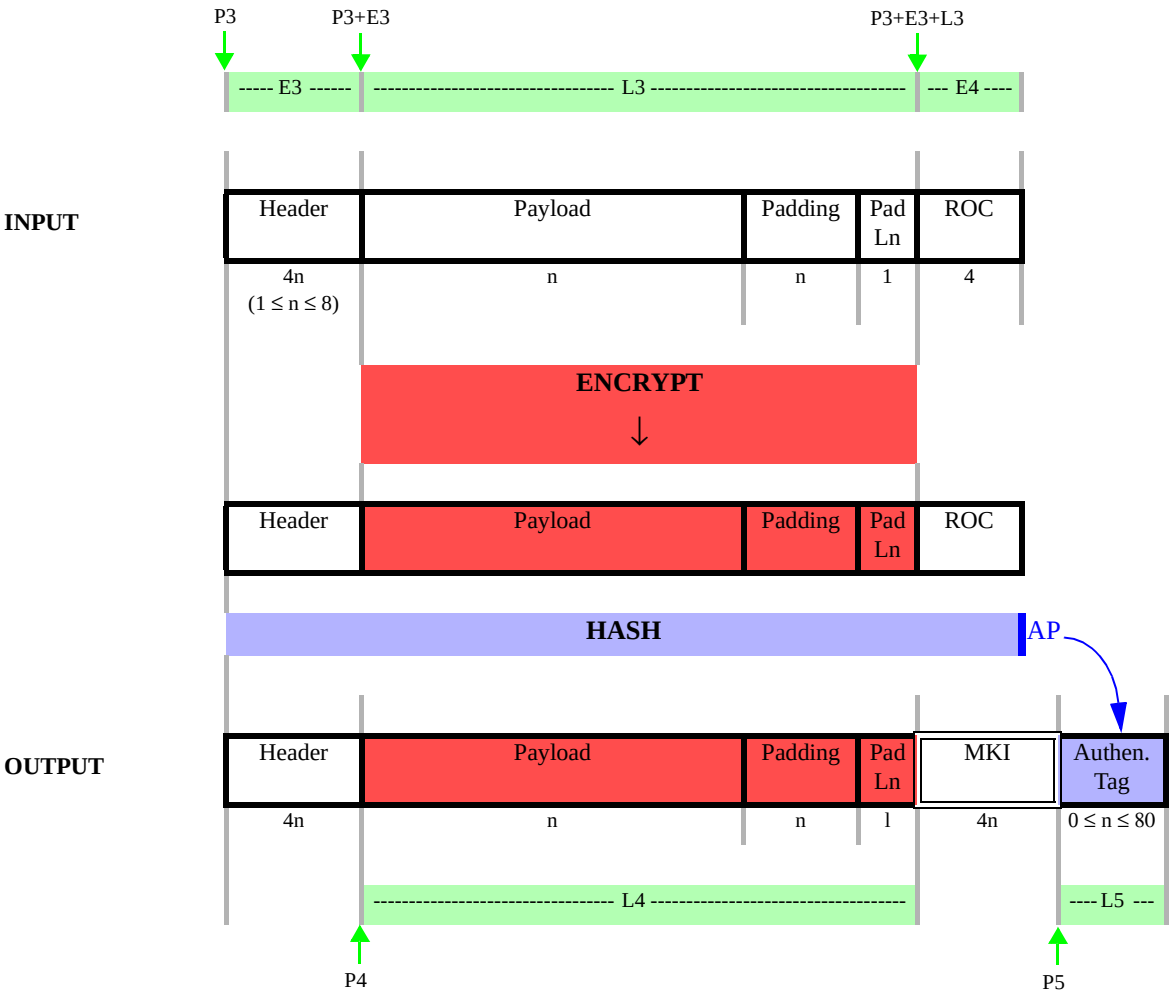


Figure 27-25. SRTP Outbound Packet Pointer Diagram

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

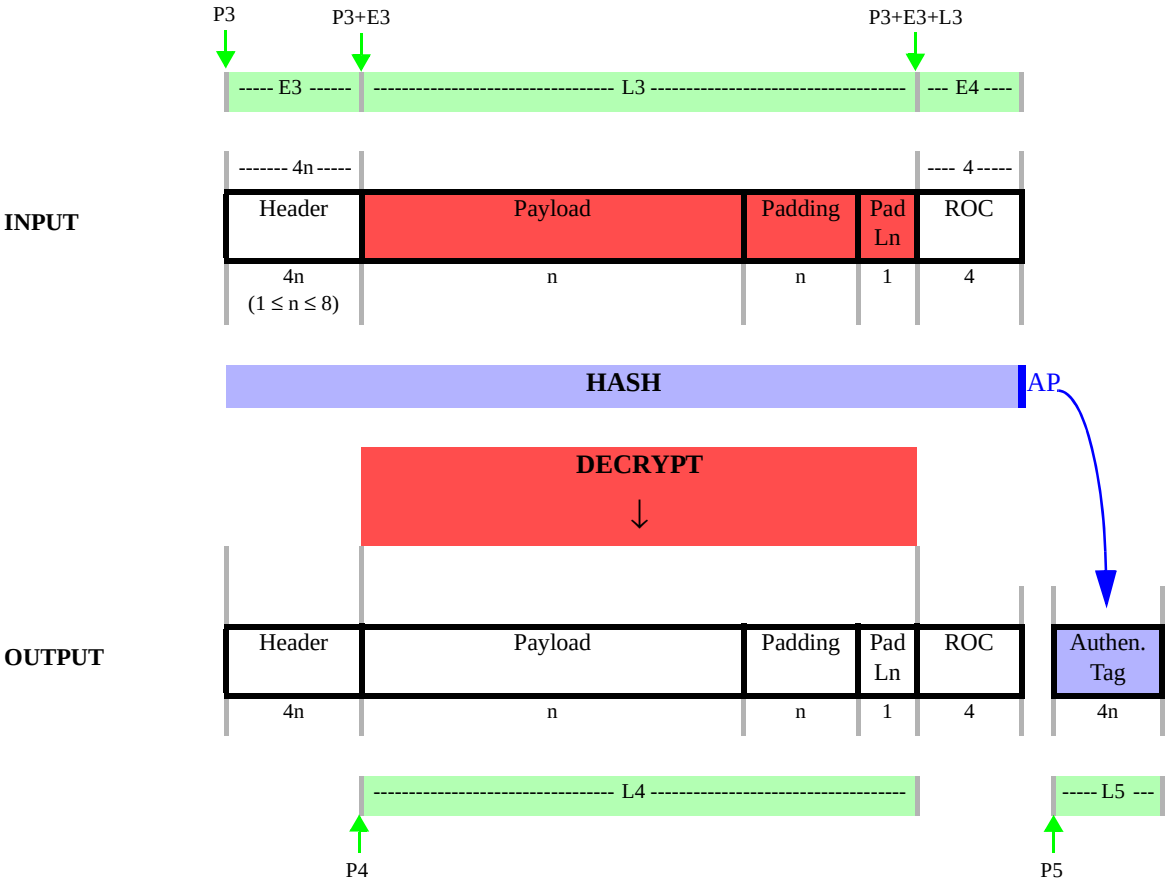


Figure 27-27. SRTP Inbound without ICV Compare Packet Pointer Diagram

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

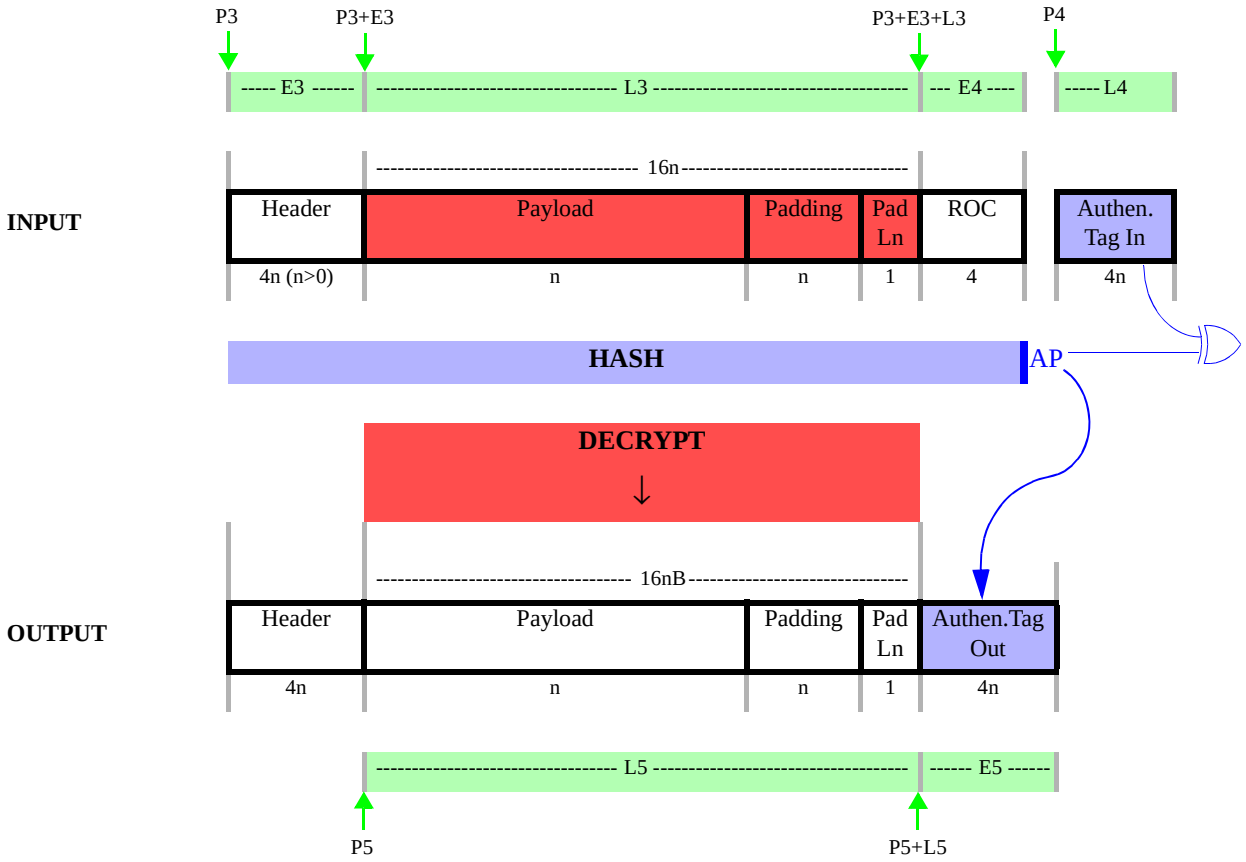


Figure 27-29. SRTP Inbound with ICV Compare Packet Pointer Diagram

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

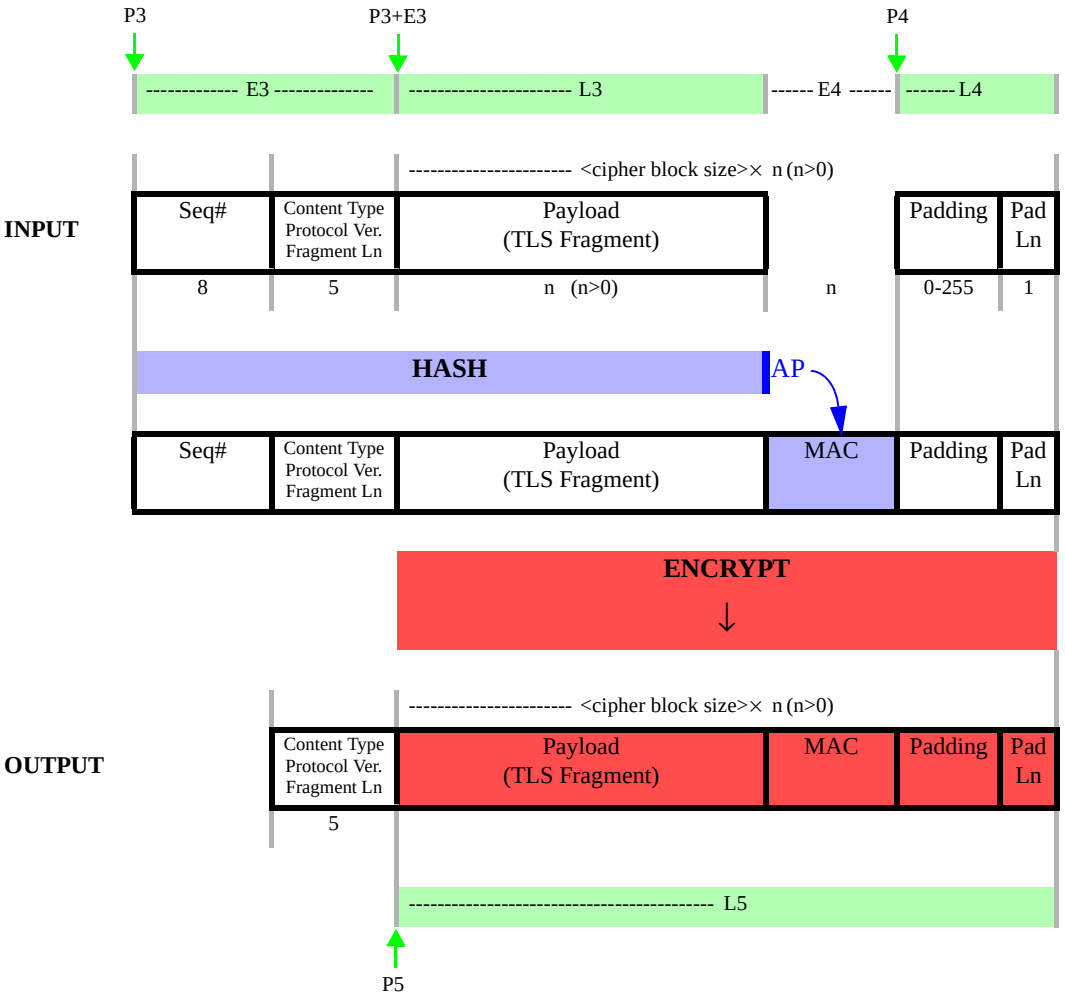


Figure 27-34. TLS / SSL Block Cipher Outbound Packet Pointer Diagram

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

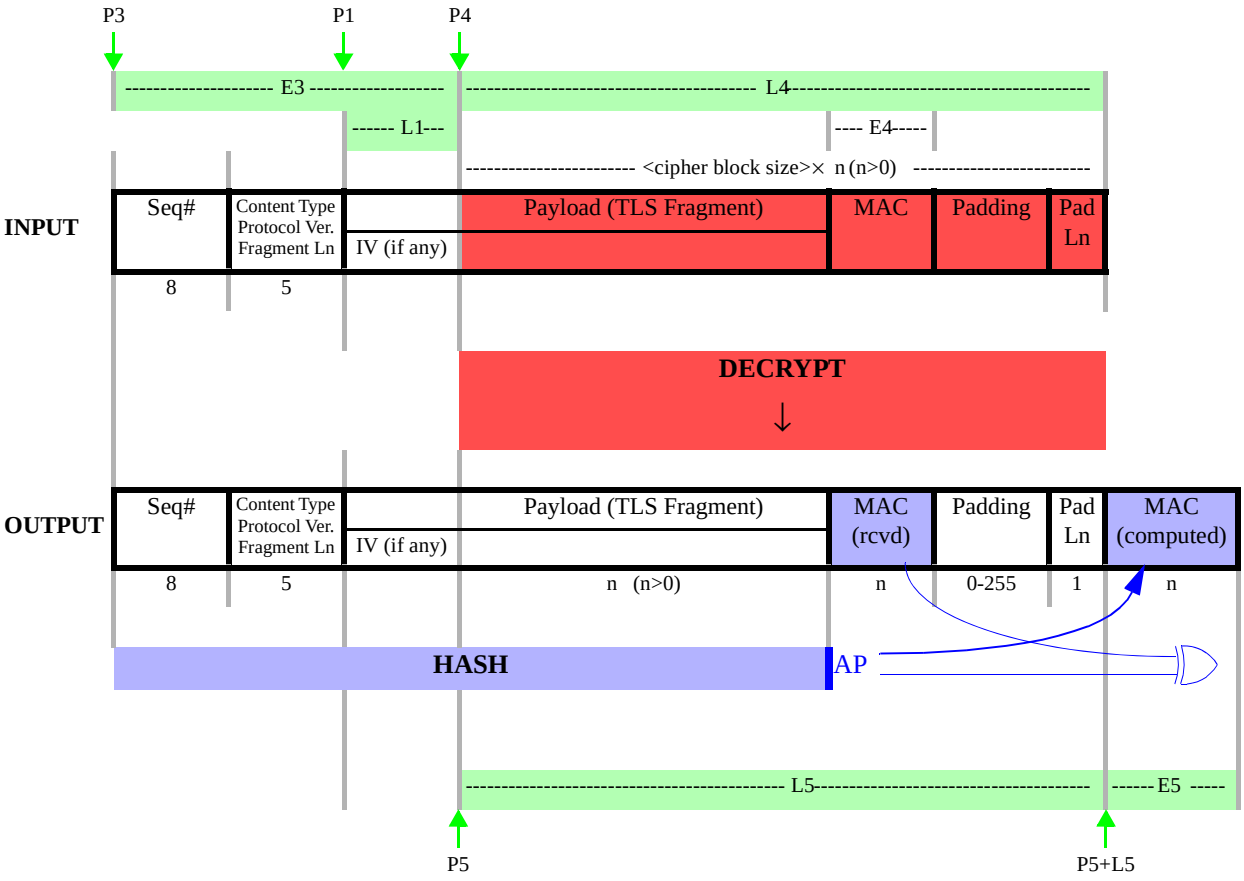


Figure 27-36. TLS / SSL Block Cipher Inbound Packet Pointer Diagram

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 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.

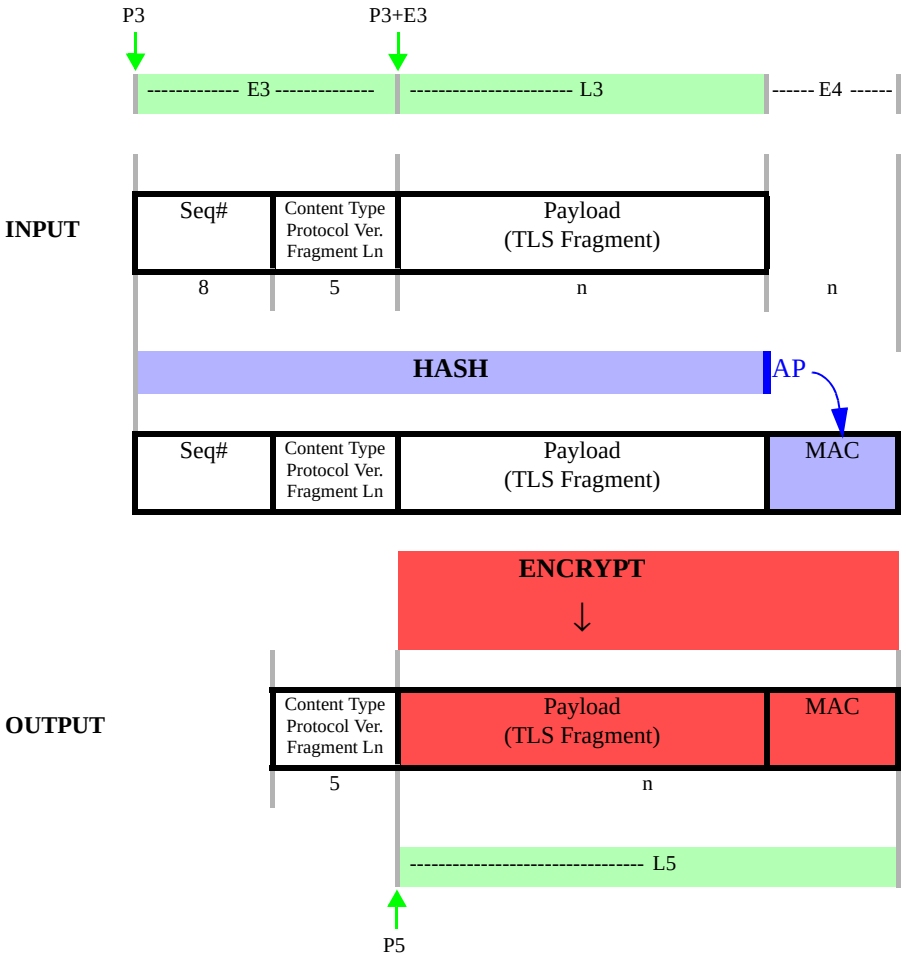


Figure 27-38. TLS / SSL Stream Cipher Outbound Packet Pointer Diagram

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

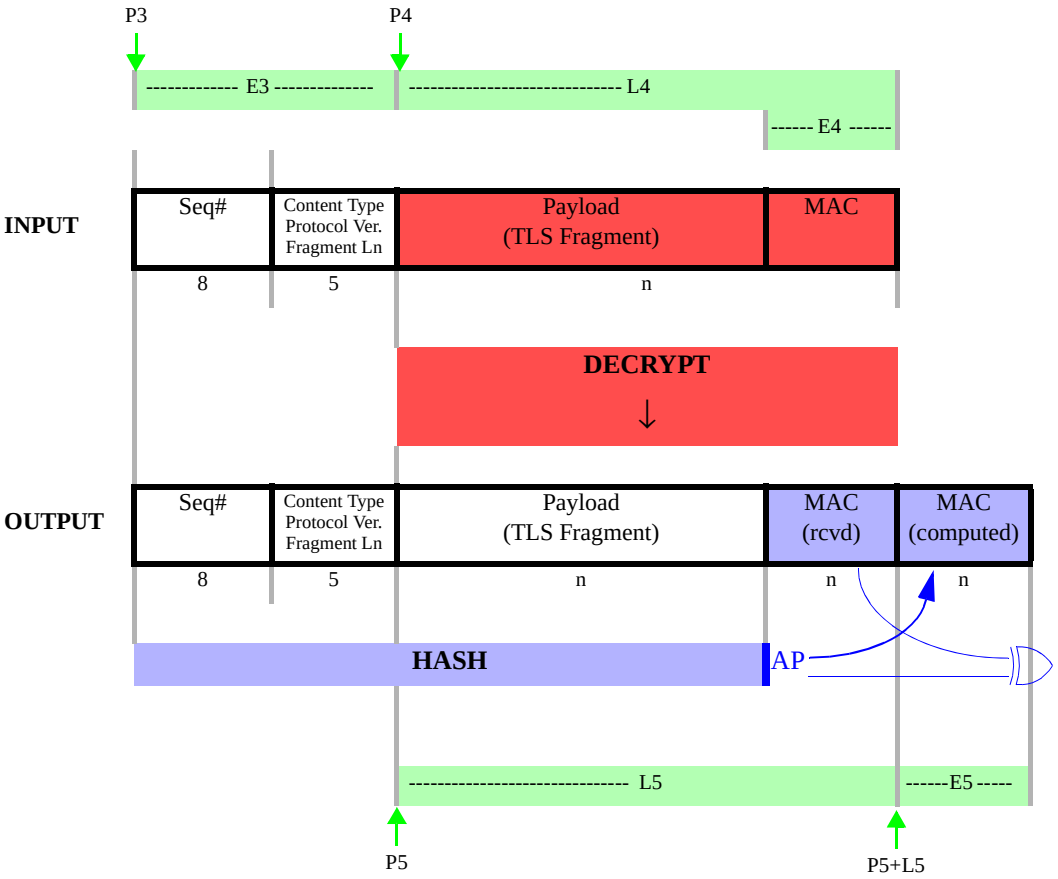


Figure 27-40. TLS / SSL Stream Cipher Inbound Packet Pointer Diagram

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

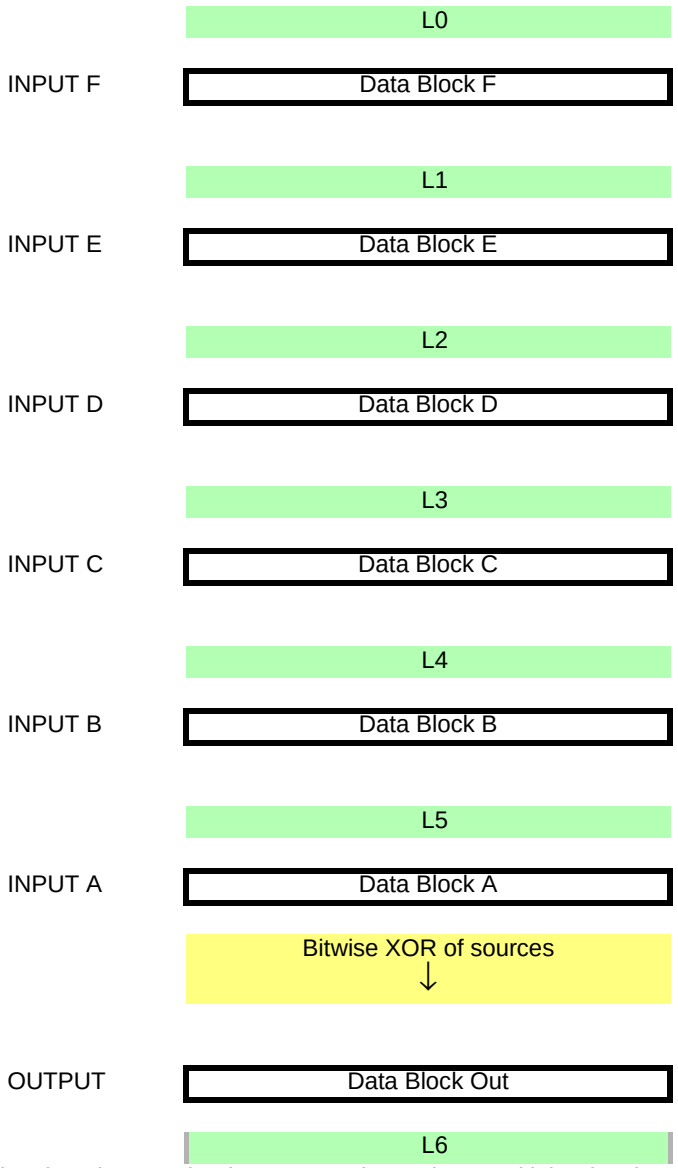
27.7.1.2.27 Descriptor Type 1010_1: raid_xor

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

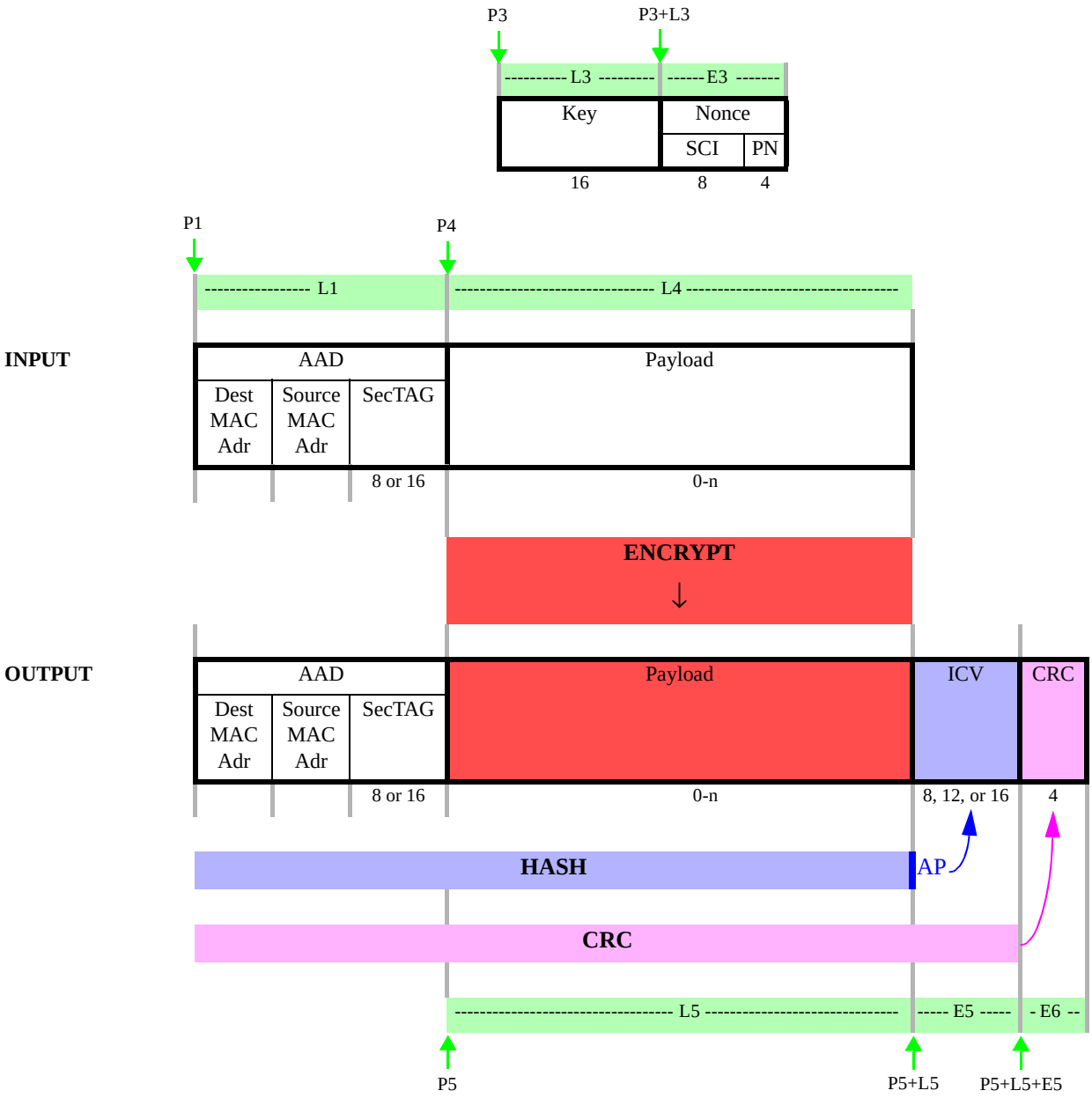


Figure 27-45. AES_GCM Outbound MACSec Packet Pointer Diagram

\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

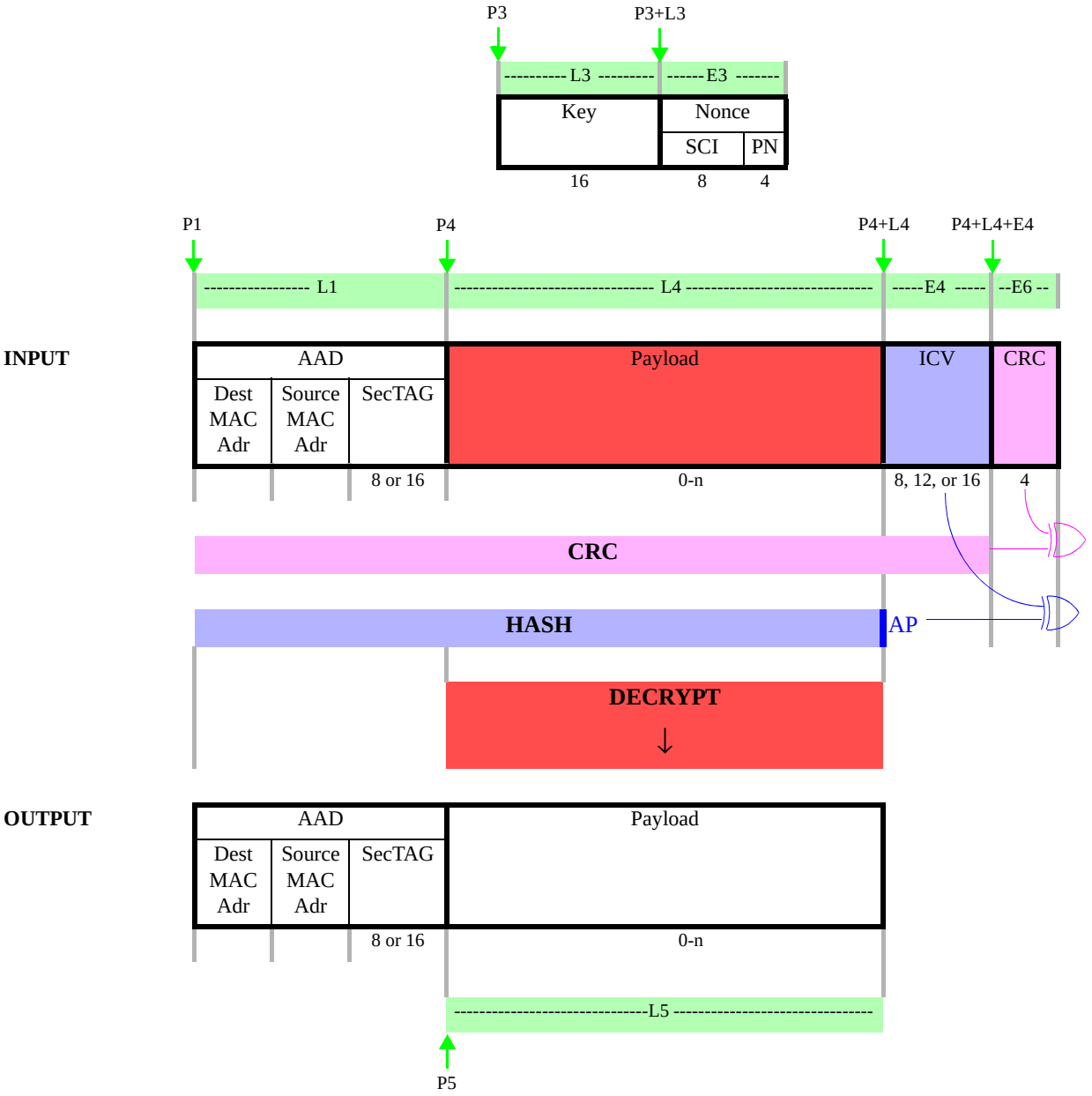


Figure 27-47. AES_GCM Inbound MACSec Packet Pointer Diagram

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.

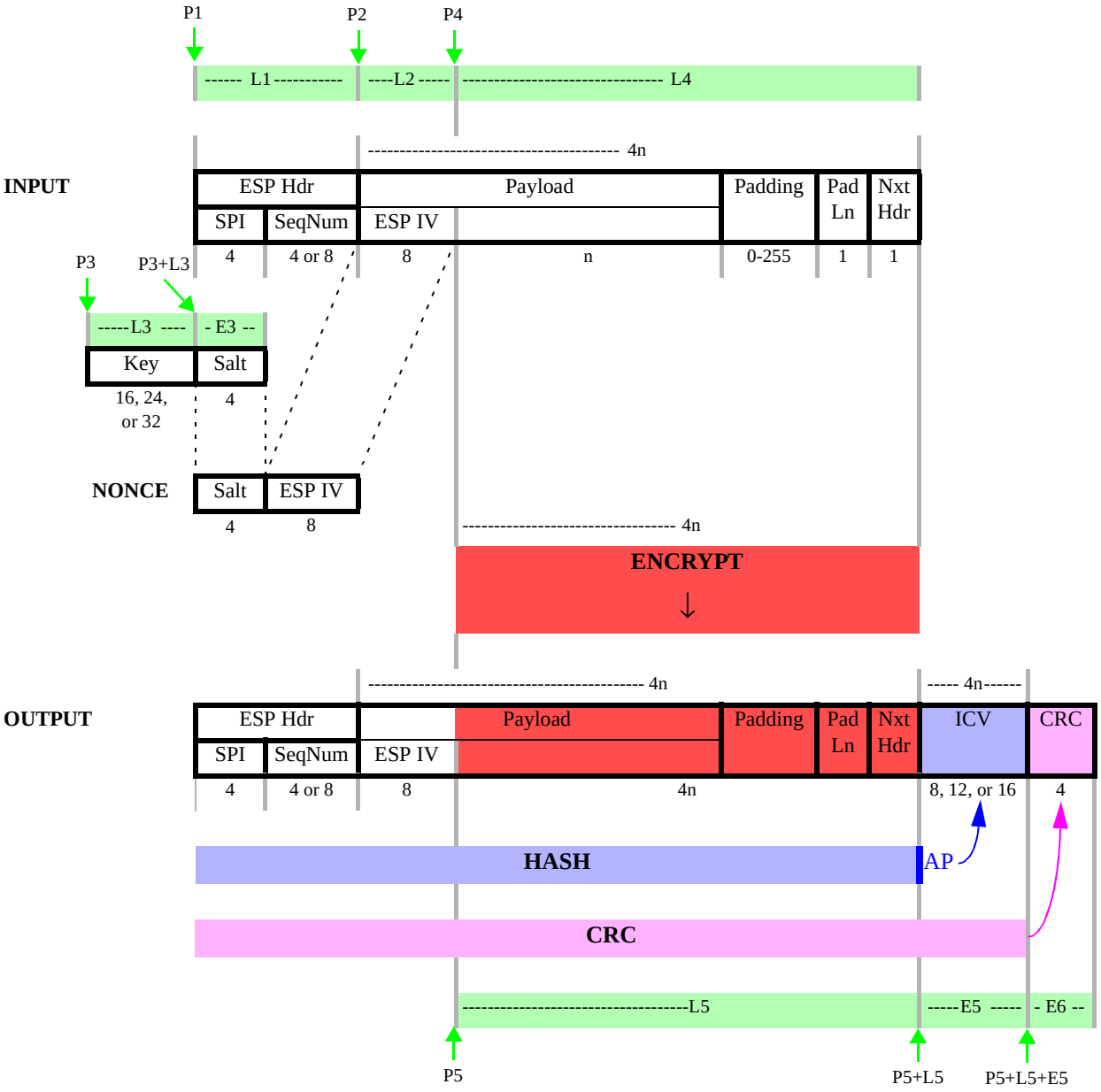


Figure 27-49. AES_GCM Outbound IPsec Packet Pointer Diagram

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

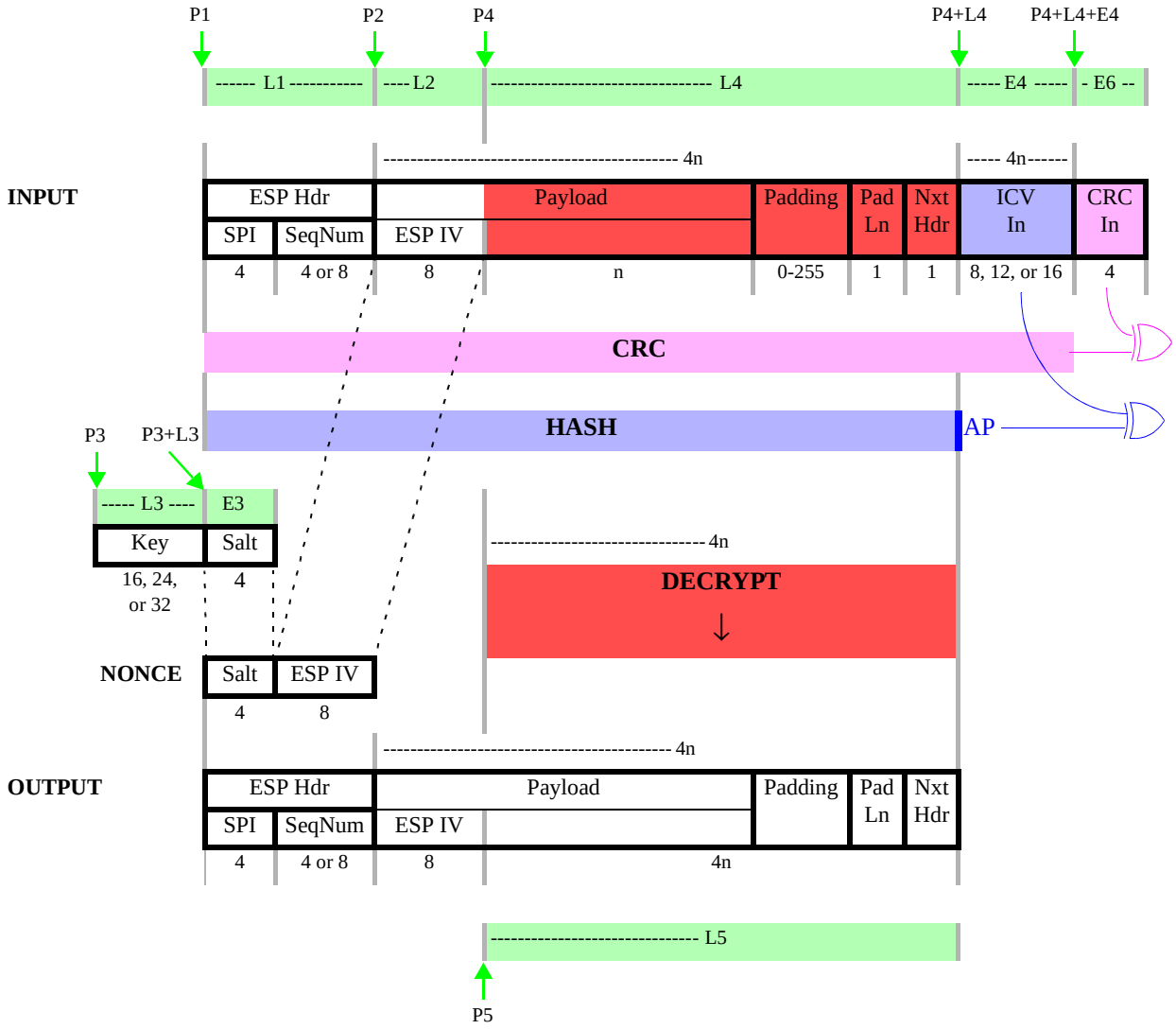


Figure 27-51. AES_GCM Inbound IPsec Packet Pointer Diagram

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

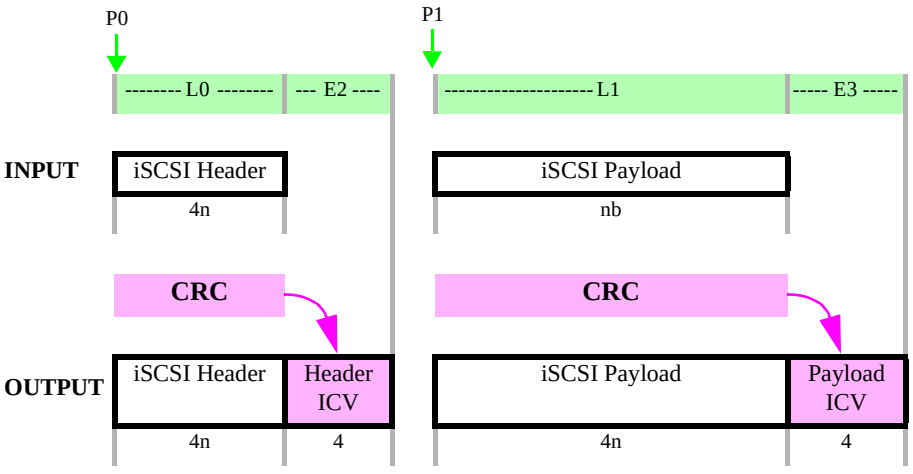


Figure 27-53. SCSI dbl_crc Outbound Packet Pointer Diagram

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

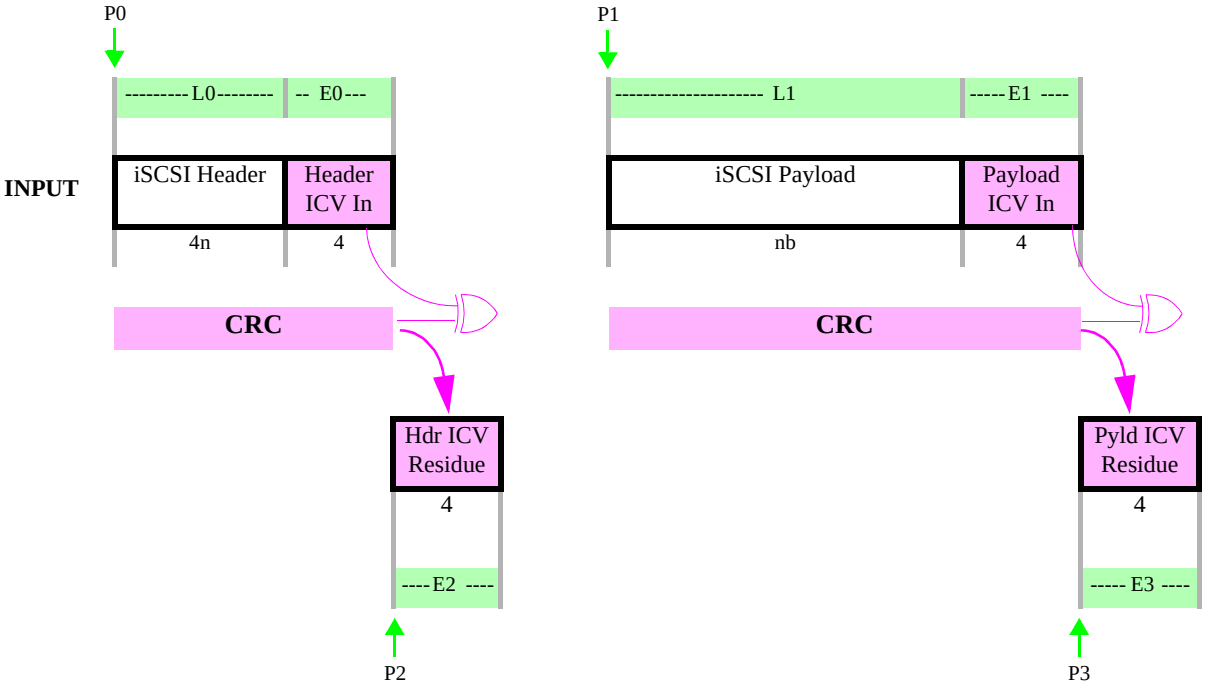


Figure 27-55. iSCSI dbl_crc Inbound Packet Pointer Diagram

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 27-177. 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.

The SEC EU Assignment Status Register shows which channels own which EUs at a particular moment in time. However, there is a missing value in the EU Assignment Register (EUASR) such that the STEU (Snow3G Execution Unit) field in the EUASR is not updated properly, and always reads back as 0xF. As a result, it is not possible to know which (if any) channel is using the STEU. When operating the SEC in descriptor mode, the user does not have any reason to care which channel owns the STEU at a particular moment in time. STEU descriptors can be dispatched to any channel, and the SEC controller ensures that all channels get access to the STEU.

The only conditions under which the EU Assignment Register is useful is when mixing descriptor based operations with direct access mode (slave mode) operations. Direct access mode operations involve software writing directly to EU registers, bypassing the channels entirely. This is not a recommended mode of operation, but if there is a compelling reason to use direct access mode for STEU operations, only use direct access mode. If not, software must ensure that all STEU descriptors have completed before performing an STEU direct access. The recommendation is to use descriptors for all STEU operations. If the alternative is required, use direct access mode for all STEU operations. Do not switch between the two modes.

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.	

Table 27-61. EUASR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
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.

Table 27-63. CIER Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
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.
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.

Table 27-63. CIER Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
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.	
RNGU Err 17	0	**RNGU Error Interrupt** Indicates whether the RNGU generated an error.	0 Error interrupt disabled. 1 Error interrupt enabled.

Table 27-63. CIER Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
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 27-192. 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 27-189).

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 27-189).

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 27-189).

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 27-189).

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. **Note:** Always enable the timer to enable detection of stalled channels. This is a one-time configuration after device reset.	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.	

Table 27-66. CCR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
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.
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

Table 27-67. Writeback Options (Continued)

CCR Field				Descriptor Header	Writeback Action for a Descriptor Completing Without Error
AWSE	CDWE	IWSE	NT	DN	
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.

Table 27-69. CSR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
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.
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.

Table 27-69. CSR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
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.
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

Table 27-71. CHN_STATE Field Values (Continued)

Value	Channel State
0x0C	WRITE_RESET_PRI
0x0D	RELEASE_PRI_CHA
0x0E	WRITE_RESET_SEC
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

Table 27-71. CHN_STATE Field Values (Continued)

Value	Channel State
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.

Table 27-72. CDPR Bit Field Descriptions (Continued)

Bits	Reset	Description
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 27-100.

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 27-217).	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 27-217).	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 27-189).	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 27-189)	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 27-189). 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 27-202) 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 27-227).	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 27-227).	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 27-189).	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 27-189)	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 27-189). 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 27-202) 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 27-226). 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 27-240).	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 27-240).	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 27-189).	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 27-189)	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 27-189). 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 27-202) 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 27-189).	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 27-189)	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 27-274).

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 27-189). 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 27-202) 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 27-179) 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 27-271).

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 27-272).

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 27-189).	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 27-189).	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 27-189). 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 27-202) 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 27-267). 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 27-269). 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 27-277 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 27-283).	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 27-283).	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 27-189).	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 27-189).	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 27-189). 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 27-202) 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 27-282).

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 27-302).	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 27-302).	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 27-189).	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 27-189).	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 27-305).

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 27-189). 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 27-202) 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 27-300). 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 27-303), 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 27-315).	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 27-315).	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 27-189).	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 27-189).	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 27-318).

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 27-189). 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 27-202) 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 27-316), 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 27-189).	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 27-189 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 27-95). 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 27-94). 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 27-202) 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 27-94), 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.