MSC8144E Reference Manual

Quad Core Media Signal Processor

MSC8144ERM
Rev 3, July 2009

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

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

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

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

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

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

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

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.



Freescale™, the Freescale logo, CodeWarrior, QUICC Engine, PowerQUICC, and StarCore are trademarks of Freescale Semiconductor, Inc. RapidIO is a trademark of the RapidIO Trade Association. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2005, 2009.

MSC8144ERM
Rev. 3
6/2009

Overview 1
SC3400 Core Overview 2
External Signals 3
Chip-Level Arbitration and Switching System (CLASS) 4
Reset 5
Boot Program 6
Clocks 7
General Configuration Registers 8
Memory Map 9
MSC8144 SC3400 DSP Subsystem 10
Internal Memory Subsystem 11
DDR-SDRAM Controller 12
Interrupt Processing 13
Direct Memory Access (DMA) Controller 14
PCI 15
Serial RapidIO™ Controller 16
RapidIO Interface Dedicated DMA Controller 17
QUICC Engine™ Subsystem 18
TDM Interface 19
UART 20
Timers 21
GPIO 22
Hardware Semaphores 23
I^2C 24
Debugging, Profiling, and Performance Monitoring 25
Security Engine (SEC) 26

1 Overview
2 SC3400 Core Overview
3 External Signals
4 Chip-Level Arbitration and Switching System (CLASS)
5 Reset
6 Boot Program
7 Clocks
8 General Configuration Registers
9 Memory Map
10 MSC8144 SC3400 DSP Subsystem
11 Internal Memory Subsystem
12 DDR-SDRAM Controller
13 Interrupt Processing
14 Direct Memory Access (DMA) Controller
15 PCI
16 Serial RapidIO™ Controller
17 RapidIO Interface Dedicated DMA Controller
18 QUICC Engine™ Subsystem
19 TDM Interface
20 UART
21 Timers
22 GPIO
23 Hardware Semaphores
24 I^2C
25 Debugging, Profiling, and Performance Monitoring
26 Security Engine (SEC)

Contents

About This Book

Before Using This Manual—Important Note xlii
Audience and Helpful Hints xlii
Notational Conventions and Definitions xliii
Conventions for Registers xliv
Organization xliv
Other MSC8144E Documentation xlvii
Further Reading xlvii

1 Overview

1.1 Features 1-2
1.2 Block Diagram 1-9
1.3 Architecture 1-10
1.4 StarCore SC3400 DSP Subsystem 1-10
1.4.1 StarCore SC3400 DSP Core 1-11
1.4.2 L1 Instruction Cache 1-12
1.4.3 L1 Data Cache 1-13
1.4.4 Memory Management Unit (MMU) 1-14
1.4.5 Debug and Profiling Unit (DPU) 1-15
1.5 Chip-Level Arbitration and Switching System (CLASS) 1-16
1.6 Memory System 1-17
1.6.1 M2 Memory 1-17
1.6.2 M3 Memory 1-17
1.6.3 L2 Instruction Cache 1-18
1.7 Clocks 1-18
1.8 DDR Controller (DDRC) 1-19
1.9 DMA Controller 1-20
1.10 TDM 1-20
1.11 QUICC Engine Subsystem 1-21
1.11.1 Ethernet Controllers 1-22
1.11.2 ATM Controller 1-24
1.11.3 Serial Peripheral Interface (SPI) 1-24
1.12 PCI 1-24
1.13 Serial RapidIO Subsystem 1-25
1.13.1 Serial RapidIO and Host Interactions 1-25

1.13.2 RapidIO Messaging Unit (RMU) Operation 1-26
1.14 Global Interrupt Controller (GIC) 1-27
1.15 UART 1-27
1.16 Timers 1-28
1.17 Hardware Semaphores 1-28
1.18 Virtual Interrupts 1-28
1.19 I^2C Interface 1-28
1.20 GPIOs 1-28
1.21 Boot Options 1-29
1.22 JTAG 1-29
1.23 Security Engine (SEC) 1-29
1.24 Development Environment 1-29
1.25 Application Software 1-30
1.26 Example Applications 1-32
1.26.1 Application 1 1-33
1.26.2 Application 2 1-34
1.26.3 Application 3 1-35
1.26.4 Application 4 1-36
1.26.5 Application 5 1-37

2 SC3400 Core Overview

2.1 Architecture 2-2
2.1.1 Data Arithmetic Logic Unit (Data ALU) 2-3
2.1.1.1 Data Registers 2-4
2.1.1.2 Multiply-Accumulate (MAC) Unit 2-4
2.1.1.3 Bit-Field Unit (BFU) 2-4
2.1.1.4 Back Trace Registers (BTR) 2-4
2.1.2 Address Generation Unit (AGU) 2-4
2.1.2.1 Stack Pointer Registers 2-6
2.1.2.2 Bit Mask Unit (BMU) 2-6
2.1.3 Program Sequencer Unit (PSEQ) 2-7
2.1.4 Resource Stall Unit (RSU) 2-7
2.1.5 On-Chip Emulator (OCE) 2-7
2.2 Programming Model 2-8
2.2.1 AGU Programming Model 2-8
2.2.2 Data Arithmetic Logic Programming Model 2-11
2.2.3 Program Control Unit Programming Model 2-12
2.3 Instruction Set Overview 2-13
2.4 Additional Programming Considerations 2-22

3 External Signals

3.1 Power Signals . . . 3-5
3.2 Clock Signals . . . 3-6
3.3 Reset and Configuration Signals . . . 3-7
3.4 Memory Controller . . . 3-11
3.5 Serial RapidIO Signals . . . 3-12
3.6 PCI Signals . . . 3-14
3.7 Ethernet Signals . . . 3-27
3.8 ATM UTOPIA Signals . . . 3-32
3.9 Serial Peripheral Interface (SPI) Signal Summary . . . 3-41
3.10 GPIO/Maskable Interrupt Signal Summary . . . 3-42
3.11 Timer Signals . . . 3-48
3.12 UART Signals . . . 3-49
3.13 I^2C Signals . . . 3-50
3.14 TDM Signals . . . 3-50
3.15 Other Interrupt Signals . . . 3-58
3.16 OCE Event and JTAG Test Access Port Signals . . . 3-58

4 Chip-Level Arbitration and Switching System (CLASS)

4.1 CLASS Features . . . 4-3
4.2 Functional Description . . . 4-4
4.2.1 Expander Module and Transaction Flow . . . 4-4
4.2.2 Multiplexer and Arbiter Module . . . 4-5
4.2.2.1 Atomic Stall Unit (ASU) . . . 4-5
4.2.2.1.1 CLASS Arbiter . . . 4-5
4.2.2.1.2 Weighted Arbitration . . . 4-6
4.2.2.1.3 Late Arbitration . . . 4-6
4.2.2.1.4 Priority Masking . . . 4-6
4.2.2.1.5 Auto Priority Upgrade . . . 4-6
4.2.2.2 CLASS Multiplexer . . . 4-7
4.2.3 Normalizer Module . . . 4-7
4.2.4 CLASS Control Interface (CCI) . . . 4-7
4.3 CLASS Error Interrupts . . . 4-7
4.4 CLASS Debug Profiling Unit . . . 4-9
4.4.1 Profiling . . . 4-9
4.4.2 Watch Point Unit . . . 4-10
4.4.3 Event Selection . . . 4-10
4.4.4 Debug and Profiling Events . . . 4-12
4.5 CLASS Reset . . . 4-12
4.5.1 Soft Reset . . . 4-12
4.5.2 Hard Reset . . . 4-12

4.6 Limitations 4-13
4.7 Programming Model 4-14
4.7.1 CLASS MBus Target Configuration Registers (CnMTCRx) 4-15
4.7.2 CLASS Priority Mapping Registers (CnPMRx) 4-17
4.7.3 CLASS Priority Auto Upgrade Value Registers (CnPAVRx) 4-19
4.7.4 CLASS Priority Auto Upgrade Control Registers (CnPACRx) 4-20
4.7.5 CLASS 1 Error Address Registers (C1EARx) 4-21
4.7.6 CLASS 1 Error Extended Address Registers (C1EEARx) 4-22
4.7.7 CLASS Initiator Profiling Configuration Registers (CnIPCRx) 4-24
4.7.8 CLASS Initiator Watch Point Control Registers (CnIWPCRx) 4-25
4.7.9 CLASS Arbitration Weight Registers (CnAWRx) 4-26
4.7.10 CLASS IRQ Status Register (CnISR) 4-27
4.7.11 CLASS IRQ Enable Register (CnIER) 4-28
4.7.12 CLASS Target Profiling Configuration Register (CnTPCR) 4-29
4.7.13 CLASS Profiling Control Register (CnPCR) 4-31
4.7.14 CLASS Watch Point Control Registers (CnWPCR) 4-32
4.7.15 CLASS Watch Point Access Configuration Register (CnWPACR) 4-34
4.7.16 CLASS Watch Point Extended Access Configuration Register (CnWPEACR) 4-36
4.7.17 CLASS Watch Point Address Mask Registers (CnWPAMR) 4-38
4.7.18 CLASS Profiling Time-Out Registers (CnPTOR) 4-39
4.7.19 CLASS Target Watch Point Control Registers (CnTWPCR) 4-40
4.7.20 CLASS Profiling IRQ Status Register (CnPISR) 4-41
4.7.21 CLASS Profiling IRQ Enable Register (CnPIER) 4-42
4.7.22 CLASS Profiling Reference Counter Register (CnPRCR) 4-42
4.7.23 CLASS Profiling General Counter Registers (CnPGCRx) 4-43
4.7.24 CLASS General Purpose Register (CnGPR) 4-44
4.7.25 CLASS Arbitration Control Register (CnACR) 4-45
4.7.26 CLASS1 Start Address Decoder x (C1SADx) 4-46
4.7.27 CLASS1 End Address Decoder x (C1EADx) 4-47
4.7.28 CLASS1 Attributes Decoder x (C1ATDx) 4-48

5 Reset

5.1 Reset Operations 5-1
5.1.1 Reset Sources 5-2
5.1.2 Reset Actions 5-2
5.1.3 Power-On Reset Flow 5-3
5.1.4 Detailed Power-On Reset Flow 5-4
5.1.5 $\overline{\text{HRESET}}$ Flow 5-6
5.1.6 $\overline{\text{SRESET}}$ Flow 5-6
5.2 Reset Configuration 5-7

5.2.1 Reset Configuration Signals 5-7
5.2.2 Reset Configuration Words Source 5-7
5.2.3 CLKIN Frequency Range Signal 5-8
5.2.4 Reset Configuration Load Fail Signal 5-8
5.2.5 Selecting Reset Configuration Input Signals 5-8
5.2.6 Reset Configuration Words 5-9
5.2.7 Loading The Reset Configuration Words 5-9
5.2.7.1 Loading From an I2C EEPROM 5-9
5.2.7.1.1 Using The Boot Sequencer For Reset Configuration 5-10
5.2.7.1.2 EEPROM Addressing 5-10
5.2.7.1.3 EEPROM Data Format In Reset Configuration Mode 5-10
5.2.7.2 Loading Multiple Devices From a Single I^2C EEPROM 5-11
5.2.7.2.1 Multiple Device External Reset Logic 5-12
5.2.7.2.2 Multiple Device ROM Code 5-12
5.2.7.3 Single Device Loading From I2C EEPROM 5-13
5.2.7.4 Loading Reduced RCW From External Pins 5-13
5.2.7.5 Hard Coded Reset Configuration Word High Fields Values 5-14
5.2.7.6 External Reset Configuration Word Low 5-15
5.2.7.7 External Reset Configuration Word High Fields Values 5-16
5.3 Reset Programming Model 5-17
5.3.1 Reset Configuration Word Low Register (RCWLR) 5-17
5.3.2 Reset Configuration Word High Register (RCWHR) 5-19
5.3.3 Reset Status Register (RSR) 5-21
5.3.4 Reset Protection Register (RPR) 5-23
5.3.5 Reset Control Register (RCR) 5-24
5.3.6 Reset Control Enable Register (RCER) 5-25

6 Boot Program

6.1 Functional Description 6-2
6.2 Private Configuration 6-3
6.3 M3 Initialization Delay 6-3
6.4 Shared Configuration 6-4
6.4.1 Multi Device Support for the I^2C Bus 6-5
6.4.2 Example Configuration 6-6
6.5 Boot Modes 6-10
6.5.1 I^2C EEPROM 6-10
6.5.2 Ethernet 6-14
6.5.2.1 DHCP Client 6-16
6.5.2.2 TFTP Client 6-17
6.5.2.3 Boot File Format 6-17
6.5.3 Serial RapidIO Interconnect 6-19

6.5.3.1 Serial RapidIO Without I2C Support 6-19
6.5.3.2 Serial RapidIO with I2C Support 6-20
6.5.4 PCI 6-20
6.5.5 SPI 6-21
6.6 Jump to User Code 6-21
6.7 System after Boot 6-22
6.8 Boot Errors 6-22

7 Clocks
7.1 Clock Control Logic 7-3
7.1.1 Clock Modes 7-3
7.1.2 Clock Locking 7-11
7.1.2.1 Reset Initialization 7-11
7.1.2.2 Clock Reprogramming 7-11
7.2 Clock Programming Model 7-12
7.2.1 System Clock Control Register (SCCR) 7-12
7.2.2 PLL Clock Mode Register 0 (PCMR0B/PCMR0F) 7-14
7.2.3 PLL Clock Mode Register 1 (PCMR1B/PC1MR1F) 7-16
7.2.4 PLL Clock Mode Register 2 (PCMR2B/PCMR2F) 7-18
7.2.5 Dividers Clock Mode Register 0 (DCMR0B/DCMR0F) 7-20
7.2.6 Dividers Clock Mode Register 1 (DCMR1B, DCMR1F) 7-22
7.2.7 PLL Auxiliary Mode Register 0 (PAMR0B/PAMR0F) 7-23
7.2.8 PLL Auxiliary Mode Register 1 (PCMR1B/PC1MR1F) 7-24
7.2.9 PLL Auxiliary Mode Register 2 (PAMR2B/PAMR2F) 7-24
7.2.10 Values for Reprogramming Clock Modes 7-25

8 General Configuration Registers
8.1 Programming Model 8-1
8.2 Detailed Register Descriptions 8-2
8.2.1 General Configuration Register 1 (GCR1) 8-2
8.2.2 General Configuration Register 2 (GCR2) 8-3
8.2.3 General Status Register 1 (GSR1) 8-4
8.2.4 Lynx General Configuration Register (L_GCR) 8-6
8.2.5 DDR General Control Register (DDR_GCR) 8-7
8.2.6 RapidIO Control Register (RIO_CR) 8-8
8.2.7 SGMII Control Register (SGMII_CR) 8-9
8.2.8 QUICC Engine Control Register (QECTLR) 8-10
8.2.9 GPIO Input Enable Register (GIER) 8-11
8.2.10 System Part and Revision ID Register (SPRIDR) 8-12
8.2.11 General Configuration Register 4 (GCR4) 8-13
8.2.12 General Interrupt Register 1 (GIR1) 8-14

8.2.13 General Interrupt Register 1 (GIER1_x) 8-16
8.2.14 General Interrupt Register 2 (GIR2) 8-17
8.2.15 General Interrupt Enable Register 2 (GIER2_x) 8-19
8.2.16 General Interrupt Register 3 (GIR3) 8-21
8.2.17 General Interrupt Enable Register 3 (GIER3_x) 8-22

9 Memory Map

9.1 Shared Memory Address Space 9-1
9.2 SC3400 DSP core subsystem Internal Address Space 9-2
9.3 QUICC Engine Module Internal Address Space 9-2
9.4 CCSR Address Space 9-3
9.5 L2 ICache Address Space 9-5
9.6 SC3400 (Data) View of the System Address Space 9-5
9.7 Peripherals View of the System Address Space 9-6
9.8 PCI View of the System Address Space 9-6
9.9 Consolidated Memory Map 9-7

10 MSC8144E SC3400 DSP Subsystem

10.1 SC3400 DSP Core 10-2
10.2 Memory Management Unit (MMU) 10-3
10.3 Instruction Channel 10-4
10.4 Data Channel and Write Queue 10-5
10.5 Interrupt Processing 10-6
10.6 Real-Time Debug Support 10-6
10.7 Dual Timer 10-8
10.8 Interfaces 10-8
10.9 DSP Core Subsystem Operating States 10-8
10.9.1 Reset State 10-9
10.9.2 Execution State 10-9
10.9.3 Debug State 10-9
10.9.4 WAIT State 10-10
10.9.5 STOP State 10-10
10.9.6 State Transitions 10-11
10.9.6.1 Transition from Reset to Execution State 10-11
10.9.6.2 Transition from Reset to Debug State 10-11
10.9.6.3 Transition to Reset State 10-11
10.9.6.4 Transition from Execution to Debug State 10-12
10.9.6.5 Transition from Execution to STOP State 10-12
10.9.6.6 Transition from Execution to WAIT State 10-12
10.9.6.7 Transition from STOP/WAIT to Debug state 10-12
10.9.6.8 Transition from STOP to Execution state 10-13

10.9.6.9 Transition from WAIT to Execution state 10-13
10.9.6.10 Transition from Debug to Execution State 10-13

11 Internal Memory Subsystem

11.1 Memory Management Unit (MMU) 11-2
11.2 Instruction Channel (ICache and IFU) 11-4
11.3 Data Channel and Write Queue (DCache) 11-6
11.4 L2 Instruction Cache 11-8
11.4.1 CLASS 11-10
11.4.2 ICache Modules 11-11
11.4.3 Instruction Fetch Unit 11-11
11.4.4 L2 ICache Register Block 11-12
11.4.5 Global Attributes Support 11-12
11.4.6 Functional Mode of Operation 11-12
11.4.6.1 Enabling the L2 ICache 11-12
11.4.6.2 L2 ICache Global Invalidation Command 11-13
11.4.6.3 L2 ICache Global Lock Mode 11-13
11.4.6.4 L2 ICache Partial Lock 11-13
11.4.6.5 L2 ICache Line Replacement 11-14
11.4.6.5.1 PLRU Replacement 11-14
11.4.6.5.2 PLRU Bit Updates 11-15
11.4.6.5.3 PLRU Bits In Partial Lock 11-15
11.4.6.5.4 PLRU Inverse Mechanism 11-15
11.4.6.6 L2 ICache Sweep Operation 11-15
11.4.6.6.1 Invalidation Sweep 11-16
11.4.6.6.2 L2 ICache Global Sweep Operation 11-16
11.4.6.7 Fetch Operation 11-17
11.4.7 Debug Mode 11-17
11.4.7.1 Initialize/Read and Update State Registers 11-18
11.4.7.2 L2 ICache Array Access During Debug 11-21
11.5 M2 Memory 11-22
11.6 M3 Memory 11-23
11.7 Internal Boot ROM 11-24
11.8 Programming Model 11-25
11.8.1 L2 ICache Control Register 0 (L2IC_CR0) 11-26
11.8.2 L2 ICache Control Register 1 (L2IC_CR1) 11-26
11.8.3 L2 ICache Control Register 2 (L2IC_CR2) 11-28
11.8.4 L2 ICache LRM State Register (L2IC_LRM) 11-29
11.8.5 L2 ICache Tag State Register (L2IC_TAG) 11-31
11.8.6 L2 ICache Valid State Register (L2IC_VALID) 11-32
11.8.7 L2 ICache Debug Data Register (L2IC_DBG_DATA) 11-32

11.8.8 L2 ICache Debug Access Register (L2IC_DBG_ACS) 11-33
11.8.9 L2 ICache Cacheable Area Start Address (L2IC_CSA) 11-34
11.8.10 L2 ICache Cacheable Area End Address (L2IC_CEA) 11-35
11.8.11 L2 ICache Cacheable Area Enable (L2IC_CEN) 11-36
11.9 L2 ICache Programming Limitations 11-37

12 DDR SDRAM Memory Controller

12.1 Architecture 12-2
12.1.1 DDR SDRAM Interface Operation 12-4
12.1.2 DDR SDRAM Organization 12-6
12.1.3 DDR SDRAM Address Multiplexing 12-7
12.2 JEDEC Standard DDR SDRAM Interface Commands 12-10
12.3 DDR SDRAM Clocking and Interface Timing 12-12
12.3.1 Clock Distribution 12-15
12.3.2 DDR SDRAM Mode-Set Command Timing 12-16
12.3.3 DDR SDRAM Write Timing Adjustments 12-16
12.3.4 DDR SDRAM Refresh 12-17
12.3.4.1 DDR SDRAM Refresh Timing 12-18
12.3.4.2 DDR SDRAM Refresh and Power-Saving Modes 12-19
12.3.4.3 DDR Memory Controller Clock Stop Mode 12-21
12.3.5 DDR Data Beat Ordering 12-21
12.3.6 Page Mode and Logical Bank Retention 12-22
12.4 Error Checking and Correction 12-22
12.5 Error Management 12-24
12.6 Set-Up and Initialization 12-25
12.6.1 Programming Differences Between Memory Types 12-27
12.6.2 DDR SDRAM Initialization Sequence 12-29
12.7 Memory Controller Programming Model 12-30
12.7.1 Chip-Select Bounds (CSx_BNDS) 12-32
12.7.2 Chip-Select x Configuration Register (CSx_CONFIG) 12-33
12.7.3 DDR SDRAM Extended Refresh Recovery Register (TIMING_CFG_3) . . 12-34
12.7.4 DDR SDRAM Timing Configuration Register 0 (TIMING_CFG_0) 12-35
12.7.5 DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1) 12-38
12.7.6 DDR SDRAM Timing Configuration Register 2 (TIMING_CFG_2) 12-40
12.7.7 DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG) . . . 12-42
12.7.8 DDR SDRAM Control Configuration Register 2 (DDR_SDRAM_CFG_2) 12-44
12.7.9 DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE) . . . 12-46
12.7.10 DDR SDRAM Mode Configuration 2 Register (DDR_SDRAM_MODE_2)12-47
12.7.11 DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL) 12-47
12.7.12 DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL) 12-49

12.7.13 DDR SDRAM Data Initialization Register (DDR_DATA_INIT) 12-50
12.7.14 DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL) 12-50
12.7.15 DDR SDRAM Initialization Address Register (DDR_INIT_ADDRESS) . . 12-51
12.7.16 DDR SDRAM Initialization Address Enable Register (DDR_INIT_EN) . . 12-52
12.7.17 DDR SDRAM IP Block Revision 1 Register (DDR_IP_REV1)(. 12-53
12.7.18 DDR SDRAM IP Block Revision 2 Register (DDR_IP_REV2) 12-53
12.7.19 DDR SDRAM Memory Data Path Error Injection Mask High Register (DDR_ERR_INJECT_HI) 12-54
12.7.20 DDR SDRAM Memory Data Path Error Injection Mask Low Register (DDR_ERR_INJECT_LO) 12-55
12.7.21 DDR SDRAM Memory Data Path Error Injection Mask ECC Register (DDR_ERR_INJECT). 12-55
12.7.22 DDR SDRAM Memory Data Path Read Capture Data High Register (CAPTURE_DATA_HI). 12-56
12.7.23 DDR SDRAM Memory Data Path Read Capture Data Low Register (CAPTURE_DATA_LO) 12-57
12.7.24 DDR SDRAM Memory Data Path Read Capture ECC Register (CAPTURE_ECC) 12-57
12.7.25 DDR SDRAM Memory Error Detect Register (ERR_DETECT) 12-58
12.7.26 DDR SDRAM Memory Error Disable Register (ERR_DISABLE) 12-59
12.7.27 DDR SDRAM Memory Error Interrupt Enable Register (ERR_INT_EN) . 12-60
12.7.28 DDR SDRAM Memory Error Attributes Capture Register (CAPTURE_ATTRIBUTES) 12-61
12.7.29 DDR SDRAM Memory Error Address Capture Register (CAPTURE_ADDRESS) 12-62
12.7.30 DDR SDRAM Single-Bit ECC Memory Error Management Register (ERR_SBE). 12-62
12.7.31 DDR SDRAM DDR Status (DDR_STOP_STATUS) 12-63
12.7.32 DDDR SDRAM Power Control Register (DDR_PWR). 12-64
12.7.33 DDR SDRAM MDIC Output Enable Control Register (MDIC_OE_CONT) 12-65
12.7.34 DDR SDRAM Termination, OCD, and ODT Control Register (TERM_OCD_ODT_CONT) 12-66
12.7.35 DDR SDRAM Clock Ratio Control Register (CLK_RATIO_CONT). 12-67

13 Interrupt Handling

13.1 Global Interrupt Controller (GIC) 13-2
13.1.1 Virtual Interrupt Generation 13-2
13.1.2 Virtual $\overline{\text{NMI}}$ Generation 13-3
13.2 General Configuration Block. 13-3

13.2.1 Interrupt Groups 13-3
13.2.2 External Interrupts 13-4
13.2.3 Interrupt Handling 13-5
13.3 Interrupt Mapping 13-6
13.4 Restrictions 13-11
13.5 Programming Model 13-13
13.5.1 Global Interrupt Controller 13-13
13.5.1.1 Virtual Interrupt Generation Register (VIGR) 13-13
13.5.1.2 Virtual Interrupt Status Register (VISR) 13-14
13.5.1.3 Virtual NMI Generation Register (VNMIGR) 13-14
13.5.2 General Interrupt Configuration 13-15
13.5.2.1 General Interrupt Register 1 (GIR1) 13-15
13.5.2.2 General Interrupt Enable Register 1 for Cores 0–3 (GIER1_[0–3]) 13-16
13.5.2.3 General Interrupt Register 2 (GIR2) 13-17
13.5.2.4 General Interrupt Enable Register 2 for Cores 0–3 (GIER2_[0–3]) 13-19
13.5.2.5 General Interrupt Register 3 (GIR3) 13-21
13.5.2.6 General Interrupt Enable Register 3 for Cores 0–3 (GIER3_[0–3]) 13-23

14 Direct Memory Access (DMA) Controller

14.1 Operating Modes 14-2
14.2 Buffer Types 14-2
14.2.1 One-Dimensional Simple Buffer 14-3
14.2.2 One-Dimensional Cyclic Buffer 14-4
14.2.3 One-Dimensional Chained Buffer 14-5
14.2.4 One-Dimensional Incremental Buffer 14-6
14.2.5 One-Dimensional Complex Buffers With Dual Cyclic Buffers 14-7
14.2.6 Two-Dimensional Simple Buffer 14-8
14.2.7 Three-Dimensional Simple Buffer 14-10
14.2.8 Four-Dimensional Simple Buffer 14-11
14.2.9 Multi-Dimensional Chained Buffer 14-13
14.2.10 Two-Dimensional Cyclic Buffer 14-16
14.2.11 Three-Dimensional Cyclic Buffer 14-17
14.3 Arbitration Types 14-19
14.3.1 Round-Robin Arbitration 14-19
14.3.2 EDF Arbitration 14-20
14.3.2.1 Issuing Interrupts 14-21
14.3.2.2 Counter Control 14-21
14.3.2.3 Clock Source to the Counters 14-21
14.4 Interrupts 14-21
14.4.1 Maskable Interrupts 14-21
14.4.2 Nonmaskable Interrupts 14-22

14.5 Profiling 14-22
14.6 DMA Programming Model 14-23
14.6.1 DMA Buffer Descriptor Base Registers x (DMABDBRx) 14-24
14.6.2 DMA Controller Channel Configuration Registers x (DMACHCRx) 14-25
14.6.2.1 DMA Controller Global Configuration Register (DMAGCR) 14-27
14.6.3 DMA Channel Enable Register (DMACHER) 14-28
14.6.4 DMA Channel Disable Register (DMACHDR) 14-28
14.6.5 DMA Channel Freeze Register (DMACHFR) 14-29
14.6.6 DMA Channel Defrost Register (DMACHDFR) 14-29
14.6.7 DMA Time-To-Dead Line Registers x (DMAEDFTDLx) 14-30
14.6.8 DMA EDF Control Register (DMAEDFCTRL) 14-31
14.6.9 DMA EDF Mask Register (DMAEDFMR) 14-32
14.6.10 DMA EDF Mask Update Register (DMAEDFMUR) 14-32
14.6.11 DMA EDF Status Register (DMAEDFSTR) 14-34
14.6.12 DMA Mask Register (DMAMR) 14-34
14.6.13 DMA Mask Update Register (DMAMUR) 14-35
14.6.14 DMA Status Register (DMASTR) 14-36
14.6.15 DMA Error Register (DMAERR) 14-37
14.6.16 DMA Debug Event Status Register (DMADESR) 14-39
14.6.17 DMA Local Profiling Configuration Register (DMALPCR) 14-39
14.6.18 DMA Round-Robin Priority Group Update Register (DMARRPGUR) . . . 14-40
14.6.19 DMA Channel Active Status Register (DMACHASTR) 14-41
14.6.20 DMA Channel Freeze Status Register (DMACHFSTR) 14-41
14.6.21 DMA Channel Buffer Descriptors 14-42
14.6.21.1 Buffer Attributes (BD_ATTR) 14-44
14.6.21.2 Multi-Dimensional Buffer Attributes (BD_MD_ATTR) 14-47

15 PCI

15.1 Functional Description 15-2
15.1.1 PCI Operation Modes 15-2
15.1.2 Bus Commands 15-3
15.1.3 PCI Protocol Fundamentals 15-4
15.1.4 Addressing 15-4
15.1.5 Device Selection 15-5
15.1.6 Byte Enable Signals 15-5
15.1.7 Bus Driving and Turnaround 15-5
15.1.8 Bus Transactions 15-6
15.1.8.1 Read and Write Transactions 15-6
15.1.8.2 Transaction Termination 15-8
15.1.8.3 Other Bus Operations 15-10
15.1.8.3.1 Fast Back-to-Back Transactions 15-10

15.1.8.3.2 Dual Address Cycles 15-11
15.1.8.3.3 Data Streaming 15-11
15.1.8.3.4 Configuration Access 15-12
15.1.8.3.5 Special Cycle Command 15-12
15.1.8.3.6 Interrupt Acknowledge 15-13
15.1.8.4 Error Functions 15-13
15.1.8.4.1 Parity 15-13
15.1.8.4.2 Error Reporting 15-14
15.1.8.5 PCI Inbound Address Translation 15-15
15.1.8.6 PCI Outbound Address Translation 15-16
15.1.8.7 Transaction Ordering 15-17
15.1.9 Initialization Sequence 15-18
15.2 Programming Model 15-18
15.2.1 PCI Configuration Access Registers 15-20
15.2.1.1 PCI Configuration Address Register (CONFIG_ADDRESS) 15-20
15.2.1.2 PCI Configuration Data Register (CONFIG_DATA) 15-21
15.2.1.3 PCI Interrupt Acknowledge Register (PCI_INT_ACK) 15-22
15.2.2 PCI Configuration Space Registers 15-22
15.2.2.1 Vendor ID Configuration Register (VIDCR) 15-22
15.2.2.2 Device ID Configuration Register (DIDCR) 15-23
15.2.2.3 PCI Command Configuration Register (PCICCR) 15-23
15.2.2.4 PCI Status Configuration Register (PCISCR) 15-24
15.2.2.5 Revision ID Configuration Register (RIDCR) 15-25
15.2.2.6 Standard Programming Interface Configuration Register (SPICR) 15-26
15.2.2.7 Subclass Code Configuration Register (SCCR) 15-26
15.2.2.8 Base Class Code Configuration Register (BCCCR) 15-26
15.2.2.9 Cache Line Size Configuration Register (CLSCR) 15-27
15.2.2.10 Latency Timer Configuration Register (LTCR) 15-27
15.2.2.11 Header Type Configuration Register (HTCR) 15-28
15.2.2.12 BIST Control Configuration Register (BISTCCR) 15-28
15.2.2.13 PIMMR Base Address Configuration Register (PIMMRBACR) 15-28
15.2.2.14 GPL Base Address Register 0 (GPLBAR0) 15-29
15.2.2.15 GPL Base Address Registers 1–2 (GPLBAR[1–2]) 15-29
15.2.2.16 GPL Extended Base Address Registers 1–2 (GPLEXTBAR[1–2]) 15-30
15.2.2.17 Sub-System Vendor ID Configuration Register (SVIDCR) 15-31
15.2.3 Sub-System Device ID Configuration Register (SDIDCR) 15-31
15.2.3.1 Capabilities Pointer Configuration Register (CAPPTRCR) 15-32
15.2.3.2 Interrupt Line Configuration Register (INTLINCR) 15-32
15.2.3.3 Interrupt Pin Configuration Register (INTPINCR) 15-32
15.2.3.4 MIN GNT Configuration Register (MINGNTCR) 15-33
15.2.3.5 MAX LAT Configuration Register (MAXLATCR) 15-33

15.2.3.6 PCI Function Configuration Register (PCIFCR) 15-33
15.2.4 PCI Memory-Mapped Control and Status Registers 15-34
15.2.4.1 PCI Error Status Register (PCI_ESR) 15-34
15.2.4.2 PCI Error Capture Disable Register (PCI_ECDR) 15-35
15.2.4.3 PCI Error Enable Register (PCI_EER) 15-36
15.2.4.4 PCI Error Attributes Capture Register (PCI_EARCR) 15-37
15.2.4.5 PCI Error Address Capture Register (PCI_EACR) 15-39
15.2.4.6 PCI Error Extended Address Capture Register (PCI_EEACR) 15-39
15.2.4.7 PCI Error Data Low Capture Register (PCI_EDLCR) 15-40
15.2.4.8 PCI Inbound Translation Address Registers 0–2 (PITAR[0–2]) 15-40
15.2.4.9 PCI Inbound Base Address Registers 0–2 (PIBAR[0–2]) 15-41
15.2.4.10 PCI Inbound Extended Base Address Registers (PIEBAR[1–2]) 15-41
15.2.4.11 PCI Inbound Window Attribute Registers 0–2 (PIWAR[0–2]) 15-42
15.2.4.12 PCI Outbound Translation Address Registers 0–5 (PORAR[0–5]) 15-43
15.2.4.13 PCI Outbound Base Address Registers 0–5 (POBAR[0–5]) 15-44
15.2.4.14 PCI Outbound Comparison Mask Registers 0–5 (POCMR[0–5]) 15-44
15.2.4.15 Discard Timer Control Register (DTCR) 15-46

16 Serial RapidIO® Controller

16.1 Introduction 16-2
16.1.1 Features 16-2
16.1.2 Operating Modes 16-4
16.1.3 1x/4x LP-Serial Signals 16-5
16.2 RapidIO Interface Basics 16-5
16.2.1 RapidIO Transactions 16-6
16.2.2 RapidIO Packet Format 16-6
16.2.3 RapidIO Control Symbol Summary 16-8
16.2.4 Accessing Configuration Registers via RapidIO Packets 16-10
16.2.4.1 Inbound Maintenance Accesses 16-10
16.2.4.2 RapidIO Non-Maintenance Accesses Using LCSBA1CSR 16-10
16.2.4.3 RapidIO Maintenance Accesses 16-11
16.2.4.3.1 Guidelines 16-11
16.2.4.3.2 Outbound Maintenance Accesses 16-12
16.2.5 RapidIO ATMU Implementation 16-12
16.2.5.1 RapidIO Outbound ATMU 16-13
16.2.5.2 Outbound Windows 16-14
16.2.5.3 Window Size and Segmented Windows 16-15
16.2.5.3.1 Valid Hits to Multiple ATMU Windows 16-17
16.2.5.3.2 Window Boundary Crossing Errors 16-18
16.2.5.4 RapidIO Inbound ATMU 16-20
16.2.5.4.1 Hits to Multiple ATMU Windows 16-21

16.2.5.4.2 Window Boundary Crossing Errors 16-22
16.2.6 Generating Link-Request/Reset-Device 16-23
16.2.7 Outbound Drain Mode 16-23
16.2.8 Input Port Disable Mode 16-24
16.2.9 Software Assisted Error Recovery Register Support 16-25
16.2.10 Errors and Error Handling 16-25
16.2.10.1 RapidIO Error Description 16-26
16.2.10.2 Physical Layer RapidIO Errors 16-26
16.2.10.3 Logical Layer RapidIO Errors 16-30
16.3 RapidIO Message Unit 16-51
16.3.1 Features 16-51
16.3.2 Outbound Message Controller Operation 16-52
16.3.2.1 Direct Mode 16-52
16.3.2.2 Software Error Handling 16-55
16.3.2.3 Disabling and Enabling the Message Controller 16-56
16.3.2.4 Hardware Error Handling 16-56
16.3.2.5 Chaining Mode 16-60
16.3.2.5.1 Changing Descriptor Queues in Chaining Mode 16-63
16.3.2.5.2 Preventing Queue Overflow in Chaining Mode 16-63
16.3.2.5.3 Switching Between Direct and Chaining Modes 16-63
16.3.2.5.4 Chaining Mode Descriptor Format 16-63
16.3.2.5.5 Chaining Mode Controller Interrupts 16-64
16.3.2.6 Software Error Handling 16-66
16.3.2.7 Hardware Error Handling 16-67
16.3.2.8 Outbound Message Controller Arbitration 16-68
16.3.3 Inbound Message Controller Operation 16-68
16.3.3.1 Inbound Message Controller Initialization 16-69
16.3.3.2 Inbound Controller Operation 16-70
16.3.3.3 Message Steering 16-71
16.3.3.4 Retry Response Conditions 16-71
16.3.3.5 Inbound Message Controller Interrupts 16-71
16.3.3.6 Software Error Handling 16-72
16.3.3.7 Hardware Error Handling 16-73
16.3.3.8 Programming Errors 16-78
16.3.3.9 Disabling and Enabling the Inbound Message Controller 16-78
16.3.4 RapidIO Message Passing Logical Specification Register Bits 16-79
16.4 RapidIO Doorbell 16-79
16.4.1 Features 16-79
16.4.2 Doorbell Controller 16-80
16.4.3 Outbound Doorbell Controller 16-81
16.4.3.1 Interrupts 16-82

16.4.3.2 Software Error Handling 16-82
16.4.3.3 Hardware Error Handling 16-83
16.4.3.4 Programming Errors 16-86
16.4.4 Inbound Doorbell Controller 16-86
16.4.4.1 Doorbell Queue Entry Format 16-88
16.4.4.2 Retry Response Conditions 16-89
16.4.4.3 Doorbell Controller Interrupts 16-89
16.4.4.4 Transaction Errors 16-89
16.4.4.5 Software Error Handling 16-90
16.4.4.6 Hardware Error Handling 16-90
16.4.4.7 Programming Errors 16-93
16.4.4.8 Disabling and Enabling the Doorbell Controller 16-93
16.4.4.9 RapidIO Message Passing Logical Specification Registers 16-94
16.5 Port-Write Controller 16-94
16.5.1 Port-Write Controller Initialization 16-95
16.5.2 Port-Write Controller Operation 16-95
16.5.3 Port-Write Controller Interrupts 16-96
16.5.4 Discarding Port-Writes 16-96
16.5.5 Transaction Errors 16-96
16.5.6 Software Error Handling 16-96
16.5.7 Hardware Error Handling 16-97
16.5.8 Disabling and Enabling the Port-Write Controller 16-100
16.5.9 RapidIO Message Passing Logical Specification Registers 16-100
16.6 RapidIO Programming Model 16-101
16.6.1 Device Identity Capability Register (DIDCAR) 16-104
16.6.2 Device Information Capability Register (DICAR) 16-105
16.6.3 Assembly Identity Capability Register (AIDCAR) 16-105
16.6.4 Assembly Information Capability Register (AICAR) 16-106
16.6.5 Processing Element Features Capability Register (PEFCAR) 16-106
16.6.6 Source Operations Capability Register (SOCAR) 16-108
16.6.7 Destination Operations Capability Register (DOCAR) 16-109
16.6.8 Mailbox Command and Status Register (MCSR) 16-111
16.6.9 Port Write and Doorbell Command and Status Register (PWDCSR) 16-112
16.6.10 Processing Element Logical Layer Control Command and Status Register (PELLCCSR) 16-114
16.6.11 Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR) 16-115
16.6.12 Base Device ID Command and Status Register (BDIDCSR) 16-116
16.6.13 Host Base Device ID Lock Command and Status Register (HBDIDLCSR) 16-117
16.6.14 Component Tag Command and Status Register (CTCSR) 16-118

16.6.15 Port Maintenance Block Header 0 (PMBH0) 16-119
16.6.16 Port Link Time-Out Control Command and Status Register (PLTOCCSR) 16-120
16.6.17 Port Response Time-Out Control Command and Status Register (PRTOCCSR) 16-121
16.6.18 Port General Control Command and Status Register (PGCCSR). 16-122
16.6.19 Port 0 Link Maintenance Request Command and Status Register (P0LMREQCSR) 16-123
16.6.20 Port 0 Link Maintenance Response Command and Status Register (P0LMRESPCSR). 16-124
16.6.21 Port 0 Local ackID Command and Status Register (P0LASCR) 16-125
16.6.22 Port 0 Error and Status Command and Status Register (P0ESCSR). 16-126
16.6.23 Port 0 Control Command and Status Register (P0CCSR) 16-128
16.6.24 Error Reporting Block Header (ERBH) 16-130
16.6.25 Logical/Transport Layer Error Detect Command and Status Register (LTLEDCSR) 16-130
16.6.26 Logical/Transport Layer Error Enable Command and Status Register (LTLEECSR) 16-132
16.6.27 Logical/Transport Layer Address Capture Command and Status Register (LTLACCSR) 16-133
16.6.28 Logical/Transport Layer Device ID Capture Command and Status Register (LTLDIDCCSR) 16-134
16.6.29 Logical/Transport Layer Control Capture Command and Status Register (LTLCCCSR) 16-135
16.6.30 Port 0 Error Detect Command and Status Register (P0EDCSR) 16-136
16.6.31 Port 0 Error Rate Enable Command and Status Register (P0ERECSR). . . 16-137
16.6.32 Port 0 Error Capture Attributes Command and Status Register (P0ECACSR) 16-138
16.6.33 Port 0 Packet/Control Symbol Error Capture Command and Status Register (P0PCSECCSR) 16-140
16.6.34 Port 0 Packet Error Capture Command and Status Register 1 (P0PECCSR1). 16-141
16.6.35 Port 0 Packet Error Capture Command and Status Register 2 (P0PECCSR2). 16-142
16.6.36 Port 0 Packet Error Capture Command and Status Register 3 (P0PECCSR3). 16-143
16.6.37 Port 0 Error Rate Command and Status Register (P0ERCSR) 16-144
16.6.38 Port 0 Error Rate Threshold Command and Status Register (P0ERTCSR) 16-145
16.6.39 Logical Layer Configuration Register (LLCR). 16-146
16.6.40 Error/Port-Write Status Register (EPWISR). 16-147
16.6.41 Logical Retry Error Threshold Configuration Register (LRETCR) 16-148
16.6.42 Physical Retry Error Threshold Configuration Register (PRETCR) 16-149

16.6.43 Port 0 Alternate Device ID Command and Status Register (P0ADIDCSR) 16-150
16.6.44 Port 0 Accept-All Configuration Register (P0AACR) 16-151
16.6.45 Port 0 Logical Outbound Packet Time-to-Live Configuration Register (P0LOPTTLCR) 16-152
16.6.46 Port 0 Implementation Error Command and Status Register (P0IECSR). . 16-153
16.6.47 Port 0 Physical Configuration Register (P0PCR) 16-154
16.6.48 Port 0 Serial Link Command and Status Register (P0SLCSR). 16-155
16.6.49 Port 0 Serial Link Error Injection Configuration Register (P0SLEICR) . . 16-156
16.6.50 IP Block Revision Register 1 (IPBRR1). 16-157
16.6.51 IP Block Revision Register 2 (IPBRR2). 16-157
16.6.52 Port 0 RapidIO Outbound Window Translation Address Registers x (P0ROWTARx). 16-158
16.6.53 Port 0 RapidIO Outbound Window Translation Extended Address Registers x (P0ROWTEARx) 16-159
16.6.54 Port 0 RapidIO Outbound Window Attributes Registers x (P0ROWARx) 16-160
16.6.55 Port 0 RapidIO Outbound Window Base Address Registers x (P0ROWBARx) 16-162
16.6.56 Port 0 RapidIO Outbound Window Segment 1–3 Registers 1–8 (P0ROWSxRn) 16-163
16.6.57 Port 0 RapidIO Inbound Window Translation Address Registers x (P0RIWTARx) 16-164
16.6.58 Port 0 RapidIO Inbound Window Base Address Registers x (P0RIWBARx) 16-165
16.6.59 Port 0 RapidIO Inbound Window Attributes Registers x (P0RIWARx) . . 16-166
16.6.60 Outbound Message x Mode Registers (OMxMR) 16-167
16.6.61 Outbound Message x Status Registers (OMxSR). 16-169
16.6.62 Outbound Message x Descriptor Queue Dequeue Pointer Address Registers (OMxDQDPAR) 16-171
16.6.63 Outbound Message x Source Address Registers (OMxSAR). 16-172
16.6.64 Outbound Message x Destination Port Register (OMxDPR) 16-173
16.6.65 Outbound Message x Destination Attributes Register (OMxDATR). 16-174
16.6.66 Outbound Message x Double-Word Count Register (OMxDCR) 16-175
16.6.67 Outbound Message x Descriptor Queue Enqueue Pointer Address Registers (OMxDQEPAR) 16-176
16.6.68 Outbound Message x Retry Error Threshold Configuration Register (OMxRETCR). 16-177
16.6.69 Outbound Message x Multicast Group Registers (OMxMGR) 16-178
16.6.70 Outbound Message x Multicast List Registers (OMxMLR). 16-179
16.6.71 Inbound Message x Mode Registers (IMxMR). 16-180
16.6.72 Inbound Message x Status Registers (IMxSR) 16-182

16.6.73 Inbound Message x Frame Queue Dequeue Pointer Address Registers (IMxFQDPAR) 16-184
16.6.74 Inbound Message x Frame Queue Enqueue Pointer Address Registers (IMxFQEPAR) 16-185
16.6.75 Inbound Message x Maximum Interrupt Report Interval Registers (IMxMIRIR) 16-186
16.6.76 Outbound Doorbell Mode Register (ODMR) 16-187
16.6.77 Outbound Doorbell Status Register (ODSR) 16-188
16.6.78 Outbound Doorbell Destination Port Register (ODDPR) 16-189
16.6.79 Outbound Doorbell Destination Attributes Register (ODDATR) 16-190
16.6.80 Outbound Doorbell Retry Error Threshold Configuration Register (ODRETCR) 16-191
16.6.81 Inbound Doorbell Mode Registers (IDMR) 16-192
16.6.82 Inbound Doorbell Status Register (IDSR) 16-194
16.6.83 Inbound Doorbell Queue Dequeue Pointer Address Register (IDQDPAR) 16-195
16.6.84 Inbound Doorbell Queue Enqueue Pointer Address Registers (IDQEPAR) 16-196
16.6.85 Inbound Doorbell Maximum Interrupt Report Interval Register (IDMIRIR) 16-197
16.6.86 Inbound Port-Write Mode Register (IPWMR) 16-198
16.6.87 Inbound Port-Write Status Register (IPWSR) 16-199
16.6.88 Inbound Port-Write Queue Base Address Register (IPWQBAR) 16-200

17 RapidIO Interface Dedicated DMA Controller

17.1 Overview 17-2
17.1.1 Features 17-2
17.1.2 Modes of Operation 17-2
17.2 Functional Description 17-4
17.2.1 DMA Channel Operation 17-4
17.2.1.1 Basic DMA Mode Transfer 17-4
17.2.1.1.1 Basic Direct Mode 17-5
17.2.1.1.2 Basic Direct Single-Write Start Mode 17-5
17.2.1.1.3 Basic Chaining Mode 17-6
17.2.1.1.4 Basic Chaining Single-Write Start Mode 17-6
17.2.1.1.5 Extended DMA Mode Transfer 17-7
17.2.1.2 Channel Continue Mode for Cascading Transfer Chains 17-9
17.2.1.2.1 Basic Mode 17-9
17.2.1.2.2 Extended Mode 17-9
17.2.1.3 Channel Abort 17-10
17.2.1.4 Bandwidth Control 17-10
17.2.1.5 Channel State 17-10

17.2.1.6 Illustration of Stride Size and Stride Distance 17-11
17.2.2 DMA Transfer Interfaces 17-11
17.2.3 DMA Errors. 17-11
17.2.4 DMA Descriptors 17-12
17.2.5 Local Access ATMU Registers. 17-14
17.2.6 Limitations and Restrictions 17-15
17.3 Dedicated DMA Controller Programming Model 17-16
17.3.1 Local Access Window Base Address Registers 0–9 (LAWBAR[0–9]) 17-17
17.3.2 Local Access Window Attributes Registers 0–9 (LAWAR[0–9]) 17-18
17.3.3 Mode Registers 0–3 (MR[0–3]). 17-20
17.3.4 Status Registers (SRn). 17-23
17.3.5 Current Link Descriptor Extended Address Registers (ECLNDARn) 17-25
17.3.6 Current Link Descriptor Address Registers (CLNDARn) 17-26
17.3.7 Source Attributes Registers (SATRn). 17-27
17.3.8 Source Address Registers (SARn) 17-29
17.3.9 Destination Attributes Registers (DATRn). 17-30
17.3.10 Destination Address Registers (DARn) 17-32
17.3.11 Byte Count Registers (BCRn). 17-33
17.3.12 Extended Next Link Descriptor Address Registers (ENLNDARn) 17-34
17.3.13 Next Link Descriptor Address Registers (NLNDARn) 17-35
17.3.14 Extended Current List Descriptor Address Registers (ECLSDARn) 17-36
17.3.15 Current List Descriptor Address Registers (CLSDARn) 17-37
17.3.16 Extended Next List Descriptor Address Registers (ENLSDARn) 17-38
17.3.17 Next List Descriptor Address Registers (NLSDARn) 17-39
17.3.18 Source Stride Registers (SSRn). 17-40
17.3.19 Destination Stride Registers (DSRn). 17-41
17.3.20 DMA General Status Register (DGSR)) 17-42

18 QUICC Engine™ Subsystem

18.1 Overview 18-2
18.2 RISC Processors. 18-3
18.2.1 SC3400 Core Interface 18-3
18.2.2 Peripheral Interface 18-4
18.2.3 Parameter RAM. 18-4
18.2.4 Buffer Descriptors (BDs) 18-5
18.2.5 Multithreading. 18-6
18.2.6 Serial Numbers (SNUMs) 18-7
18.2.7 IRAM 18-8
18.3 Serial DMA Controller. 18-8
18.3.1 Data Paths 18-8
18.3.2 SDMA and Bus Error 18-9

18.3.2.1 Simple Recovery from Bus Error . . . 18-9
18.3.2.2 Selective Peripheral Recovery Procedure . . . 18-10
18.3.3 SDMA and Reset . . . 18-10
18.3.4 MBus Access . . . 18-10
18.3.5 SDMA Internal Resource . . . 18-11
18.4 Clocking . . . 18-11
18.4.1 Multiplexer Logic . . . 18-11
18.4.2 Baud-Rate Generators (BRGs) . . . 18-15
18.5 Interrupt Controller . . . 18-15
18.5.1 QUICC Engine Subsystem List of Interrupts to Core . . . 18-16
18.5.2 Interrupt Configuration . . . 18-18
18.6 UCCs . . . 18-19
18.7 Ethernet Controllers . . . 18-20
18.7.1 Operating Modes . . . 18-23
18.7.1.1 MII Mode . . . 18-23
18.7.1.2 RMII Mode . . . 18-23
18.7.1.3 SMII Mode . . . 18-23
18.7.1.4 RGMII Mode . . . 18-23
18.7.1.5 SGMII Mode . . . 18-24
18.7.2 Ethernet Physical Interfaces . . . 18-24
18.7.3 Media-Independent Interface (MII) . . . 18-27
18.7.4 Reduced Media-Independent Interface (RMII) Signals . . . 18-29
18.7.5 SMII Interface . . . 18-30
18.7.5.1 Reduced Gigabit Media-Independent Interface (RGMII) Signals . . . 18-31
18.7.5.2 Serial Gigabit Media-Independent Interface (SGMII) Signals . . . 18-33
18.7.5.2.1 SGMII Signals . . . 18-33
18.7.6 Controlling PHY Links (Management Interface) . . . 18-34
18.7.7 Ethernet Controller Initialization . . . 18-34
18.8 Asynchronous Transfer Mode (ATM) Controller . . . 18-35
18.8.1 Background . . . 18-35
18.8.2 ATM Controller Architecture . . . 18-37
18.8.3 UTOPIA Physical Interfaces . . . 18-37
18.8.4 UTOPIA Interface Master Mode . . . 18-38
18.8.5 UTOPIA Interface Slave Mode . . . 18-38
18.8.6 POS Interface Master Mode . . . 18-39
18.8.7 POS Interface Slave Mode . . . 18-40
18.9 Serial Peripheral Interface (SPI) . . . 18-40
18.9.1 SPI Operating Modes . . . 18-41
18.9.1.1 SPI as a Master Device . . . 18-41
18.9.1.2 SPI as a Slave Device . . . 18-43
18.9.2 SPI in Multi-Master Operation . . . 18-43

18.9.3 External Signal Configuration 18-45
18.9.4 SPI Transmission and Reception Process 18-45
18.10 Programming Model 18-46

19 TDM Interface

19.1 Typical Configurations 19-5
19.2 TDM Basics 19-6
19.2.1 Common Signals for the TDM Modules 19-8
19.2.2 Receiver and Transmitter Independent or Shared Operation 19-10
19.2.3 TDM Data Structures 19-13
19.2.4 Serial Interface 19-15
19.2.4.1 Sync Out Configuration 19-15
19.2.4.2 Sync In Configuration 19-16
19.2.4.3 Serial Interface Synchronization 19-18
19.2.4.4 Reverse Data Order 19-20
19.2.5 TDM Local Memory 19-22
19.2.6 Buffers Mapped on System Memory 19-23
19.2.6.1 Data Buffer Size and A/m-law Channels 19-23
19.2.6.2 Data Buffer Address 19-24
19.2.6.3 Threshold Pointers and Interrupts 19-26
19.2.6.4 Unified Buffer Mode 19-28
19.2.7 Adaptation Machine 19-29
19.3 TDM Power Saving 19-31
19.4 Channel Activation 19-31
19.5 Loopback Support 19-32
19.6 TDM Initialization 19-33
19.7 TDM Programming Model 19-34
19.7.1 Configuration Registers 19-36
19.7.1.1 TDMx General Interface Register (TDMxGIR) 19-36
19.7.1.2 TDMx Receive Interface Register (TDMxRIR) 19-44
19.7.1.3 TDMx Transmit Interface Register (TDMxTIR) 19-46
19.7.1.4 TDMx Receive Frame Parameters (TDMxRFP) 19-48
19.7.1.5 TDMx Transmit Frame Parameters (TDMxTFP) 19-51
19.7.1.6 TDMx Receive Data Buffer Size (TDMxRDBS) 19-53
19.7.1.7 TDMx Transmit Data Buffer Size (TDMxTDBS) 19-54
19.7.1.8 TDMx Receive Global Base Address 19-54
19.7.1.9 TDMx Transmit Global Base Address 19-55
19.7.1.10 TDMx Transmit Force Register (TDMxTFR) 19-55
19.7.1.11 TDMx Receive Force Register (TDMxRFR) 19-56
19.7.1.12 TDMx Parity Control Register (TDMxPCR) 19-57
19.7.2 Control Registers 19-57

19.7.2.1 TDMx Adaptation Control Register 19-57
19.7.2.2 TDMx Receive Control Register 19-58
19.7.2.3 TDMx Transmit Control Register (TDMxTCR) 19-59
19.7.2.4 TDMx Receive Data Buffer First Threshold (TDMxRDBFT) 19-59
19.7.2.5 TDMx Transmit Data Buffer First Threshold 19-60
19.7.2.6 TDMx Receive Data Buffer Second Threshold 19-61
19.7.2.7 TDMx Transmit Data Buffer Second Threshold 19-61
19.7.2.8 TDMx Receive Channel Parameter Register n 19-62
19.7.2.9 TDMx Transmit Channel Parameter Register n 19-63
19.7.2.10 TDMx Receive Interrupt Enable Register (TDMXRIER) 19-64
19.7.2.11 TDMx Transmit Interrupt Enable Register (TDMxTIER) 19-65
19.7.3 Status Registers 19-66
19.7.3.1 TDMx Adaptation Sync Distance Register (TDMxASDR) 19-66
19.7.3.2 TDMx Receive Data Buffers Displacement Register (TDMxRDBDR) . . 19-67
19.7.3.3 TDMx Transmit Data Buffer Displacement Register (TDMxTDBDR) . . 19-67
19.7.3.4 TDMx Receive Number of Buffers (TDMxRNB) 19-68
19.7.3.5 TDMx Transmitter Number of Buffers (TDMxTNB) 19-69
19.7.3.6 TDMx Receive Event Register (TDMxRER) 19-69
19.7.3.7 TDMx Transmit Event Register (TDMxTER) 19-70
19.7.3.8 TDMx Adaptation Status Register (TDMxASR) 19-71
19.7.3.9 TDMx Receive Status Register (TDMxRSR) 19-72
19.7.3.10 TDMx Transmit Status Register (TDMxTSR) 19-73
19.7.3.11 TDMx Parity Error Register (TDMxPER) 19-73

20 UART

20.1 Transmitter 20-6
20.1.1 Character Transmission 20-7
20.1.2 Break Characters 20-9
20.1.3 Idle Characters 20-10
20.1.4 Parity Bit Generation 20-10
20.2 Receiver 20-10
20.2.1 Character Reception 20-11
20.2.2 Data Sampling 20-12
20.2.3 Framing Error 20-17
20.2.4 Parity Error 20-18
20.2.5 Break Characters 20-18
20.2.6 Baud-Rate Tolerance 20-18
20.2.6.1 Slow Data Tolerance 20-19
20.2.6.2 Fast Data Tolerance 20-20
20.2.7 Receiver Wake-Up 20-20
20.2.7.1 Idle Input Line Wake-Up (WAKE = 0) 20-21

20.2.7.2 Address Mark Wake-Up (WAKE = 1) 20-21
20.3 Reset Initialization 20-21
20.4 Modes of Operation 20-22
20.4.1 Run Mode 20-22
20.4.2 Single-Wire Operation 20-22
20.4.3 Loop Operation 20-23
20.4.4 Stop Mode 20-23
20.4.5 Receiver Standby Mode 20-23
20.5 Interrupt Operation 20-24
20.6 UART Programming Model 20-24
20.6.1 SCI Baud-Rate Register (SCIBR) 20-25
20.6.2 SCI Control Register (SCICR) 20-26
20.6.3 SCI Status Register (SCISR) 20-29
20.6.4 SCI Data Register (SCIDR) 20-31
20.6.5 SCI Data Direction Register (SCIDDR) 20-32

21 Timers

21.1 Device-Level Timers 21-1
21.1.1 Features 21-2
21.1.2 Timer Module Architecture 21-2
21.1.3 Setting Up Counters for Cascaded Operation 21-3
21.1.3.1 Operation of the Cascaded Timer 21-4
21.1.3.2 Cascading Restrictions 21-4
21.1.4 Timer Operating Modes 21-5
21.1.4.1 One-Shot Mode 21-7
21.1.4.2 Pulse Output Mode 21-7
21.1.4.3 Fixed Frequency PWM Mode 21-7
21.1.4.4 Variable Frequency PWM Mode 21-8
21.1.5 Timer Compare Functionality 21-10
21.1.5.1 Compare Preload Registers 21-11
21.1.5.2 Capture Register Use 21-11
21.1.5.3 Broadcast from an Initiator Timer 21-12
21.1.6 Resets and Interrupts 21-12
21.1.6.1 Timer Compare Interrupts 21-12
21.1.6.2 Timer Overflow Interrupts 21-13
21.1.6.3 Timer Input Edge Interrupts 21-13
21.2 SC3400 DSP Core Subsystem Timers 21-13
21.3 Software Watchdog Timers 21-13
21.3.1 Features 21-14
21.3.2 Modes of Operation 21-14
21.3.3 Software WDT Servicing 21-15

21.4 Timers Programming Model 21-16
21.4.1 Device-Level Timers 21-16
21.4.1.1 Timer Channel Control Registers (TMRnCTLx) 21-17
21.4.1.2 Timer Channel Status and Control Registers (TMRnSCTLx) 21-19
21.4.1.3 Timer Channel Compare 1 Registers (TMRnCMP1x) 21-21
21.4.1.4 Timer Channel Compare 2 Registers (TMRnCMP2x) 21-21
21.4.1.5 Timer Channel Compare Load 1 Registers (TMRnCMPLD1x) 21-22
21.4.1.6 Timer Channel Compare Load 2 Registers (TMRnCMPLD2x) 21-22
21.4.1.7 Timer Channel Comparator Status and Control Registers (TMRnCOMSCx) 21-22
21.4.1.8 Timer Channel Capture Registers (TMRnCAPx) 21-23
21.4.1.9 Timer Channel Load Registers (TMRnLOADx) 21-24
21.4.1.10 Timer Channel Hold Registers (TMRnHOLDx) 21-24
21.4.1.11 Timer Channel Counter Registers (TMRnCNTRx) 21-24
21.4.2 SC3400 DSP Core Subsystem Timers 21-24
21.4.3 Software Watchdog Timers 21-25
21.4.3.1 System Watchdog Control Register 0–4 (SWCRR[0–4]) 21-25
21.4.3.2 System Watchdog Count Register 0–4 (SWCNR[0–4]) 21-26
21.4.3.3 System Watchdog Service Register 0–4 (SWSRR[0–4]) 21-27

22 GPIO

22.1 Features 22-1
22.2 GPIO Block Diagram 22-2
22.3 I/O Mode Functionality of GPIO 22-4
22.4 GPIO Connection Functions 22-5
22.5 GPIO Programming Model 22-7
22.5.1 Pin Open-Drain Register (PODR) 22-7
22.5.2 Pin Data Register (PDAT) 22-8
22.5.3 Pin Data Direction Register (PDIR) 22-9
22.5.4 Pin Assignment Register (PAR) 22-9
22.5.5 Pin Special Options Register (PSOR) 22-10

23 Hardware Semaphores

24 I^2C

24.1 Modes of Operation 24-2
24.2 I^2C Module Functional Blocks 24-3
24.2.1 Clock Control 24-3
24.2.2 Input Synchronization 24-3
24.2.3 Digital Input Filter 24-3
24.2.4 Transaction Monitoring 24-4

24.2.5 Arbitration Control 24-4
24.2.6 Transfer Control 24-4
24.2.7 In/Out Data Shift Register 24-5
24.2.8 Address Compare 24-5
24.3 Functional Description 24-6
24.3.1 START Condition 24-6
24.3.2 Target Address Transmission 24-7
24.3.3 Repeated START Condition 24-7
24.3.4 STOP Condition 24-8
24.3.5 Arbitration Procedure 24-8
24.3.6 Clock Synchronization 24-9
24.3.7 Handshaking 24-9
24.3.8 Clock Stretching 24-9
24.4 Initialization/Application Information 24-9
24.4.1 Initialization Sequence 24-10
24.4.2 Generation of START 24-10
24.4.3 Post-Transfer Software Response 24-10
24.4.4 Generation of STOP 24-11
24.4.5 Generation of Repeated START 24-11
24.4.6 Generation of SCL When SDA Low 24-11
24.4.7 Target Mode Interrupt Service Routine 24-12
24.4.8 Target Transmitter and Received Acknowledge 24-12
24.4.9 Loss of Arbitration and Forcing of Target Mode 24-12
24.4.10 Interrupt Service Routine Flowchart 24-13
24.5 Programming Model 24-15
24.5.1 I2C Address Register (I2CADR) 24-15
24.5.2 I2C Frequency Divider Register (I2CFDR) 24-16
24.5.3 I2C Control Register (I2CCR) 24-17
24.5.4 I2C Status Register (I2CSR) 24-18
24.5.5 I2C Data Register (I2CDR) 24-20
24.5.6 Digital Filter Sampling Rate Register (I2CDFSRR) 24-20

25 Debugging, Profiling, and Performance Monitoring

25.1 TAP, Boundary Scan, and OCE 25-1
25.1.1 Overview 25-2
25.1.2 TAP Controller 25-4
25.1.3 Instruction Decoding 25-5
25.1.4 Multi-Core JTAG and OCE Module Concept 25-9
25.1.5 Enabling the OCE Module 25-10
25.1.6 DEBUG_REQUEST and ENABLE_ONCE Commands 25-11
25.1.7 RD_STATUS Command 25-11

25.1.8 Reading/Writing OCE Registers Through JTAG 25-12
25.1.9 Signalling a Debug Request 25-13
25.1.10 EE_CTRL Modifications for the MSC8144E 25-13
25.1.11 ESEL_DM and EDCA_CTRL Register Programming 25-14
25.1.12 Real-Time Debug Request 25-14
25.1.13 Exiting Debug Mode 25-15
25.1.14 Debug Exception Request 25-16
25.1.15 Tracing in the MSC8144E 25-16
25.1.16 General JTAG Mode Restrictions 25-16
25.1.17 JTAG and OCE Module Programming Model 25-17
25.1.17.1 Identification Register 25-17
25.1.17.2 Boundary Scan Register (BSR) 25-17
25.1.17.3 Shift Registers 25-20
25.1.17.4 Bypass Register 25-20
25.1.17.5 Identification Register 25-20
25.2 Debug and Profiling 25-21
25.2.1 Features 25-21
25.2.2 Entering Debug Mode 25-22
25.2.3 Exiting Debug mode 25-22
25.2.4 SC3400 Debug and Profiling 25-23
25.2.5 L1 ICache and DCache Debug and Profiling 25-23
25.2.6 L2 ICache Debug and Profiling 25-23
25.2.6.1 CLASS Debug Profiling Units (CDPU) in the L2 ICache 25-24
25.2.6.2 L2 ICache Bank Profiling Events 25-25
25.2.7 DMA Controller Debug and Profiling 25-25
25.2.7.1 Debug Errors 25-25
25.2.7.2 Profiling Unit 25-26
25.2.8 CLASS Modules 25-26
25.2.8.1 Debug 25-26
25.2.8.2 CLASS Debug Profiling Unit (CDPU) 25-26
25.2.9 QUICC Engine Debug and Profiling 25-27
25.2.9.1 Trace Buffer 25-27
25.2.9.2 Loopback Modes 25-28
25.2.10 TDM Debug and Profiling 25-28
25.2.10.1 Debug 25-28
25.2.10.2 TDM Loopback Support 25-28
25.2.11 RapidIO Debug and Profiling 25-29
25.2.11.1 Debug Errors 25-29
25.2.11.2 Profiling Unit 25-29
25.2.12 PCI Debug 25-29
25.2.13 Software Watchdog (SWT) 25-29

25.2.14 Profiling Unit Programming Model 25-30
25.2.14.1 DPU Control Register (DP_CR) 25-31
25.2.14.2 DPU Status Register (DP_SR) 25-33
25.2.14.3 DPU Monitor Register (DP_MR) 25-34
25.2.14.4 DPU PID Detection Reference Value Register (DP_RPID) 25-36
25.2.14.5 DPU DID Detection Reference Value Register (DP_RDID) 25-36
25.2.14.6 DPU Counter Triad A Control Register (DP_TAC) 25-37
25.2.14.7 DPU Counter Triad B Control Register (DP_TBC) 25-40
25.2.14.8 DPU Counter A0 Control Register (DP_CA0C) 25-42
25.2.14.9 DPU Counter A0 Value Register (DP_CA0V) 25-45
25.2.14.10 DPU Counter A1 Control Register (DP_CA1C) 25-45
25.2.14.11 DPU Counter A1 Value Registers (DP_CA1V) 25-48
25.2.14.12 DPU Counter A2 Control Register (DP_CA2C) 25-48
25.2.14.13 DPU Counter A2 Value Registers (DP_CA2V) 25-51
25.2.14.14 DPU Counter B0 Control Register (DP_CB0C) 25-51
25.2.14.15 DPU Counter B0 Value Registers (DP_CB0V) 25-54
25.2.14.16 DPU Counter B1 Control Register (DP_CB1C) 25-54
25.2.14.17 DPU Counter B1 Value Registers (DP_CB1V) 25-57
25.2.14.18 DPU Counter B2 Control Register (DP_CB2C) 25-57
25.2.14.19 DPU Counter B2 Value Registers (DP_CB2V) 25-60
25.2.14.20 DPU Trace Control Register (DP_TC) 25-60
25.2.14.21 DPU VTB Start Address Register (DP_TSA) 25-64
25.2.14.22 DPU VTB End Address Register (DP_TEA) 25-65
25.2.14.23 DPU Trace Event Request Register (DP_TER) 25-66
25.2.14.24 DPU Trace Write Pointer Register (DP_TW) 25-67
25.2.14.25 DPU Trace Data Register (DP_TD) 25-68
25.3 Performance Monitor 25-68
25.3.1 Functional Description 25-70
25.3.1.1 Performance Monitor Interrupts 25-70
25.3.1.2 Event Counting 25-70
25.3.1.3 Threshold Events 25-71
25.3.1.4 Chaining 25-71
25.3.1.5 Triggering 25-72
25.3.1.6 Performance Monitor Events 25-72
25.3.1.7 Performance Monitor Examples 25-78
25.3.2 Performance Monitor Programming Model 25-79
25.3.2.1 Performance Monitor Global Control Register (PMGC) 25-80
25.3.2.2 Performance Monitor Local Control A0 Register 25-81
25.3.2.3 Performance Monitor Local Control An 25-82
25.3.2.4 Performance Monitor Local Control B0 25-83
25.3.2.5 Performance Monitor Local Control Bn 25-84

25.3.2.6 Performance Monitor Counter 0 (PMC0) 25-85
25.3.2.7 Performance Monitor Counter 1–8 (PMC[1–8]) 25-86

26 Security Engine (SEC)

26.1 Architecture Overview 26-1
26.1.1 Functional Diagram 26-2
26.1.2 Controller 26-3
26.1.2.1 Controller Operation 26-3
26.1.2.1.1 Channel-Controlled Access 26-3
26.1.2.1.2 Core Processor-Controlled Access 26-3
26.1.2.2 Descriptors and Link Tables 26-4
26.1.3 Channels 26-5
26.1.3.1 Channel Servicing 26-5
26.1.3.2 Channel Completion 26-6
26.1.3.3 Integrity Check Value (ICV) Generation and Checking 26-6
26.1.3.4 Encryption and Hashing 26-6
26.1.3.5 Snooping 26-6
26.1.4 Common EU Interface 26-7
26.1.5 Execution Units (EUs) 26-7
26.2 SEC Controller 26-8
26.2.1 Operation Definition and Data Control 26-8
26.2.1.1 Descriptors 26-9
26.2.1.1.1 Descriptor Structure 26-10
26.2.1.1.2 Descriptor Header 26-11
26.2.1.1.3 Descriptor Pointers 26-11
26.2.1.2 Link Tables 26-12
26.2.1.3 Using Descriptors and Link Tables 26-12
26.2.2 Assignment of EUs to Channels 26-14
26.2.2.1 Weighted Priority Arbitration 26-14
26.2.2.2 Round-Robin Arbitration 26-15
26.2.3 Bus Transfers 26-15
26.2.3.1 Arbitration for Use of the Controller and Buses 26-16
26.2.3.2 System Bus Master Read 26-17
26.2.3.3 System Bus Master Write 26-17
26.2.4 Controller Interrupts 26-18
26.2.5 Controller Registers 26-19
26.3 Channels 26-20
26.3.1 Channel Registers and Structures 26-21
26.3.2 Channel Interrupts 26-22
26.3.2.1 Channel Done Interrupt 26-22
26.3.2.2 Channel Error Interrupt 26-22

26.3.2.3 Channel Reset 26-22
26.4 Execution Units 26-23
26.4.1 Public Key Execution Unit (PKEU) 26-24
26.4.1.1 PKEU Mode Register 26-24
26.4.1.2 PKEU Key Size Register 26-24
26.4.1.3 PKEU AB Size Register 26-24
26.4.1.4 PKEU Data Size Register 26-25
26.4.1.5 PKEU Reset Control Register 26-25
26.4.1.6 PKEU Status Register 26-25
26.4.1.7 PKEU Interrupt Status Register 26-25
26.4.1.8 PKEU Interrupt Mask Register 26-26
26.4.1.9 PKEU End_of_Message Register 26-26
26.4.1.10 PKEU Parameter Memories 26-26
26.4.1.11 PKEU Routines 26-27
26.4.1.11.1 CLEARMEMORY: Clear Memory (0x01) 26-28
26.4.1.11.2 MOD_EXP: Prime field (Fp) Exponential mod N and Deconvert From Montgomery Format (0x02) 26-28
26.4.1.11.3 MOD_R2MODN: Prime Field (Fp) Compute Montgomery Converter Constant (0x03) 26-28
26.4.1.11.4 MOD_RRMODP: Prime Field (Fp) Compute Montgomery Converter Constant for Chinese Remainder Theorem (0x04) 26-29
26.4.1.11.5 EC_FP_AFF_PTMULT: Prime Field Elliptic Curve Scalar Point Multiply in Affine Coordinates (0x05) 26-29
26.4.1.11.6 EC_F2M_AFF_PTMULT: Polynomial Field Elliptic Curve Scalar Point Multiply in Affine Coordinates (0x06) 26-30
26.4.1.11.7 EC_FP_PROJ_PTMULT: Prime Field Elliptic Curve Scalar Point Multiply in Projective Coordinates (0x07) 26-30
26.4.1.11.8 EC_F2M_PROJ_PTMULT: Polynomial Field Elliptic Curve Scalar Point Multiply in Projective Coordinates (0x08) 26-31
26.4.1.11.9 EC_FP_ADD: Prime Field Elliptic Curve Point Add in Projective Coordinates (0x09) 26-32
26.4.1.11.10 EC_FP_DOUBLE: Prime Field Elliptic Curve Point Double in Projective Coordinates (0x0A) 26-32
26.4.1.11.11 EC_F2M_ADD: Polynomial Field Elliptic Curve Point Add in Projective Coordinates (0x0B) 26-33
26.4.1.11.12 EC_F2M_DOUBLE: Polynomial Field Elliptic Curve Point Double in Projective Coordinates (0x0C) 26-33
26.4.1.11.13 F2M_R2: Polynomial Field (F2m) Compute Montgomery Converter Constant (0x0D) 26-34
26.4.1.11.14 F2M_INV: Polynomial Field (F2m) Modular Inversion (0x0E) 26-34
26.4.1.11.15 MOD_INV: Prime Field (Fp) Modular Inversion (0x0F) 26-34

26.4.1.11.16 MOD_ADD: Prime Field (Fp) Modular Addition (0x10) 26-35
26.4.1.11.17 MOD_SUB: Prime Field (Fp) Modular Subtraction (0x20). 26-35
26.4.1.11.18 MOD_MULT1_MONT: Prime Field (Fp) Montgomery Modular Multiplication (0x30) 26-35
26.4.1.11.19 MOD_MULT2_DECONV: Prime Field (Fp) Montgomery Modular Multiplication and Deconvert From Montgomery Format (0x40) 26-36
26.4.1.11.21 F2M_MULT1_MONT: Polynomial Field (F2m) Montgomery Modular Multiplication (0x60) 26-36
26.4.1.11.22 F2M_MULT2_DECONV: Polynomial Field (F2m) Montgomery Modular Multiplication and Deconvert From Montgomery Format (0x70) 26-37
26.4.1.11.23 RSA_SSTEP: RSA Single Step Modular Exponentiation (0x80) 26-37
26.4.1.11.24 SPK_BUILD: Build PK Data Structure (0xFF) 26-38
26.4.2 Data Encryption Standard Execution Unit (DEU) 26-38
26.4.2.1 DEU Mode Register 26-38
26.4.2.2 DEU Key Size Register 26-38
26.4.2.3 DEU Data Size Register 26-38
26.4.2.4 DEU Reset Control Register 26-39
26.4.2.5 DEU Status Register. 26-39
26.4.2.6 DEU Interrupt Status Register 26-39
26.4.2.7 DEU Interrupt Mask Register. 26-39
26.4.2.8 DEU End_of_Message Register. 26-39
26.4.2.9 DEU IV Register 26-40
26.4.2.10 DEU Key Registers 26-40
26.4.2.11 DEU FIFOs. 26-40
26.4.3 Advanced Encryption Standard Execution Unit (AESU) 26-41
26.4.3.1 AESU Mode Register 26-41
26.4.3.2 AESU Key Size Register 26-42
26.4.3.3 AESU Data Size Register. 26-42
26.4.3.4 AESU Reset Control Register 26-42
26.4.3.5 AESU Status Register. 26-43
26.4.3.6 AESU Interrupt Status Register 26-43
26.4.3.7 AESU Interrupt Mask Register 26-43
26.4.3.8 AESU End_of_Message Register. 26-43
26.4.3.9 AESU Context Registers 26-44
26.4.3.9.1 Context for CBC Mode. 26-44
26.4.3.9.2 Context for Counter Mode 26-44
26.4.3.9.3 Context for SRT Mode 26-44
26.4.3.9.4 Context for CCM Mode 26-45
26.4.3.9.5 CCM Encryption Processing. 26-45
26.4.3.9.6 CCM Decryption Processing 26-46
26.4.3.10 AESU Key Registers 26-47

26.4.3.11 AESU FIFOs 26-48
26.4.4 Message Digest Execution Unit (MDEU) 26-48
26.4.4.1 MDEU Mode Register 26-49
26.4.4.2 MDEU Key Size Register 26-50
26.4.4.3 MDEU Data Size Register 26-50
26.4.4.4 MDEU Reset Control Register 26-51
26.4.4.5 MDEU Status Register 26-51
26.4.4.6 MDEU Interrupt Status Register 26-51
26.4.4.7 MDEU Interrupt Mask Register 26-52
26.4.4.8 MDEU ICV Size Register 26-52
26.4.4.9 MDEU End_of_Message Register 26-52
26.4.4.10 MDEU Context Registers 26-52
26.4.4.11 MDEU Key Registers 26-53
26.4.4.12 MDEU FIFOs 26-53
26.4.5 ARC Four Execution Unit (AFEU) 26-53
26.4.5.1 AFEU Mode Register 26-54
26.4.5.1.1 Core Processor-Provided Context via Prevent Permute 26-54
26.4.5.1.2 Dump Context 26-54
26.4.5.2 AFEU Key Size Register 26-54
26.4.5.3 AFEU Context/Data Size Register 26-55
26.4.5.4 AFEU Reset Control Register 26-55
26.4.5.5 AFEU Status Register 26-55
26.4.5.6 AFEU Interrupt Status Register 26-56
26.4.5.7 AFEU Interrupt Mask Register 26-56
26.4.5.8 AFEU End_of_Message Register 26-56
26.4.5.9 AFEU Context 26-57
26.4.5.9.1 AFEU Context Memory 26-57
26.4.5.10 AFEU Key Registers 26-57
26.4.6 Kasumi Execution Unit (KEU) 26-58
26.4.6.1 KEU Mode Register 26-59
26.4.6.2 KEU Key Size Register 26-59
26.4.6.3 KEU Data Size Register 26-59
26.4.6.4 KEU Reset Control Register 26-60
26.4.6.5 KEU Status Register 26-60
26.4.6.6 KEU Interrupt Status Register 26-60
26.4.6.7 KEU Interrupt Mask Register 26-61
26.4.6.8 KEU Data Out Register (F9 MAC) 26-61
26.4.6.9 KEU End_of_Message Register 26-61
26.4.6.10 KEU IV_1 Register 26-61
26.4.6.11 KEU ICV_In Register 26-61
26.4.6.12 KEU IV_2 Register (Fresh) 26-62

26.4.6.13 KEU Context Data Registers 26-62
26.4.6.14 KEU Key Data Registers_[1–2] (Confidentiality Key) 26-62
26.4.6.15 KEU Key Data Registers _[3–4] (Integrity Key) 26-62
26.4.6.16 KEU FIFOs 26-63
26.4.7 Random Number Generator (RNG) 26-63
26.4.7.1 RNG Mode Register 26-64
26.4.7.2 RNG Data Size Register 26-64
26.4.7.3 RNG Reset Control Register 26-64
26.4.7.4 RNG Status Register 26-64
26.4.7.5 RNG Interrupt Status Register 26-64
26.4.7.6 RNG Interrupt Mask Register 26-65
26.4.7.7 RNG End_of_Message Register 26-65
26.4.7.8 RNG FIFO 26-65
26.5 Programming Model 26-66
26.5.1 Descriptors and Link Tables 26-68
26.5.1.1 Descriptor Structure 26-69
26.5.1.2 Descriptor Header 26-72
26.5.2 Pointers 26-74
26.5.3 Link Tables 26-76
26.5.4 SEC Controller 26-77
26.5.4.1 Master Control Register (MCR) 26-77
26.5.4.2 Controller Identification Register (CIDR) 26-79
26.5.4.3 Controller IP Block Revision Register (CIPBRR) 26-79
26.5.4.4 EU Assignment Status (EUASR) 26-80
26.5.4.5 Controller Interrupt Enable Register (CIER) 26-81
26.5.4.6 Controller Interrupt Status Register (CISR) 26-84
26.5.4.7 Controller Interrupt Clear Register (CICR) 26-87
26.5.5 Channel Registers and Structures 26-89
26.5.5.1 Channel Configuration Registers for Channels 1–4 (CCR[1–4]) 26-89
26.5.5.2 Channel Pointer Status Registers (CPSR[1–4]) 26-92
26.5.5.3 Current Descriptor Pointer Register (CDPR) 26-98
26.5.5.4 Channel Fetch FIFO (CFF) 26-99
26.5.5.5 Channel Descriptor Buffer (DB) 26-100
26.5.6 PKEU Registers 26-101
26.5.6.1 PKEU Mode Register (PKEUMR) 26-101
26.5.6.2 PKEU Key Size Register (PKEUKSR) 26-103
26.5.6.3 PKEU AB Size Register (PKEUABSR) 26-104
26.5.6.4 PKEU Data Size Register (PKEUDSR) 26-105
26.5.6.5 PKEU Reset Control Register (PKEURCR) 26-106
26.5.6.6 PKEU Status Register (PKEUSR) 26-107
26.5.6.7 PKEU Interrupt Status Register (PKEUISR) 26-108

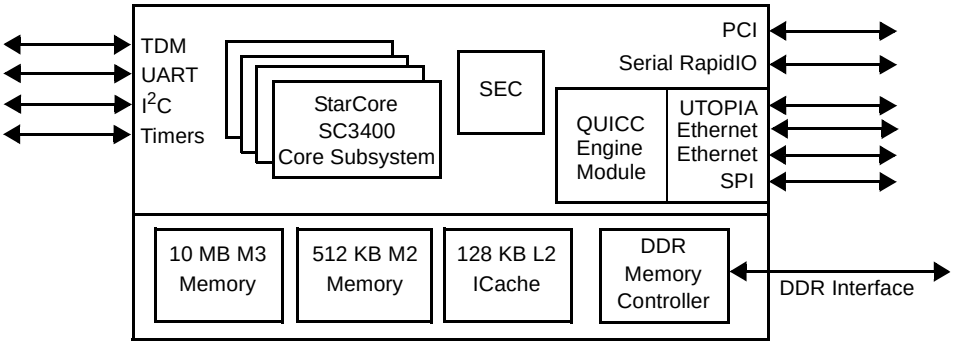
26.5.6.8 PKEU Interrupt Mask Register (PKEUIMR) 26-110
26.5.6.9 PKEU End_of_Message Register (PKEUEOMR) 26-111
26.5.6.10 PKEU Parameter Memories 26-111
26.5.6.10.1 PKEU Parameter Memory A 26-112
26.5.6.10.2 PKEU Parameter Memory B 26-112
26.5.6.10.3 PKEU Parameter Memory E 26-112
26.5.6.10.4 PKEU Parameter Memory N 26-112
26.5.7 DEU Registers 26-113
26.5.7.1 DEU Mode Register (DEUMR) 26-113
26.5.7.2 DEU Key Size Register (DEUKSR) 26-114
26.5.7.3 DEU Data Size Register (DEUDSR) 26-115
26.5.7.4 DEU Reset Control Register (DEURCR) 26-116
26.5.7.5 DEU Status Register (DEUSR) 26-117
26.5.7.6 DEU Interrupt Status Register (DEUISR) 26-118
26.5.7.7 DEU Interrupt Mask Register (DEUIMR) 26-120
26.5.7.8 DEU End_of_Message Register (DEUEOMR) 26-122
26.5.7.9 DEU IV Register (DEUIVR) 26-122
26.5.7.10 DEU Key Registers (DEUKR[1–3]) 26-123
26.5.7.11 DEU FIFOs 26-123
26.5.8 AESU Registers 26-124
26.5.8.1 AESU Mode Register (AESUMR) 26-124
26.5.8.2 AESU Key Size Register (AESUKSR) 26-126
26.5.8.3 AESU Data Size Register (AESUDSR) 26-127
26.5.8.4 AESU Reset Control Register (AESURCR) 26-128
26.5.8.5 AESU Status Register (AESUSR) 26-129
26.5.8.6 AESU Interrupt Status Register (AESUISR) 26-130
26.5.8.7 AESU Interrupt Mask Register (AESUIMR) 26-132
26.5.8.8 AESU End_of_Message Register (AESUEOMR) 26-134
26.5.8.9 AESU Context Registers (AESUCR[1–7]) 26-134
26.5.8.10 AESU Key Registers (AESUKR[1–3]) 26-135
26.5.8.11 AESU FIFOs 26-136
26.5.9 MDEU Registers 26-137
26.5.9.1 MDEU Mode Register (MDEUMR) 26-137
26.5.9.2 MDEU Key Size Register (MDEUKSR) 26-139
26.5.9.3 MDEU Data Size Register (MDEUDSR) 26-139
26.5.9.4 MDEU Reset Control Register (MDEURCR) 26-140
26.5.9.5 MDEU Status Register (MDEUSR) 26-141
26.5.9.6 MDEU Interrupt Status Register (MDEUISR) 26-142
26.5.9.7 MDEU Interrupt Mask Register (MDEUIMR) 26-144
26.5.9.8 MDEU ICV Size Register (MDEUICVSR) 26-146
26.5.9.9 MDEU End_of_Message Register (MDEUEOMR) 26-147

26.5.9.10 MDEU Context Registers (MDEUCR) 26-147
26.5.9.11 MDEU Key Registers (MDEUKR[1–8]) 26-149
26.5.9.12 MDEU Input FIFO 26-149
26.5.10 AFEU Registers. 26-150
26.5.10.1 AFEU Mode Register (AFEUMR). 26-150
26.5.10.2 AFEU Key Size Register (AFEUKSR) 26-151
26.5.10.3 AFEU Context/Data Size Register (AFEUCDSR). 26-152
26.5.10.4 AFEU Reset Control Register (AFEURCR) 26-153
26.5.10.5 AFEU Status Register (AFEUSR) 26-154
26.5.10.6 AFEU Interrupt Status Register (AFEUISR) 26-155
26.5.10.7 AFEU Interrupt Mask Register (AFEUIMR). 26-157
26.5.10.8 AFEU End_of_Message Register (AFEUEOMR). 26-159
26.5.10.9 AFEU Context Memory 26-159
26.5.10.10 AFEU Context Memory Pointer Register (AFEUCMPR) 26-160
26.5.10.11 AFEU Key Registers (AFEUKR[1–2]) 26-160
26.5.10.12 AFEU FIFOs 26-160
26.5.11 KEU Registers. 26-161
26.5.11.1 KEU Mode Register (KEUMR) 26-161
26.5.11.2 KEU Key Size Register (KEUKSR) 26-163
26.5.11.3 KEU Data Size Register (KEUDSR) 26-164
26.5.11.4 KEU Reset Control Register (KEURCR). 26-165
26.5.11.5 KEU Status Register (KEUSR) 26-166
26.5.11.6 KEU Interrupt Status Register (KEUISR) 26-167
26.5.11.7 KEU Interrupt Mask Register (KEUIMR) 26-169
26.5.11.8 KEU Data Out Register (KEUDOR) for F9 MAC. 26-171
26.5.11.9 KEU End_of_Message Register (KEUEOMR) 26-172
26.5.11.10 KEU IV1 Register (KEUIV1R) 26-173
26.5.11.11 KEU ICV_In Register (KEUICVIR) 26-174
26.5.11.12 KEU IV2 Register (KEUIV2R) 26-175
26.5.11.13 KEU Context 1–6 Registers (KEUCR[1–6]) 26-176
26.5.11.14 KEU Key Data Registers 1–2 (KEUKDR[1–2]) 26-177
26.5.11.15 KEU Key Data Registers 3–4 (KEUKDR[3–4]) 26-178
26.5.11.16 KEU Input FIFO/Output FIFO 26-178
26.5.12 RNG Registers. 26-179
26.5.12.1 RNG Mode Register (RNGMR). 26-179
26.5.12.2 RNG Data Size Register (RNGDSR). 26-180
26.5.12.3 RNG Reset Control Register (RNGRCR) 26-181
26.5.12.4 RNG Status Register (RNGSR) 26-182
26.5.12.5 RNG Interrupt Status Register (RNGISR) 26-183
26.5.12.6 RNG Interrupt Mask Register (RNGIMR). 26-184
26.5.12.7 RNG End_of_Message Register (RNGEOMR). 26-186

26.5.12.8 RNG Output FIFO 26-186

About This Book

The MSC8144E device is based on the StarCore SC3400 DSP core. It addresses the challenges of the media processing networking market. The benefits of the MSC8144E include not only a very high level of performance but also a product design that enables effective software development and integration. 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 MSC8144E device includes four DSP core subsystems, a large internal memory subsystem and DDR memory controller for external memory, and a variety of communication processors and interfaces.



Four DSP Core Subsystems

Each DSP core subsystem includes an SC3400 DSP core, a 16 KB 8-way level 1 ICache, a 32 KB 8-way level 1 DCache, 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 128 KB of shared L2 ICache, 512 KB of shared M2 memory for critical data and temporary data buffering, 10 MB of shared M3 memory that permit most applications to run using no external memory, a DDR-SDRAM controller to access up to 0.5 GB of DDR1 and DDR2 external memory, and a 32-channel direct memory access (DMA) controller optimized for DDR-SDRAM.

Security Engine (SEC)

The SEC subsystem includes an internal bus controller, four data channels, six execution units, and a shared random number generator to provide encryption/decryption support for communications security applications.

Communications Processors and Interfaces

Includes a PCI interface, serial RapidIO® interface, eight 512-channel (256 transmit and 256 receive) TDM interfaces, a UART interface, an I^2C interface, eight timer input/outputs, and a QUICC Engine module with one asynchronous transfer mode (UTOPIA) controller, two 10/100/1000Base-T Ethernet controllers, and an SPI. In addition, the global interrupt controller (GIC) consolidates all chip-maskable and non-maskable interrupts and routes them to $\overline{\text{NMI_OUT}}$, $\overline{\text{INT_OUT}}$, and to the cores. The hardware semaphores allow initiators to protect and reserve the system hardware resources.

Before Using This Manual—Important Note

This manual describes the structure and function of the MSC8144E 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 MSC8144E. 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:

- Features
- Architecture
- Signals
- Operation/operating modes
- Programming
- Programming Examples
- 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 MSC8144E device, the SC3400 cores are 16-bit DSP processors. The following table shows the SC3400 assembly language data types. For details, see the *StarCore SC3400 DSP Core Reference Manual*.

Name	SC3400
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 SC3400 C language data types recognized by the StarCore C compiler. For details, see the *StarCore SC100 C Compiler User's Manual (MNSC100CC/D)*.

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

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

Conventions for Registers

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

RSR Reset Status Register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RSTSRC			—					RIO	SW1	SW2	SW3	—			BSF
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—		SWSR	SWHR	—	JPO	JH	JS	—				SW	–	SRS	HRS
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Organization

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

- **Chapter 1,** *Overview*. Features, descriptive overview of main modules, configurations, and application examples.
- **Chapter 2,** *SC3400 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 MSC8144E devices.
- **Chapter 6,** *Boot Program*. Describes the bootloader program that loads and executes source code to initialize the MSC8144E 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 MSC8144E 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 MSC8144E modules; includes cross references to all registers discussed.
- **Chapter 10,** *MSC8144E SC3400 DSP Subsystem*. Describes the structure of the DSP core subsystem, which includes the SC3400 core, the instruction cache (ICache), the data cache (DCache), 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 ICache, M2 memory, and M3 memory including the configuration programming model.
- **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 MSC8144E interrupts, enabling interrupts to be handled by the SC3400 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,** *PCI*. Describes the how the PCI interface works and how to program it.
- **Chapter 16,** *Serial RapidIO® Controller*. Describes the how the serial RapidIO interface works and how to program it.
- **Chapter 17,** *RapidIO Interface Dedicated DMA Controller*. Describes the how the dedicated DMA controller supports the serial RapidIO interface and how to program it.

- **Chapter 18,** *QUICC Engine™ Subsystem*. Describes the QUICC Engine module, ATM controller, Ethernet controllers, and SPI controller. This is a general description. Detailed information is provided in the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM), available on the www.freescale.com website.
- ***Chapter 19,*** *TDM Interface.* Describes the eight TDM interfaces. Each can handle up to 256 channels. The interfaces support the serial bus rate and format for most standard TDM buses, including T1 and E1 highways, pulse-code modulation (PCM) highway, and the ISDN buses in both basic and primary rates.
- **Chapter 20,** *UART*. Describes the UART interface, which is a full-duplex serial port used to communicate with other devices.
- **Chapter 21,** *Timers*. Discusses the 32 identical 16-bit general-purpose timers residing in two timer modules (A and B) that each have their set of configuration registers.
- **Chapter 22,** *GPIO*. Discusses the thirty-two GPIO signals. Sixteen of the signals can be configured as external interrupt inputs. Each pin is multiplexed with other signals and can be configured as a general-purpose input, general-purpose output, or a dedicated peripheral pin.
- **Chapter 23,** *Hardware Semaphores*. Describes the function and programming of the hardware semaphores, which control resource sharing.
- **Chapter 24,** *I^2C*. Describes the I^2C interface. which allows the MSC8144E 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 SC3400 core and should be used with the supporting **IEEE**® Std. 1149.1™ documentation. The discussion covers the items that the standard requires to be defined and provides additional information specific to the MSC8144E implementation. Also includes debugging resources available in the SC3400 DSP core subsystem, including the OCE modules, and L2 ICache module.
- **Chapter 26,** *Security Engine (SEC)*. Describes the architecture, function, and register and memory structures used for security algorithm processing.

Other MSC8144E Documentation

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

- *MSC8144E Technical Data Sheet (*MSC8144E*)*. Details the signals, AC/DC characteristics, clock signal characteristics, package and pinout, and electrical design considerations of the MSC8144E device.
- *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM). Provides detailed register and programming information for the QUICC Engine technology. The overview and the specific chapters identify the specific blocks incorporated into the various Freescale devices.
- *Application Notes*. Cover various programming topics related to the StarCore DSP core and the MSC8144E device.

Further Reading

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

- *SC3400 DSP Core Reference Manual*. Covers the SC3400 core architecture, control registers, clock registers, program control, on-chip emulator (OCE), and instruction set.
- *MSC8144 SC3400 DSP Core Subsystem Reference Manual*. Covers the SC3400 DSP core subsystem which includes an SC3400 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.

Overview 1

The MSC8144E multicore DSP targets high-bandwidth highly computational applications and is optimized for packet telephony, wireless, and video transcoding and radio network controller (RNC) applications. The MSC8144E device is a highly integrated DSP processor that contains four StarCore SC3400 DSP subsystems, 512 KB of M2 shared memory, 10 MB of M3 shared memory, L1 instruction and data caches optimized for packet telephony, 128 KB of shared L2 ICache, a DDR memory controller, a serial RapidIO interface, two 10/100/1000Base-T Ethernet controllers, an ATM Controller supporting various ATM adaptation layers, a serial peripheral interface (SPI), eight 512-channel (256 receive and 256 transmit) time-division multiplexing (TDM) interfaces, a 16-channel DMA controller, 32-bit PCI interface that runs at 66/33MHz, a UART interface, an I^2C interface, and a security engine (SEC). The SC3400 core is a high performance DSP core in the StarCore family and is binary compatible with the SC140 core used in the MSC81xx DSP family and the SC1400 core used in the MSC711x DSP family. Each SC3400 DSP core has four ALUs and performs at 3200/4000 16 × 16-bit million multiply accumulates per second (MMACS) at 800 MHz/1 GHz yielding a maximum total performance of 12800/16000 16 × 16-bit MMACS per device. For optimized video applications, 8 × 8-bit multiply accumulate instructions can be used, delivering a peak performance of 25600/32000 MMACS per device. The MSC8144E is carefully optimized for minimal cost, power, and area per channel. Each SC3400 core connects to the following:

- 16 KB 8-way level 1 instruction cache (ICache)
- 32 KB 8-way level 1 data cache (DCache)

The MSC8144E has two types of interfaces: TDM and packet (Ethernet, UTOPIA, and RapidIO). In TDM-to-packet applications, for example, data received from the TDM interface is stored in the MSC8144E memory, processed by the SC3400 cores, and transmitted on one of the packet interfaces. In the other direction, packets are received, stored in the MSC8144E memory, processed by the SC3400 cores, and transmitted via the TDM interface.

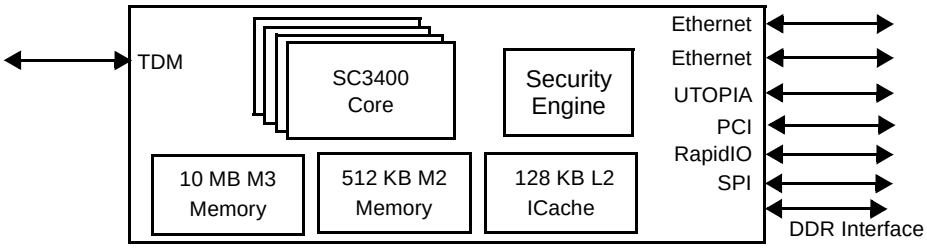


Figure 1-1. MSC8144E General Diagram

1.1 Features

Table 1-1 lists the features of the Freescale MSC8144E device.

Table 1-1. MSC8144E Features

Feature	Description
Performance	Offered with core frequencies of 800 MHz or 1 GHz, supports: • 16 x 16-bit multiply accumulate instructions. Up to 12800/16000 MMACS at 800 MHz/1 GHz within four SC3400 cores. • 8 x 8-bit multiply accumulate instructions for video applications. Up to 25600/32000 MMACS at 800 MHz/1 GHz within four SC3400 cores. The 16 ALUs deliver a performance equivalent to a single core running at 3.2/4 GHz. A multiply-accumulate operation includes a multiply-add instruction with the associated data move and pointer update.
StarCore DSP Subsystem	Each high performance DSP subsystem includes: • StarCore SC3400 core. • L1 ICache: – 16 KB – 8 way with 8 lines per way. – Multitasking support. – Real-time support through locking flexible boundaries. – Prefetch capability. – Software coherency support. • L1 DCache: – 32 KB. – 8 way with 16 lines per way. – Can service two data accesses in parallel (Xa, Xb). – Multitasking 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 KB write-back buffer (WBB). – Six 64-bit entry write-through buffer (WTB). – Prefetch capability. • Memory management unit (MMU): – Virtual-to-physical address translation. – Task protection. – Multitasking. • Embedded programmable interrupt controller (EPIC). – Up to 256 interrupts. – 32 priority levels. • Two general-purpose 32-bit timers. • Debug and profiling support: – On-chip emulator (OCE) for core-related debug and profiling support. – Debug and profiling unit (DPU) for platform level debug and profiling support. – Debug state, single stepping, and command insertion from the host debugger. – Test Access Port (TAP) designed to comply with **IEEE** Std. 1149.1™. – Breakpoints on PC, data address, and data bus values. – More than 40 event counting options in 6 parallel counters. – Cache debug mode enables cache state observation (cache array, tags, valid, and dirty bits) and DCache array content modification. – Real-time tracing of PC, task ID, and profiling information to the main memory with the virtual trace write buffer. • Low-power design. • Low-power modes of operation: – Wait processing state in which the core clocks and caches are gated but peripherals continue to operate. – Stop processing state for full clock gating.

Table 1-1. MSC8144E Features (Continued)

Feature	Description
StarCore SC3400 DSP Core	Each high-performance core is binary compatible with the SC140 core used in the MSC81xx DSP family and the SC1400 core used in the MSC711x DSP family and delivers up to 3200/4000 16-bit MMACS using an internal 800 MHz/1 GHz clock at 1 V. Each core includes: • Data arithmetic and logic unit (DALU) containing 4 ALUs. • Address generation unit (AGU) containing two address arithmetic units. • Up to six instructions execute in a single clock cycle. • Variable-length execution set (VLES) that can be optimized for code density and performance. • 16 data registers, 40 bits each. • 27 address registers, 32 bits each. • Hardware support for fractional and integer data types. • Four hardware loops with zero overhead. • Very rich 16-bit wide orthogonal instruction set. • Application-specific instructions for Viterbi and multimedia processing. • Special single instruction, multiple data (SIMD) instructions working on 2-word or 4-byte operands packed in a register. Can perform 2 to 4 operations per instruction (8 to 16 operations per VLES). The SIMD instruction supports 2 × 8-bit multiply and 20-bit accumulate operation. • Dynamic interlocking for friendlier programming and more efficient compiler support. • User and supervisor privilege levels supporting a protected software model. • Precise memory access exceptions enables good RTOS support and soft error corrections. • Branch target buffer (BTB) accelerates change-of-flow operations.
Chip-Level Arbitration and Switching System (CLASS)	• A full fabric that arbitrates between the DSP cores and other CLASS initiators to the shared M2 memory, M3 memory, PCI, DDR SDRAM controller, 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 10.96 MB internal memory space total includes: • 16 KB ICache per core. • 32 KB DCache per core. • 128 KB L2 shared ICache. • 512 KB M2 low-latency memory for critical data and temporary data buffering. Accessible from all DSP subsystems and all CLASS initiators via four interleaved ports. The memory is volatile after reset. • 10 MB 128-bit wide M3 memory accessed at up to 400 MHz. Accessible from all DSP subsystems and all CLASS initiators. Most applications run with no external memory. • 96 KB of boot ROM accessible from the cores.
M2 Memory	• 512 KB of low latency SRAM that is volatile after reset. • Runs at up to 400 MHz. • Four address-interleaved banks. • 128-bit wide port per bank. • Up to four simultaneously accesses. • Burstable access support.
M3 Memory	• 128-bit wide port. • Runs at up to 400 MHz. • 10 MB of memory. • A variety of applications can run without the need for external memory. • Hidden refresh with low probability of conflict with core accesses. • Burstable accesses.
L2 Cache	• L2 shared ICache. • 128 KB of memory organized in two interleaved banks of 64 KB each. • Banks can have simultaneous access. • 8-way set associative. • Pass-through port for L2 non-cacheable instructions. • Optimized to accelerate core code execution from M3 memory and DDR memory.

Table 1-1. MSC8144E Features (Continued)

Feature	Description
Clocks	• Three input clocks: – Shared input clock. – Global input clock (PCI PLL). – Differential input clock (SRIO PLL). • Four PLLs: – System PLL. – Core PLL. – Global PLL. – SRIO PLL. • Clock ratios selected during reset via reset configuration pins. • Clock modes user-configurable after reset.
Security Engine	• Four crypto-channels, each supporting multi-command descriptor chains • Dynamic assignment of crypto-execution units via an integrated controller • Buffer size of 256 bytes for each execution unit, with flow control for large data sizes • PKEU—public key execution unit – RSA and Diffie-Hellman; programmable field size up to 2048 bits – Elliptic curve cryptography with F2m and F(p) modes and programmable field size up to 511 bits • DEU—Data Encryption Standard (DES) execution unit – DES, 3DES – Two key (K1, K2, K1) or three key (K1, K2, K3) – ECB and CBC modes for both DES and 3DES • AESU—Advanced Encryption Standard (AES) unit – Implements the Rijndael symmetric key cipher – ECB, CBC, CTR, and CCM modes – 128-, 192-, and 256-bit key lengths • AFEU—ARC four execution unit – Implements a stream cipher compatible with the RC4 algorithm – 40- to 128-bit programmable key • MDEU—message digest execution unit – SHA with 160- or 256-bit message digest – MD5 with 128-bit message digest – HMAC with either algorithm • KEU—Kasumi execution unit – Implements F8 algorithm for encryption and F9 algorithm for integrity checking – Also supports A5/3 and GEA-3 algorithms • RNG—random number generator • XOR engine accelerates parity checking in RAID storage applications • Performs computationally intensive security functions including the following: – Key generation and exchange – Authentication – Bulk encryption. • 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) – **IEEE** 802.11i™ security standard.

Table 1-1. MSC8144E Features (Continued)

Feature	Description
DDR Controller	• Up to 200 MHz clock rate (400 MHz data rate). • 16/32-bit DDR SDRAM data bus. • Supported memory includes 64 Mb to 4 Gb DDR1 and DDR2 devices with x8/x16 data ports (no direct x4 support). • DDR SDRAM chip configurations up to 0.5 GB, including: – Up to two physical banks (chip selects), each independently addressable. – 32/40-bit SDRAM data bus and 16/24-bit SDRAM data bus for DDR1 and DDR2. • Up to two memory chip selects with each chip select independently addressable. • Four or eight sub-banks per chip select. • DDR SDRAM timing parameters are fully programmable. • Data mask (DM) signal and read-modify-write (RMW) for writing less than 4 bytes. • Open page management (up to four open pages). • Double-bit error detection and single-bit error correction (ECC). • Sleep power management. • DDR controller clock independent of the DSP clock to maximize memory performance while decoupling connection to the CLASS128 clock.
DMA Controller	• 16 bidirectional channels, providing up to 16 memory-to-memory channels. • Buffer descriptor (BD) programing model. • Up to 1024 BDs per channel direction provide a total of 32 K BDs. BDs can reside in M2, M3, or DDR memory. • 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. – 1D or 2-4D complex buffers, a combination of buffer types. • High bandwidth. • Optimized for DDR memory.
TDM	• Backward-compatible with the MSC8102/MSC8122/MSC8126 TDM interface. • All the eight TDM modules together support up to 2K time-slots for receive and 2K time-slots for transmit. • Up to eight independent TDM modules: – *Independent receive and transmit mode*. Independent transmitter and receiver. Transmitter input clock, output data, and frame sync can be configured as either input or output. Up to 256 transmit channels. Receiver input clock, data, and frame sync. Up to 256 receive channels. – *Shared sync and clock mode*. Two receive and two transmit links share the same clock and frame sync. The sync can be configured as either input or output. Up to 128 transmit channels and 128 receive channels. – *Shared data link*. Up to four full-duplex data links can operate as either transmit or receive. All links have the same clock and frame sync. Each link supports up to 128 channels. • Word size of 2, 4, 8, or 16-bit. All the channels share the same size. • Hardware A-law/μ-law conversion. • Up to 500 Mbps data rate for all TDM modules combined (62.5 Mbps for each of 8 TDM modules). • Up to 16 MB per channel buffer (granularity 8 bytes), where A/μ law buffer size has double size (16-byte granularity). • Separate or shared interrupts for receive and transmit with two programmable receive and two programmable transmit thresholds for double buffering. • Each channel can be programmed as active or inactive. • Support either 0.5 ms (4 frames) or 1 ms (8 frames) latency. • Glueless interface to E1/T1 framers. • Capable of interfacing with H-MVIP/H.110 device, TSI, and codecs such as AC-97.

Table 1-1. MSC8144E Features (Continued)

Feature	Description
QUICC Engine Subsystem	The QUICC Engine subsystem handles the Ethernet and ATM interfaces, thus offloading the cores from handling those tasks. It includes the following: • Dual-RISC engine, with 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, and ability to handle serial protocols and virtual DMA • Multi-initiator 48-KB multi-port RAM • 48-KB instruction RAM (IRAM) • Serial DMA channel • Two full-duplex programmable communications controllers supporting two gigabit Ethernet controllers (10/100/1000 Mbps operation) • One full-duplex programmable communications controller with a ATM controller for a 50 MHz, 16-bit UTOPIA interface supporting AAL0, AAL2, and AAL5 operation • Interrupt controller • Multiplexer and timers logic • Baud-rate generators
Two Ethernet Controllers 10/100/1000	• Five Ethernet physical interfaces: – 10/100 Mbps MII (one controller only). – 10/100 Mbps RMII (consortium standard). – 10/100 Mbps SMII. – Designed to comply with the SGMII protocol using a 4-pin SerDes interface at 1000 Mbps data rate only. – 10/100/1000 Mbps RGMII (full duplex only). • MAC-to-MAC connection in all modes. • Half- and full-duplex operations in 10/100 Mbps mode. • Half-duplex back-pressure (10/100 Mbps only). • Full-duplex operations in 1000 Mbps mode. • 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.

Table 1-1. MSC8144E Features (Continued)

Feature	Description
ATM Controller	Universal test and operations PHY interface for ATM (UTOPIA) controller: • UTOPIA level II supports 8/16 bits 25/50 MHz. • UTOPIA target mode. • Cell-level handshake. • Multiple-PHY polling mode. • ATM adaptation layers support AAL0, AAL2, and AAL5 protocols in hardware. • Full duplex segmentation and reassembly at up to 622 Mbps for AAL5. • Full duplex segmentation and reassembly at up to 155 Mbps for AAL2. • Up to 255 active VCs internally and up to 64 K VCs using external memory. • Unassign cells screening option. • Internal rate transmit mode. • User-defined cells up to 65 bytes. • Separate TxBD and RxBD tables for each virtual channel (VC). • Special mode of global free buffer pools for dynamic and efficient memory allocation with early packet discard (EPD) support. • Interrupt report per channel using four priority interrupt queues. • Compliant with ATMF UNI 4.0 and ITU specification. • ATM pace control (APC) unit. • Receive address look-up mechanism. • Operations and maintenance (OAM) cell. • ATM layer statistic gathering on a per PHY basis.
Serial Peripheral Interface (SPI)	• Four-signal interface (SPIMOSI, SPIMISO, SPICLK and $\overline{\text{SPISEL}}$) • 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 clk /8 in master mode and QUICC Engine clk /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 • Programmable clock gap between two characters in master mode • Controlled by the QUICC Engine RISC according to user configuration.
PCI	• Designed to be compliant with the PCI specification revision 2.2 per the voltage specifications in the *MSC8144E Technical Data* sheet. • 33 MHz and 66 MHz. • 32-bit PCI interface. • PCI 3.3-V compatible. • Accesses to all PCI address spaces. • PCI-to-system and system-to-PCI streaming.
SRIO Subsystem	• One serial RapidIO port 1x/4x. • RapidIO messaging unit.
Serial RapidIO Port (SRIO)	1x/4x serial RapidIO endpoint complies with the following parts of Specification 1.2 of the RapidIO trade association interconnect specification: • Part I (input/output logical specifications). • Part II (message passing logical specification). • Part III (common transport specification). • Part VI (physical layer 1x/4x LP-serial specification). • Part VIII (error management extension specification). The serial RapidIO port supports read, write, messages, doorbells, and maintenance accesses: • Small and large transport information field only. • All priorities flow.

Table 1-1. MSC8144E Features (Continued)

Feature	Description
RapidIO Messaging Unit	• Two outbound message queues. • Two inbound message queues. • One outbound doorbell queue. • One inbound doorbell queue. • One inbound port-write queue.
Global Interrupt Controller (GIC)	Consolidates all chip maskable interrupt and non-maskable interrupt sources and routes them to INT_OUT, 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 SC3400 core. • Four TMR modules, each with the following features: – Four 16-bit timers. – Cascadable timers. – Count up/down. – Programmable count modulo. – Count once or repeatedly. – Counters are preload able. – Compare registers can be preloaded. – Counters can share available inputs. – Separate prescaler for each counter. – Each counter has capture and compare capability. – Can use one of the following clock sources: CLASS64 clock, TDM clock input, or external clock input. • Five software watchdog timer (SWT) modules
Hardware Semaphores	Eight programmable hardware semaphores, locked by simple write access without need for read-modify-write operation by the DSP cores.
Virtual interrupts	• Generation of 18 virtual interrupts by a simple write access. • Generation of four virtual NMIs by a simple write access.
I^2C	• Two-wire interface. • Multi-initiator 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.
GPIO	• 32 GPIO ports. • Each GPIO port can either serve the on-device peripherals or act as a programmable I/O port. • 16 ports can be configured as external interrupt inputs. • All ports are bidirectional. • All ports are set as GPIO inputs at system reset. • All ports values can be read while the port is connected to an internal peripheral. • All ports have open-drain output capability.

Table 1-1. MSC8144E Features (Continued)

Feature	Description
Boot Options	Boot interfaces: • Serial RapidIO. • PCI. • I^2C. • Ethernet.
JTAG	Test Access Port (TAP) designed to comply with **IEEE** Std. 1149.1.
Reduced Power Dissipation	• Very low power CMOS design. • All modules but the DSP subsystem run at 50 percent of the DSP core frequency or less. • Low-power standby modes. • Optimized power management circuitry (instruction-dependent, peripheral-dependent, and mode-dependent).
Technology	CMOS 90 nm.
Package	• Flip Chip-Plastic Ball Grid Array (FC-PBGA), 1 mm pitch, 29 mm × 29 mm.
Voltage	• Core nominal voltage: 1 V. • M3 power: 1.2 V. • I/O power: 1.8 V/2.5 V/3.3 V.

1.2 Block Diagram

Figure 1-2 shows the MSC8144E block diagram. Arrows are in the direction from initiator to target.

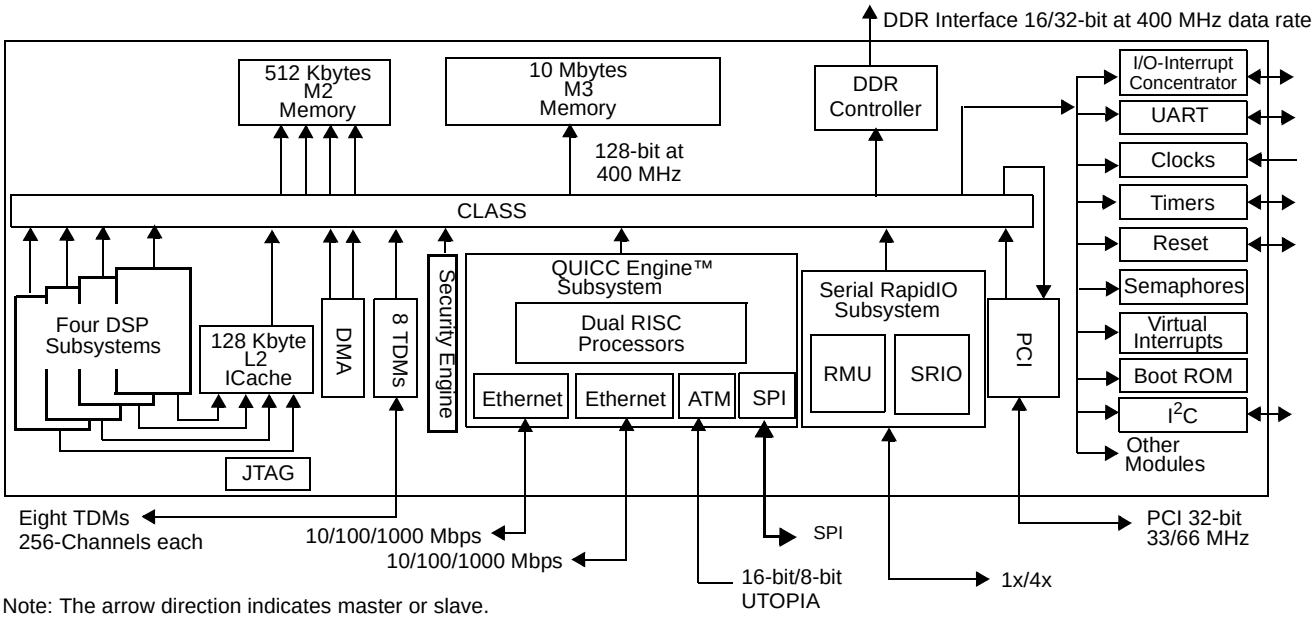


Figure 1-2. MSC8144E Block Diagram

1.3 Architecture

The MSC8144E architecture is carefully optimized to achieve the maximum channel density for a given device area, power, and cost. Also, the MSC8144E 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 SC3400 DSP Subsystem

Figure 1-3 shows the block diagram of the StarCore SC3400 DSP subsystem, which contains the SC3400 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 module, and a trace write buffer. The SC3400 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.

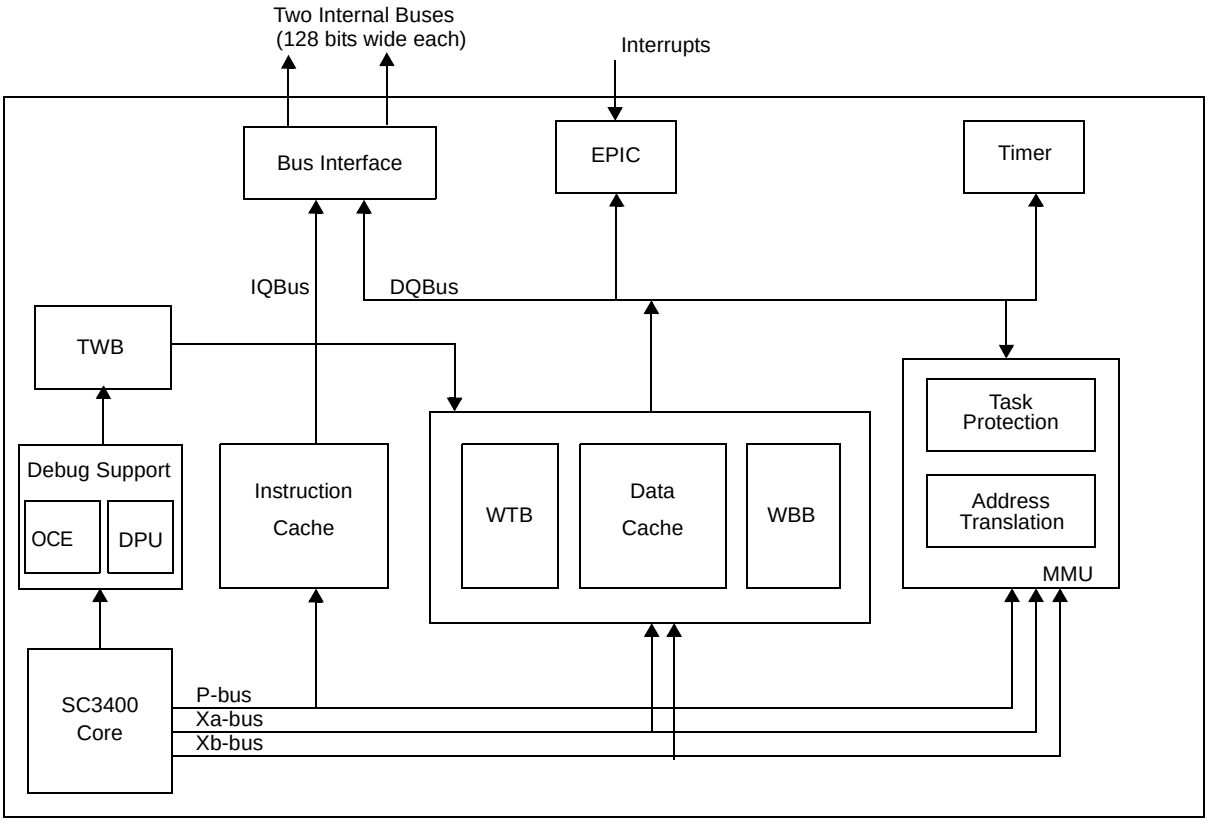


Figure 1-3. StarCore SC3400 DSP Subsystem Block Diagram

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 SC3400 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.

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 SC3400 DSP Core

The SC3400 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, MSC8122, MSC8126, and the MSC711x family, and it introduces many new features and enhancements.

The SC3400 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 SC3400 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 a 16-bit × 16-bit + 40-bit multiply-accumulate unit, 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 800 MHz/1 GHz can perform up to 3.2/4 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 102.4/128 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 SC3400 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). In addition, the SC3400 supports special instructions to support special operations, such as Viterbi and video applications.

Although the SC3400 is a DSP, the rich instruction set also gives special attention to control code, making the SC3400 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 SC3400 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 SC3400 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 SC3400 core includes a dynamic branch prediction mechanism that contains a 32-entry branch target buffer (BTB) to improve performance by reducing the change of flow latency.

1.4.2 L1 Instruction Cache

The ICache with real-time support is highly optimized for real-time DSP applications and minimizes miss ratios, latencies, bus bandwidth requirements, and silicon area. The 16 KB ICache is 8-way set associative. Each of the 8 ways contains eight 256 bytes long lines and is divided into 16 fetch sets (VBRs), each with an associated valid bit. The 3-bit index field of the address serves as an index to the line within the way. The tag field in the address selects the line with the matching tag.

When a cache miss occurs, new data is fetched in bursts from the target memory. Descriptors inside the MMU indicate the burst size. For example, accesses to the M2 may occur in bursts of four beats, and accesses to the DDR may occur in bursts of two beats. There is also an option to have an interruptible pre-fetch to the end of the line. This option, referred to as prefetch, takes advantage of the spatial locality of the code. When a new fetch is required and all the ways of the

matching index are valid, one of the cache lines is invalidated using the pseudo least recently used (PLRU) algorithm and it is replaced by the new data.

In some operations, cache thrashing can occur. For example, suppose process A must be preempted in favor of process B. While process B runs, the instructions of process A are thrashed. When process B finishes and process A takes over, process A may not find its most recently used instructions in the cache. Portions of the cache can be locked dynamically and released to reduce the effect of cache thrashing in a multitasked application. All cache entries can be invalidated by writing a cache invalidate command from the SC3400 core. This is useful, for example, when new code is written to lines in the M2 memory that are already cached.

1.4.3 L1 Data Cache

The 32 KB DCache with DSP real-time support (DCache) is 8-way set associative. Each of the 8 ways contains sixteen 256-byte long lines and is divided into 16 fetch sets, each with an associated valid bit and dirty (modify) bit. The 4-bit index field of the address serves as an index to the line within the way. The line whose tag matches the tag field of the address is the selected line. When a cache miss occurs, the new data is fetched in bursts of fetch sets. As in the ICache, there is an option to have an interruptible pre-fetch until the end of the line and the burst size is programmable through descriptors in the MMU. When there is a need to fetch new data to the cache and all the ways of the matching index are valid, one of the lines of the cache is thrashed into the WBB using the pseudo least recently used (PLRU) algorithm. It is then written back into memory in programmable burst lengths. Portions of the cache can be locked dynamically and released to reduce the effect of cache thrashing in a multitasked application.

The DCache uses the following three software-triggered operations for software coherency:

- *Invalidate memory zone*. All lines belonging to the specified zone and a specific task are treated as invalid. This instruction is used when the data in the cache is no longer valid—for example, when the I/O device brings new data into the MSC8144E memory.
- *Flush memory zone*. The DCache invalidates and writes to the MSC8144E memory all lines belonging to the specified zone and a specific task. This instruction is typically used when the SC3400 core needs to determine whether all the data resides in the destination memory before another SC3400 core or an I/O device can use the data.
- *Synchronize memory zone*. The DCache writes all the lines of the specified zone and a specific task to the MSC8144E memory without invalidating them. This instruction is useful when the SC3400 core needs to ensure that the data resides in the destination memory but still may need to use it from the cache.

Any of these instructions run in the background while the SC3400 core is not stalled. The SC3400 can determine that the instruction has finished executing by polling status or interrupt.

1.4.4 Memory Management Unit (MMU)

The memory management unit (MMU) controls external memory accesses and translates virtual addresses to physical addresses.

The MMU is responsible for three main functions:

- Supplying program and data access hardware protection for two privilege levels (user and supervisor) to enable system reliability and easier code development up to an open system approach.
- Implementing a high-speed address translation mechanism that translates from virtual to physical addresses to support memory relocation and a modular system-independent software development flow.
- Providing cache (L1 and L2) and bus controls for advanced memory management.

Following are some examples of common problems that are solved by simple block translation:

- Typically, a non-relocatable code is preferred. Simple address translation between the virtual and the physical address can assure that it executes without change after it is swapped from the DDR memory to an arbitrary location in internal MSC8144E memory. Code does not depend on physical memory allocation and can be written modularly.
- Relocatable code is not immune to addressing problems. Global variables can be addressed relative to the code. However, in a multi-core device such as MSC8144E in which all the cores may share the same code, these global variables should be addressed differently depending on the core.

The MMU enables the system designer to integrate system resources better and to define a cleaner software model. For example, the registers defining protection 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.

MMU protection enables implementation of a protected software architecture in which system code and task code is protected from corruption by an errant task. This increases the mean time between failures (MTBF) and accelerates the system debugging.

1.4.5 Debug and Profiling Unit (DPU)

Debug and profiling are supported in the DSP platform with two units:

- The on-chip emulator (OCE)
- The debug and profiling unit (DPU)

The OCE supports core-coupled debug functions such as breakpoint detection, forcing the core into debug mode, single stepping, and program trace generation. The functionality of the OCE is similar to that of the EOnCE unit in the SC140 architecture used in the MSC81xx DSP family and the OCE10 module in the SC1400 architecture used in the MSC711x DSP family.

The DPU supports the debugging and profiling of the SC3400 DSP subsystem in cooperation with the OCE. It includes two main functions:

- Set of 6 counters that enable to count a selection of platform level events.
- A trace unit that processes the OCE program trace information, adding additional information to it.

The six counters can be programmed to count down on a selection from a wide array of platform events. Each counter can generate an interrupt upon reaching zero or signal the core to enter Debug mode. The counters can be enabled and disabled by writing to their control register or by an event such as a watchpoint detected by the OCE or execution of the debug-oriented core instructions MARK and DEBUGEV. The counted events can be filtered so that only events that occur during a specific task (marked by its task ID in the MMU) are counted.

The main counted events are:

- Number of hits and misses in the ICache or in the DCache.
- Number of thrashes in the ICache and in the DCache.
- Reasons for core stall cycles.
- Different bus contentions.
- MBus profiling: cycles where the bus is busy, waiting for service, or idle.
- Number of interrupts that were serviced.
- Number of cycles the core was in the Wait processing state.
- Number of cycles a certain task executed (total execution cycles to completion and the actual number of cycles), and the number of times it was swapped out.

Most events are organized in configured sets of events for easy configuration.

The trace unit generates and manages trace writes to the virtual trace buffer (VTB), a user-defined system memory area to which the trace data is written. A selection of trace writing policies implements protocols such as continuous double-buffered DMA uploads, continuous writes for failure point history, and more.

The trace data can be one of the following:

- The OCE program trace, unchanged.
- The OCE program trace, compressed so that software and hardware loops are reported only once with the number of iterations. Task ID changes are reported as well.
- The OCE program trace (in subroutine and interrupt modes only), in which a snapshot of the 6 counters is added upon each core trace entry (jump/return from every subroutine and/or interrupt). Task ID changes are also reported.
- User-requested or OCE-triggered snapshot of the six registers to trace the task ID.
- User data written by the core software to a specified DPU memory mapped address.

1.5 Chip-Level Arbitration and Switching System (CLASS)

The CLASS is the central interconnect system for the MSC8144E device. The CLASS is a non-blocking, full fabric crossbar switch, that gives any initiator access to any target in parallel with another initiator-target couple. The CLASS demonstrates per-target prioritized round-robin arbitration, highly optimized to the target characteristics.

The CLASS initiators are:

- Four SC3400 DSP subsystems
- L2 ICache
- DMA
- TDM
- Serial RapidIO subsystem
- QUICC Engine module subsystem
- PCI Controller

The CLASS targets are:

- M2 memory
- M3 memory
- DDR controller
- PCI controller
- Configuration control and status registers (CCSRs)

The CLASS operates at 400 MHz, separate from the SC3400 core subsystem frequency, as an optimized trade-off between power dissipation, memory technology, and miss latency. Controlling the inter-device data flow, the CLASS reduces bottlenecks and imposes high bandwidth and fully pipelined traffic.

1.6 Memory System

The memory system encompasses the M2 memory, M3 memory, the L2 ICache, and the DDR controller, all of which are discussed in this section.

1.6.1 M2 Memory

Figure 1-4 shows the M2 memory architecture. The 512 KB memory performs 128-bit wide accesses at 400 MHz and is accessible to any of the DSP core subsystems or any other initiator in the system (L2 ICache, DMA controller, TDM, QUICC Engine module subsystem, Serial RapidIO subsystem, PCI). The memory is partitioned into four 128 KB memory groups to allow four simultaneous accesses. The group addresses are interleaved with 256-byte interleave resolution which is optimized to minimize contentions between accesses. The M2 memory uses SRAM technology and supports both single and burst accesses. The M2 memory is fully ECC protected. The M2 memory supports partial accesses. Automatic Read-Modify-Write accesses are generated to maintain the ECC protection.The M2 memory is volatile after reset.

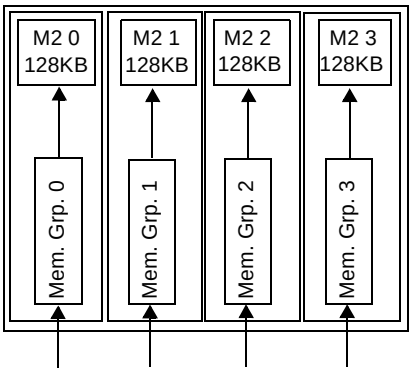


Figure 1-4. M2 Memory Architecture

1.6.2 M3 Memory

The 10 MB M3 memory can be used for both program and data and eliminates the need for an external memory in a variety of applications, thus reducing board space, power dissipation, and cost. The M3 memory has a 128-bit wide port and runs at 400 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. The M3 memory is fully ECC protected.

1.6.3 L2 Instruction Cache

The shared level 2 multi-port interleaved ICache is highly optimized for multi-core DSP applications and minimizes miss ratio, latencies, bus bandwidth requirements, and silicon area. The 8-way associative 128 KB L2 ICache contains two 64 KB banks. The line size is 256 bytes. When a cache miss occurs, the new data is fetched in a burst from the target memory. An option to fetch to the end of the line (prefetch) takes advantages of the spatial locality of the code. L2 ICache entries are invalidated via a cache invalidate command in the same way as in the SC3400 L1 ICache, which is useful when new code is written to M2, M3 or DDR memory that is already cached. All DSP subsystem instruction addresses are interleaved to two L2 ICache ports, so they can service multiple requesters concurrently if they access different interleaved banks.

1.7 Clocks

The MSC8144E device has three input clocks:

- A shared input clock for the system PLL, core PLL, and global PLL.
  - The core PLL can get an input clock either from the “shared input clock” or from the output of the system PLL (cascaded).
  - The global PLL can get an input clock either from the “shared input clock” or from an input pin dedicated for PCI.
- An input clock for the global PLL, dedicated to PCI.
- A differential input clock for the serial RapidIO PLL.

The MSC8144E device includes four PLLs:

- System PLL to clock the CLASS, the M2 memory, and all the other blocks in the device, except for those clocked by the other PLLs.
- Core PLL to clock the cores.
- Global PLL to clock the DDR controller (and the external DDR SDRAM device) and the PCI sub-system and the M3 sub system. Each of these blocks can select a clock from either the Global PLL or the System PLL.
- Serial RapidIO PLL to clock the RapidIO SerDes.

The ratios between the system PLL, the core PLL, and the global PLL 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.

1.8 DDR Controller (DDRC)

The DDR SDRAM interface is useful when the channel storage size is relatively big (as for a modem) and also when more channels are required to supplement the internal memory. When the MSC8144E device works with channel data stored in the DDR SDRAM, the DMA controller can swap the data to and from the M2 memory, thus enabling the L1 DCache to fetch from M2 memory instead of accessing the DDR SDRAM memory directly. Fetch latency is thus reduced, significantly improving the average clock cycles required per task. The M2 and M3 memories are large enough to accommodate the number of channels processed by the DSP subsystems for a variety of packet telephony and wireless transcoding application, such as basic G.711 voice coding, G.729 or G.723 premium voice coding, AMR, and EFR. However, it is not large enough for such memory-consuming applications as a V.90 modem, especially when high channel densities are required. For these applications, the MSC8144E can interface with JEDEC-compliant DDR1 or DDR2 SDRAM devices. A DDR SDRAM can be used not only as an extension for the M2 and M3 memories but also to store code. In a typical application, infrequently used code is either swapped into M2/M3 memory when needed or executed directly from an external DDR SDRAM. The DDR SDRAM interface frequency is decoupled from the DSP subsystem frequency, and it has a separate PLL to deliver the required frequency according to the bandwidth requirements. It is 16/32 bits wide and can interface with up to two 16-bit wide devices, or four 8-bit wide devices. It has a separate strobe per byte. Two logical banks (chip select) are supported, each with logical programmable bank start and end addresses. A bank size of up to 128 MB is supported. Programmable parameters allow for a variety of SDRAM organizations and timings. Using data mask bits, the SDRAM controller enables partial write operations to bytes in a word or words in a burst. Optional ECC protection is provided for the DDR SDRAM data bus. Using ECC, the memory controller detects all two-bit errors and corrects all single-bit errors within the 32-bit data bus. For ECC, an additional ECC DDR SDRAM device is usually needed. Both the data DDR SDRAM and the ECC DDR SDRAM should have the same CAS latency. There is page retention for up to four simultaneous open pages, and the number of clocks for which the pages are kept open is programmable. Pages are replaced using a pseudo-LRU replacement algorithm.

1.9 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.

1.10 TDM

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

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

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

at twice the rate, so their buffer size is twice that of the transparent channels. For receive, the buffers of specific TDM interface fill at the same rate and therefore share the same write pointer relative to the beginning of the buffer. When the write pointer reaches a predetermined threshold, an interrupt to the SC3400 core is generated. The SC3400 core empties the buffers while the TDM continues to fill the buffers until a second threshold line is reached and then an additional interrupt is generated to the SC3400 core. The SC3400 core empties the data between the first and the second threshold lines. Both the first and the second threshold lines are programmable. Using these threshold lines, the SC3400 core and the TDM can perform a double-buffer handshake. For transmit, the SC3400 core fills all the buffers of a TDM interface, and the TDM empties them. A similar method employing two threshold line interrupts is used for a double-buffer handshake between the SC3400 core and the TDM. You can program the interrupt as either shared for receive and transmit or separated.

1.11 QUICC Engine Subsystem

The MSC8144E QUICC Engine module is a versatile communications engine based on a subset of the Freescale QUICC Engine technology that integrates several communications peripheral controllers.

Note: See the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM) for functional, register, and programming details.

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
  - Communications controller 5 supports the ATM protocol through UTOPIA interface
- Interrupt controller
- Multiplexer and timers logic
- Baud-rate generators

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

Figure 1-5 shows the MSC8144E QUICC Engine module block diagram.

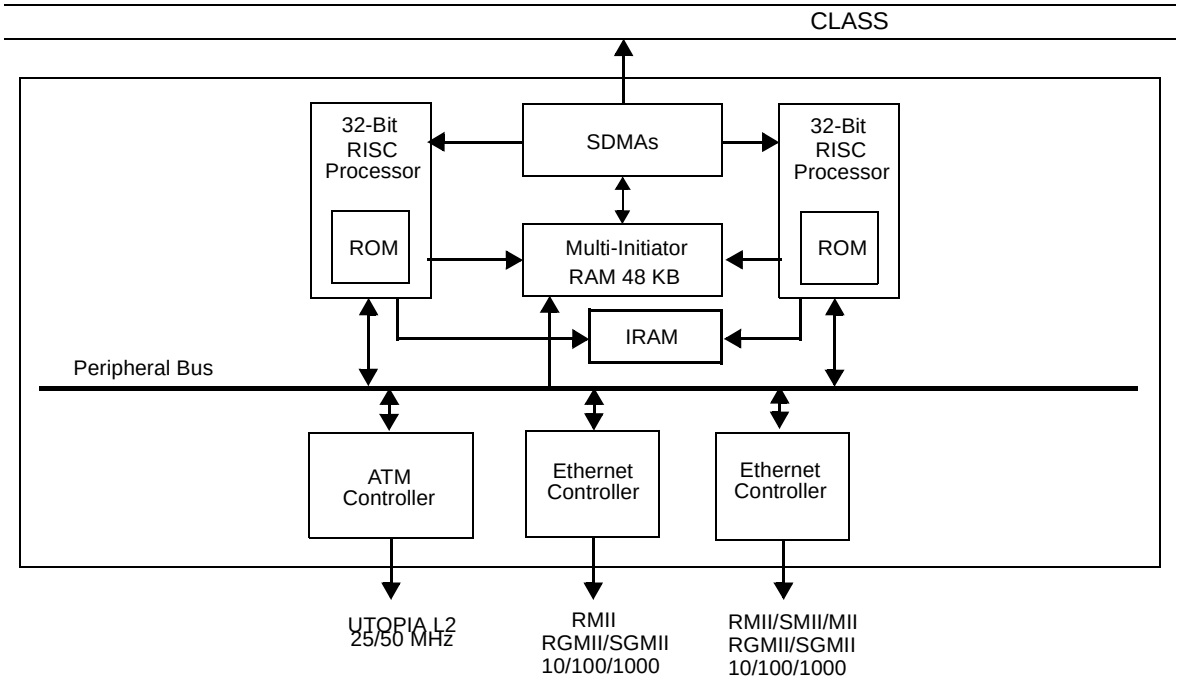


Figure 1-5. QUICC Engine Module Block Diagram

1.11.1 Ethernet Controllers

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

- 10/100 Mbps MII (one controller only)

- 10/100 Mbps RMII (consortium standard)
- 10/100 Mbps SMII
- 1000 Mbps SGMII with SerDes support
- 10/100/1000 Mbps RGMII (full duplex only)

The media-independent interface (MII) provides a standard interface between the MAC layer and the physical layer. It isolates the MAC layer and the physical layer so that the MAC layer can be used with various physical layer implementations. The RMII interface (a reduced pin count MII) is specified by the RMII Consortium standard as a low cost alternative to the MII, using 8 pins instead of 16 (excluding the MDC and MDIO signals). The SMII interface further reduces the pin count to 6 signals per port (excluding the MDC and MDIO signals), which greatly simplifies switch design and reduces board space and overall design and layout costs.

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/RMII/SMII) 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.

After a hardware reset, the software driver initializes the parameter RAM and the configuration register, and then sets MACCFG1[RX_EN]. The Ethernet receiver is enabled and immediately starts processing receive frames. The MAC hardware checks when RX_DV is asserted and as long as COL remains deasserted (full-duplex mode ignores COL), it looks for the start of a frame by searching for a valid preamble/SFD (start of frame delimiter) header, which is stripped and the frame begins to be processed. If a valid header is not found, the frame is ignored. Part of the preamble is user configurable, to allow for user configurable preamble.

If the receiver detects the first bytes of a frame, the Ethernet Controller begins to perform the frame recognition function. Two modes are supported: MSC8122/MSC8126 backward-compatible frame filtering, and extended frame filtering mode. The Extended filtering supports large memory based lookup tables, thus replacing the need for an external CAM device. In MSC8122/MSC8126 mode, the receiver can also filter frames based on physical (individual), group (multicast), and broadcast addresses. If a frame is accepted, the Ethernet controller fetches a receive buffer descriptor (RxBD) from FIFO. If RxBD is not being used by the software (RxBD[E] is set), the controller starts transferring the incoming frame. RxBD[F] is set for the first RxBD used for any particular receive frame. When the buffer is filled, the Ethernet

controller clears RxBD[E] and, if enabled, the controller generates an interrupt. If the incoming frame is larger than the buffer, the Ethernet controller fetches the next RxBD in the table. If it is empty, the controller continues receiving the rest of the frame. In half-duplex mode, if a collision is detected during the frame, no RxBDs are used; thus, no collision frames are presented to the user except late collisions, which indicates LAN problems.

1.11.2 ATM Controller

The ATM controller provides the ATM and AAL layers of the ATM protocol using the universal test and operations physical layer (PHY) interface for ATM (UTOPIA level II) for target mode only. It performs segmentation and reassembly (SAR) functions of AAL5, AAL2, and AAL0, and most of the common parts of the convergence sublayer (CP-CS) of these protocols. For each virtual channel (VC), the ATM pace control (APC) unit generates a cell transmission rate to implement constant bit rate (CBR), variable bit rate (VBR), unspecified bit rate (UBR), or UBR+ traffic. To regulate VBR traffic, the APC unit performs a continuous-state leaky bucket algorithm. The APC unit also uses up to eight priority levels to prioritize real-time ATM channels, such as CBR and real-time VBR, over non-real-time ATM channels such as VBR and UBR.

1.11.3 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.

1.12 PCI

The PCI interface connects the MSC8144E device to a 33 or 66 MHz, 3.3 V PCI bus to which the I/O components are connected. The PCI interface complies with the *PCI Local Bus Specification*, Revision 2.2. The PCI interface has a 32-bit multiplexed address/data bus, plus various control and error signals. It supports address and data parity with error checking and reporting. As an initiator, the PCI interface manages read and write transactions to the PCI memory space, the PCI I/O space and to the 256 byte PCI configuration space. As a target, the PCI interface handles read and write operations to local memory and internal registers and to the 256 byte PCI configuration registers.

1.13 Serial RapidIO Subsystem

RapidIO is a high-performance, point-to-point, low-pin-count, packet-switched system-level interconnect that can be used in a variety of applications as an open standard. The RapidIO architecture provides a rich variety of features including high data bandwidth, low-latency capability, and support for high-performance I/O devices, as well as providing message-passing and software-managed programming models. The Serial RapidIO subsystem consists of a Serial RapidIO controller and a RapidIO Message Unit (RMU). The MSC8144E device can either connect directly to a host or to a Serial RapidIO switch. Each port in the switch is point-to-point connected to the MSC8144E device through a serial RapidIO link. This link typically carries packets in both directions. Packets ready for processing are transported from the host to the MSC8144E, and the processed packets are transported from the MSC8144E back to the host.

1.13.1 Serial RapidIO and Host Interactions

The Serial RapidIO controller directs the traffic flow between a host processor and the MSC8144E device through the RMU. The host and the MSC8144E communicate as follows:

- The host may send messages to the destination MSC8144E device, which are sent back to the host after processing along with a short doorbell interrupt to indicate that the packets have been processed.
- Messages eliminate the latency of read accesses. The host writes to the MSC8144E and the MSC8144E writes to the host. In addition, messages can be used in applications where the host does not know the internal memory structure of the target DSP.
- The host may directly access the data in the MSC8144E memory for both reads and writes. It handshakes with the software running on a DSP core through a ring of descriptors in MSC8144E memory.
- The host may access the data in MSC8144E memory for both reads and writes, but instead of maintaining a ring of descriptors in the MSC8144E memory, it uses buffer descriptors (BDs) that are messaged from the DSP core to the host.
- The host may put all the data buffers into its memory and have the MSC8144E access the data.
- Any initiator on the RapidIO system can access any internal memory space as well as the DDR SDRAM using NREAD, NWRITE, MESSAGE, and DOORBELLS. In addition, it can configure the RapidIO messaging and configuration unit using MAINTENANCE packets.

In the receive path, the following steps occur:

1. The clock is recovered using a dedicated PLL.
2. The data is deserialized, 8b/10b decoded, checked for the correct CRC, and passed to the higher-level protocol logic.

3. The control symbols are stripped and used to interface with the other peers without core intervention.
4. NWRITE packets are written to the destination memory.
5. MESSAGES are directed to the RapidIO messaging unit from which they are forwarded to their destination queue/memory location.
6. RESPONSE messages are associated with their respective NREAD and go back to the internal initiator that initiated the transaction.

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

1.13.2 RapidIO Messaging Unit (RMU) Operation

The messaging unit is divided into five parts:

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

The message receiver performs the following steps:

1. Filters the received packets into multiple queues (controllers) based on selected (programmable) fields in the RapidIO message header (for example, mailbox number and letter number). This filtering mechanism can be used for filtering the messages to the different SC3400 cores or filtering the messages according to their size to the right queue.
2. Writes the message to a receive buffer pre-allocated by the SC3400 core.
3. Post-increments the buffer write pointer.
4. Optionally interrupts the SC3400 core. The core can then read the buffer, process the message data, and update the read pointer of the buffer by writing it to the messaging controller.

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

The message transmitter performs the following steps:

1. The SC3400 core sets the registers in the controller with the message parameters (read pointer, message length, destination, buffer size, available messages.)
   - Optionally, the SC3400 core initiates only a pointer to a BD queue.
   - The messaging controller reads the BD that includes the message parameters.
2. The controller reads data from memory according to predefined parameters.
3. The controller encapsulates the message and transfer it to a RapidIO endpoint.
4. The RapidIO endpoint sends the message.
5. Acknowledges are transferred from the RapidIO endpoint to the RMU.
6. Upon completion of a message the controller can:
   - Send an interrupt to the core, waiting for a new sets of parameters.
   - Proceed to the next message in queue according to the previous parameters.
   - Proceed to the next BD in queue.

The doorbell transmitter performs the following steps:

1. The SC3400 core sets the registers in the controller with the doorbell parameters (doorbell data, destination)
2. The controller encapsulates the doorbell and transfers it to the RapidIO endpoint.
3. The RapidIO endpoint sends the message.
4. Acknowledges are transferred from the RapidIO endpoint to the RMU.
5. Upon completion of a message the controller can send an interrupt to the SC3400 core and wait for a new sets of parameters.

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 SC3400 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 SC3400 core or external host should read the UART status register to identify the interrupt source and service it accordingly.

1.16 Timers

The MSC8144E device contains 16 identical 16-bit timers divided into four groups. Each group (TMR) contains four identical 16-bit timers, each with a prescaler, a counter, a load register, a hold register, a capture register, two compare registers, two status registers, and a control register. In addition, each SC3400 subsystem includes two general purpose 32-bit timers. The MSC8144E device also includes 5 software watchdog timers. Each of the software watchdog timers can be used by any of the cores within MSC8144E 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 SC3400 core/host/task has a unique predefined lock number (8-bit code). When trying to lock the semaphore, the SC3400 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 SC3400 core/host/task releases the semaphore by writing a 0 to it.

1.18 Virtual Interrupts

The global interrupt controller generates 32 virtual interrupts, with eight interrupts per SC3400 core. An interrupt is generated by a write access of each SC3400 core or by an external host CPU. A virtual $\overline{\text{NMI}}$ can also be generated by a write access.

1.19 I^2C Interface

The inter-integrated circuit (I^2C) controller enables the MSC8144E 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 (SDA) and serial clock (SCL), carry information between the integrated circuits connected to it.

1.20 GPIOs

The MSC8144E 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 MSC8144E peripheral functions

multiplexed with the GPIO ports are grouped to maximize the usefulness of the ports in the greatest number of MSC8144E applications.

1.21 Boot Options

The boot program in the internal boot ROM initializes the MSC8144E after it completes a reset sequence. The MSC8144E device can boot from an external host through the serial RapidIO or PCI interfaces or download a user boot program through the I^2C or Ethernet ports.

1.22 JTAG

The dedicated user-accessible test access port (TAP) is fully compatible with **IEEE** Std. 1149.1. The MSC8144E device supports circuit-board test strategies based on this standard. For details on the standard, refer to the standard documentation.

1.23 Security Engine (SEC)

The Security Engine (SEC) combines a master controller for the internal bus with four data channels, six execution units, and a shared random number generator (RNG) to provide encryption/decryption capability without imposing process loading on the core processor either for encryption processing or data transfer and management. The six execution units (DEU, AESU, MDEU, AFEU, PKEU, and KEU) perform specific mathematical manipulation required by protocols using cryptographic processing. The unit is optimized to process all 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), and the **IEEE** 802.11i security standard.

1.24 Development Environment

Freescale will supply a complete set of DSP development tools for MSC8144E. Our tools are focused on providing easier and more robust ways for designers to develop optimized DSP systems. Whether the application targets a wireless base station, IP telephony, or transcoding media gateways, the development environment gives the designers everything they need to exploit the advanced capabilities of the MSC8144E architecture.

The MSC8144E tool components will 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.
- *Software Simulator*. Full chip simulation; the developer can design an application and run it on the simulator before running it on the silicon. FCS integrated under IDE, the simulator provides customers with tools to create projects and debug them as they would on silicon.
- *Profiler*. The developer can analyze and identify program design inefficiencies.
- *Data Visualization*. Lets the developer graph variables, registers, regions of memory, and HSST data streams as they change over time. By changing the visualization filter, you can plot this data in a variety of ways; including line charts, logarithmic charts, polar coordinates, and scatter graphs.
- *High Speed Run Control*. PowerTAP high speed host-target interface allows users to program in Flash, ROM, and cache.
- *Host Platform Support*. Microsoft Windows and Solaris.
- *Development Board*. The application development system (ADS).
- *Kit for MSC8144E*. A complete system for developing and debugging real-time hardware and software. The kit includes the MSC8144E device with a companion memory connected to a RapidIO switch and connects to a host processor, JTAG debug interface, serial and Ethernet interface, and software device drivers.

1.25 Application Software

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

Table 1-2. Application Software Modules

Application	Modules
ETSI/3GPP1 Vocoders	GSM-FR
	GSM-HR
	GSM-AMR/EFR
	3GPP AMR-WB
TIA/3GPP2 Vocoders	IS127 EVRC
	IS893 SMW

Table 1-2. Application Software Modules (Continued)

Application	Modules
ITU G.7xx Vocoders	G.711
	G.711 App. 1 and 2 (PLC and VAD/CNG)
	G.722
	G.723.1
	G.726
	G.726A
	G.728
	G.729B
	G.729AB
	G.729E
Modems	Pumps: • V.23 CallerID • V.34 • V.90 • V.92
	V.42 MNP4 (error correction)
	Compression: • V.44 • V.42bis • MNP5
	Negotiation: • V.8 • V.8bis
	HDLC
	Relay: • V.150.1 (MoIP)
Fax	Pumps: • V.17 • V.27ter • V.29
	Relay: • T.30 (FoIP) • T.38
Echo cancellation	G.165
	G.168 (64 ms)
	G.168 2004 (128 ms, windowed)
	Noise reduction
	Acoustic level control
	Acoustic EC (roadmap)
Telephony support	DTMF detection
	Universal tone generation
	Special tone event detect
	VAD/CNG
	PLC
	RTP packetization
Security	AES
Video	MPEG4
	H.263
	H.264
	H.324MDSP (roadmap)

Table 1-2. Application Software Modules (Continued)

Application	Modules
Device Driver	• DMA driver • Serial RapidIO driver • TDM driver • Ethernet driver • PCI driver • UTOPIA driver • UART driver • Memory allocation • Interrupt handling

In addition, Freescale provides a complete VoIP framework that supports channel scheduling, explicit DMA management of channel state data, memory management, network termination, peripheral drivers, framework control API command/status, and more. The Framework is shown in **Figure 1-6**. Notice that physical drivers are encapsulated in a SmartDSP OS, thus ensuring ease of transfer between different hardware implementations.

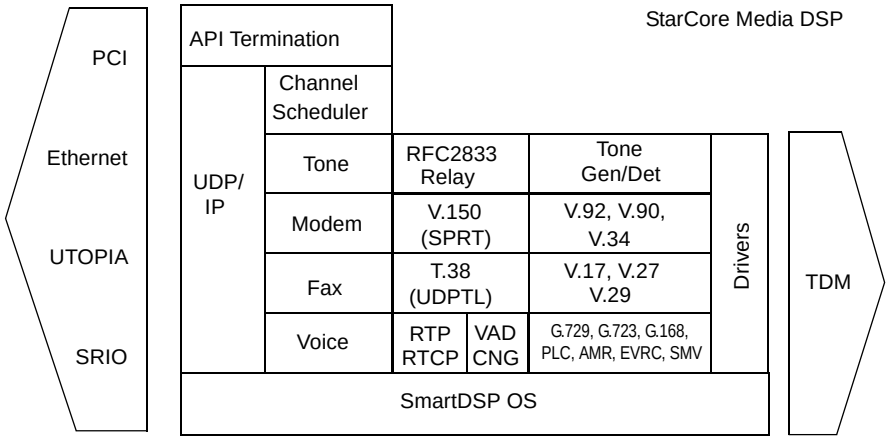


Figure 1-6. Application Software Model

1.26 Example Applications

The applications covered in this section are as follows:

- *Application 1*. Generic System Block Diagram.
- *Application 2*. Legacy Farm Example using UTOPIA and TDM.
- *Application 3*. System solution.
- *Application 4*. SerDes connectivity.

1.26.1 Application 1

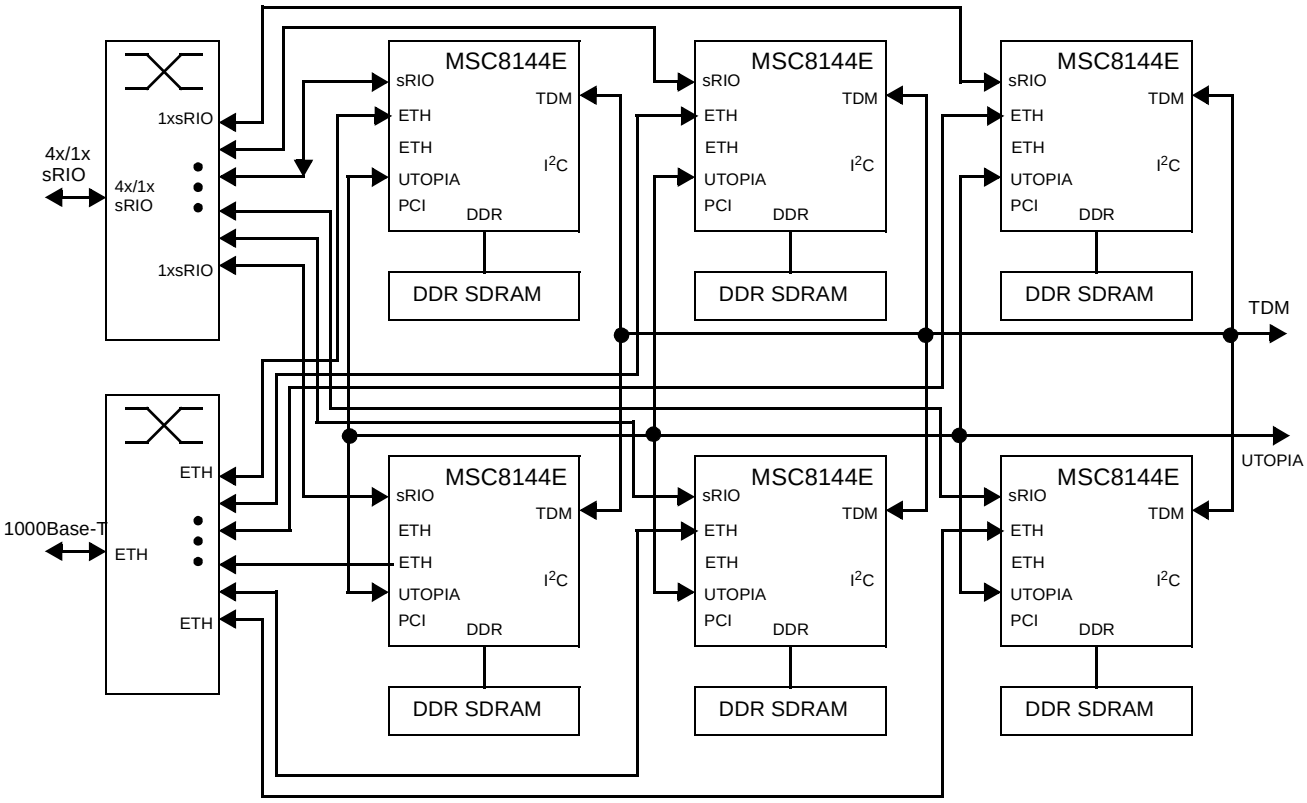


Figure 1-7. Generic System Block Diagram

MSC8144E promotes easy system building for legacy, transition, and generation system interface requirements via its flexible and varied I/O protocol support. A generic system block diagram is shown in **Figure 1-7**. For completeness, DDR SDRAMs are shown attached to the MSC8144E devices, but for most applications the onboard memory is all that is required. This is true of the examples that follow.

1.26.2 Application 2

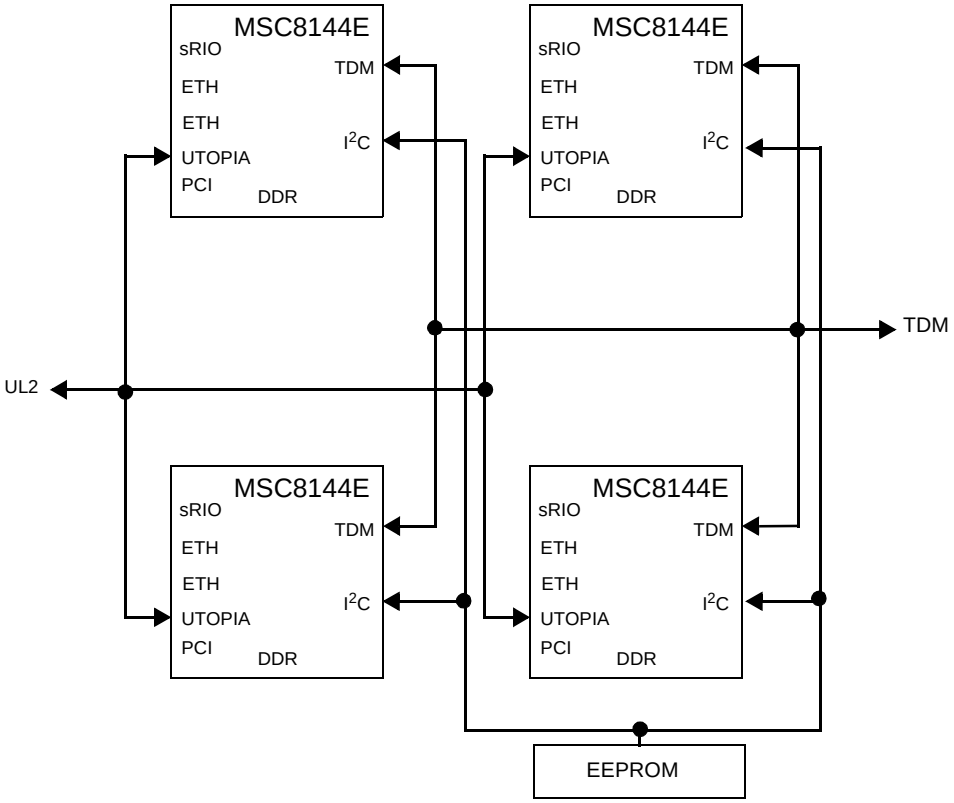


Figure 1-8. DSP Farm Example using UTOPIA and TDM

In this system, the UTOPIA level 2 provides the packet (cell) based interface via either AAL2 or AAL5. To provide the PCM side, TDM is used. The UL2 interface on MSC8144E provides parsing into the IP packet contained in the AAL5 cps. For AAL2, it can support multiple AAL2 cps packets per cell. In both cases, each SC3400 core can have individual buffer rings. In this example, a simple TDM is chosen.

1.26.3 Application 3

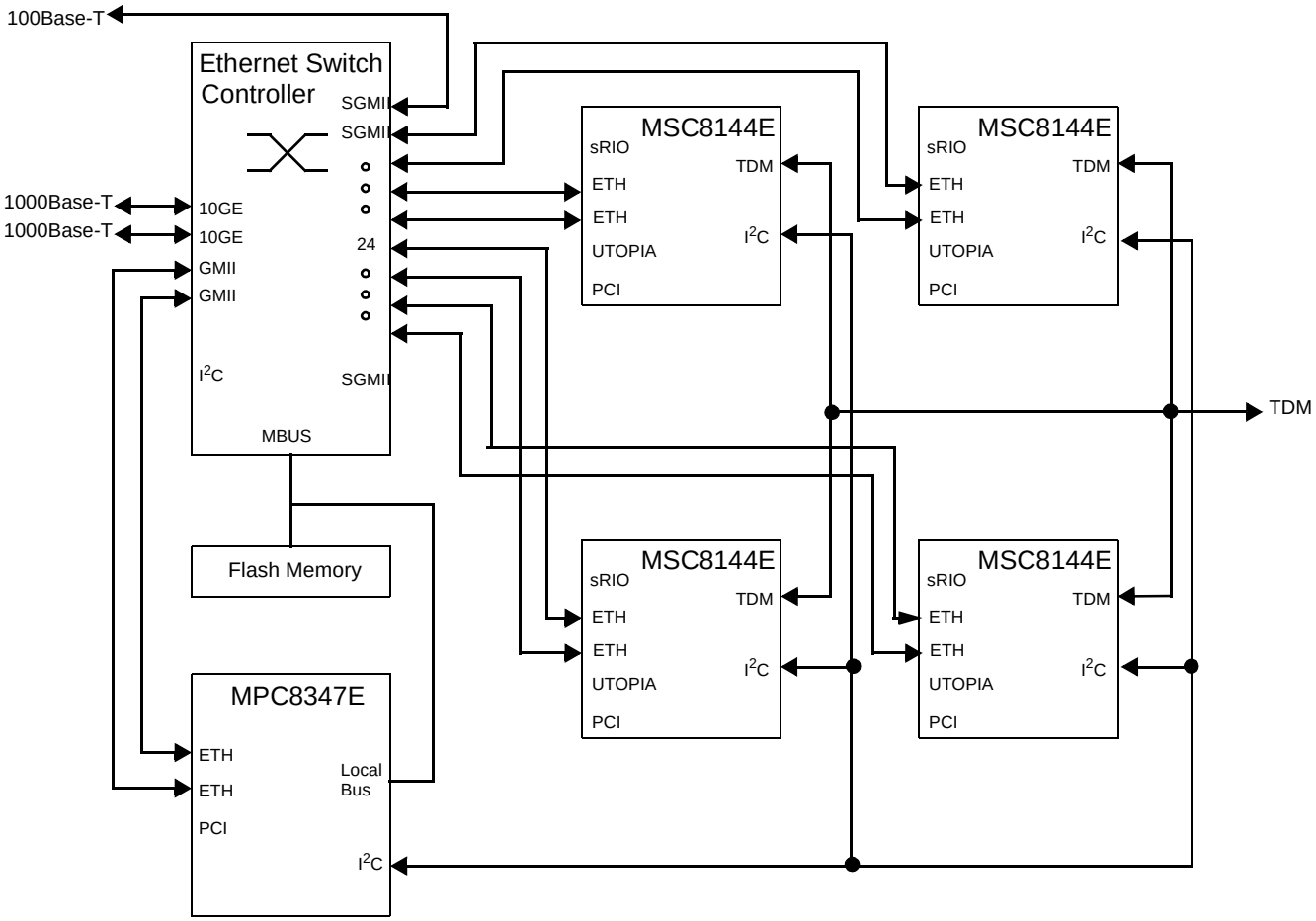


Figure 1-9. System Solution

In this example, Ethernet is used as the packet interface. To use high volume commodity switches for aggregation, a 1000Base-T is used in MAC-to-MAC configuration. A PowerQUICC device connected to the switch provides ingress routing on UDP destination port number and manages the switch so that the egress data path does not go through the PowerQUICC device. TDM is used for the PCM side. For such solutions, no time-slot assigner is required because the MSC8144E devices can interface with an H.110-like bus.

1.26.4 Application 4

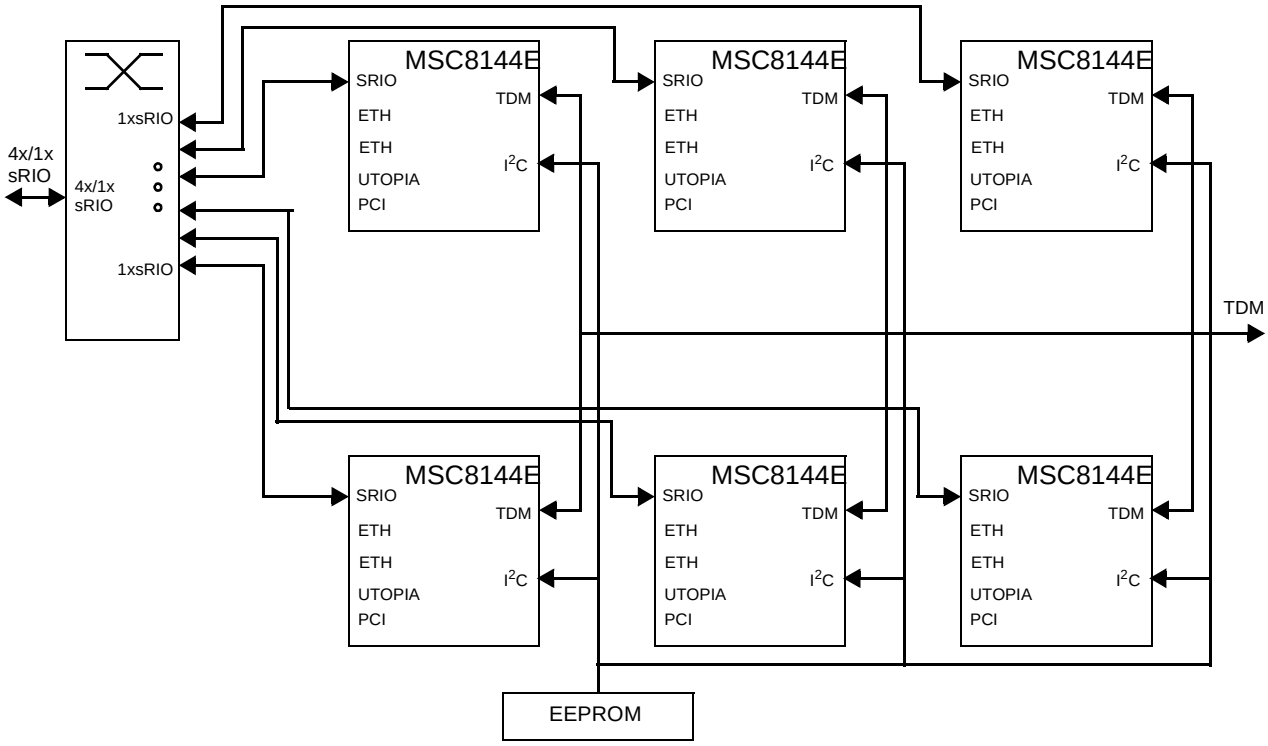


Figure 1-10. SerDes Connectivity

This example considers the SerDes interface. For MSC8144E, the interface can use either a gigabit Ethernet (SGMII) or Serial RapidIO. Thus, MSC8144E is AMC/advanced TCA ready. In this example, an H.110 TDM bus is used. For AMC, this bus would be on the extended connector with the SerDes in the fabric area.

1.26.5 Application 5

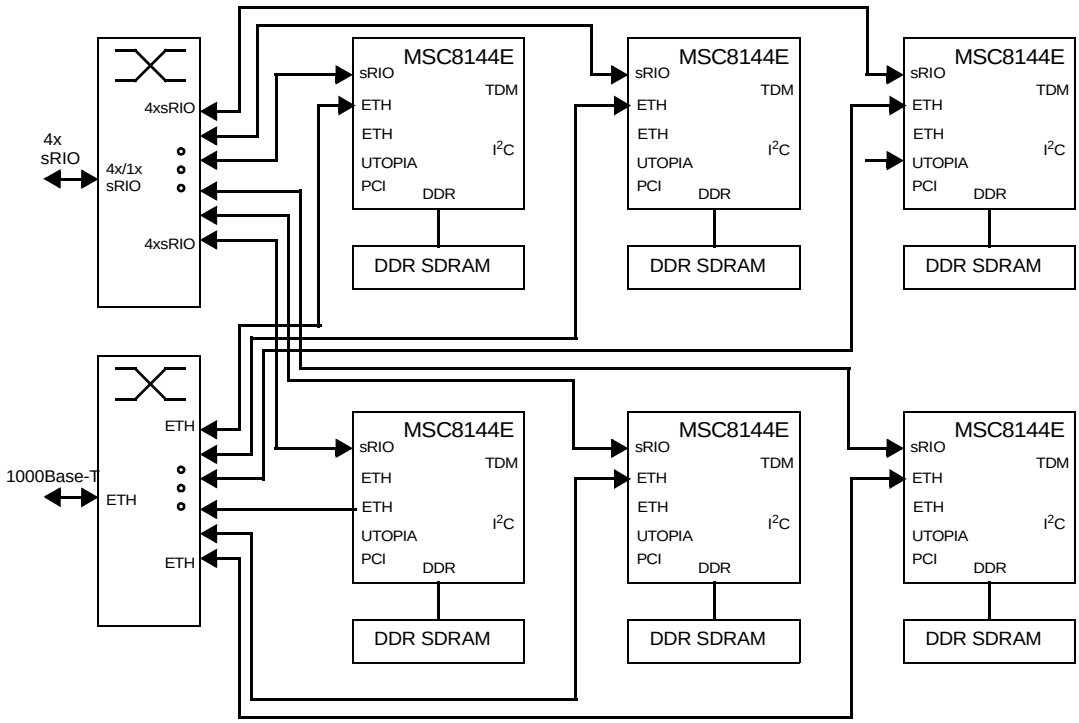


Figure 1-11. RNC Traffic Processing Board That Executes RNL Block Diagram

An application blade server that executes the radio network layer (RNL) to process RNC traffic using the MSC8144E is shown in is shown in **Figure 1-11**. This example is a farm of MSC8144E devices with external memory. The blade example shows both SerDes type interfaces on the card; however, only one type of interface is required. The system can self-boot over either the serial RapidIO interface or Ethernet.

SC3400 Core Overview 2

The SC3400 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 and the SC1400 core used in the MSC711x DSP family and delivers up to 3200/4000 16-bit MMACS using an internal 800 MHz/1 GHz clock at 1 V. Each core includes:

- Data arithmetic and logic unit (DALU) containing 4 ALUs.
- Address generation unit (AGU) containing two address arithmetic units.
- Up to six instructions execute in a single clock cycle.
- Variable-length execution set (VLES) that can be optimized for code density and performance.
- 16 data registers, 40 bits each.
- 27 address registers, 32 bits each.
- Hardware support for fractional and integer data types.
- Four hardware loops with zero overhead.
- Very rich 16-bit wide orthogonal instruction set.
- Application-specific instructions for Viterbi and multimedia processing.
- Special single instruction, multiple data (SIMD) instructions working on 2-word or 4-byte operands packed in a register, and support 2 × 8-bit multiply and 20-bit accumulate operation.
- Novel variable-length execution set (VLES) execution model that maximizes parallelism by allowing multiple address generation and data arithmetic logic units to execute 2 to 4 operations per instruction (8 to 16 operations per VLES). Can issue and execute up to six instructions per clock—for example, four independent arithmetic instructions and two pointer-related instructions (such as moves or other operations on addresses).
- Dynamic interlocking for friendlier programming, more efficient compiler support, and reduced code size.
- User and supervisor privilege levels supporting a protected software model.
- Precise memory access exceptions enables good RTOS support and soft error corrections.
- Branch target buffer (BTB) accelerates change-of-flow operations.

This section provides an overview of the key features and main modules of the SC3400 core, as well as the programming model and instruction set list.

Note: The information in this chapter is based on the *SC3400 DSP Core Reference Manual.*

At a clock speed of 800 MHz/1 GHz, the SC3400 can therefore execute 3200/4000 true DSP MIPS—3200/4000 million multiply-accumulate operations per second (MMACS), concurrent with associated data movement functions and pointer updates. The SC3400 core can sustain this high performance over time because of the flexibility of its data execution units and ability to transfer up to 128 data bits per cycle. The four data execution units can operate simultaneously in any combination. For example, the SC3400 core can execute four multiply-accumulate operations in a single clock, or one MAC, two arithmetic/logical operations and one bit field operation. All four data ALUs are identical, permitting great flexibility in assigning and executing instructions, increasing the likelihood that four execution units can be kept busy on any given cycle and enabling programs to take advantage of the SC3400 core parallel architecture.

2.1 Architecture

This section discusses the main functional blocks of the SC3400 core. **Figure 2-1** shows a block diagram of the core as used by the MSC8144E.

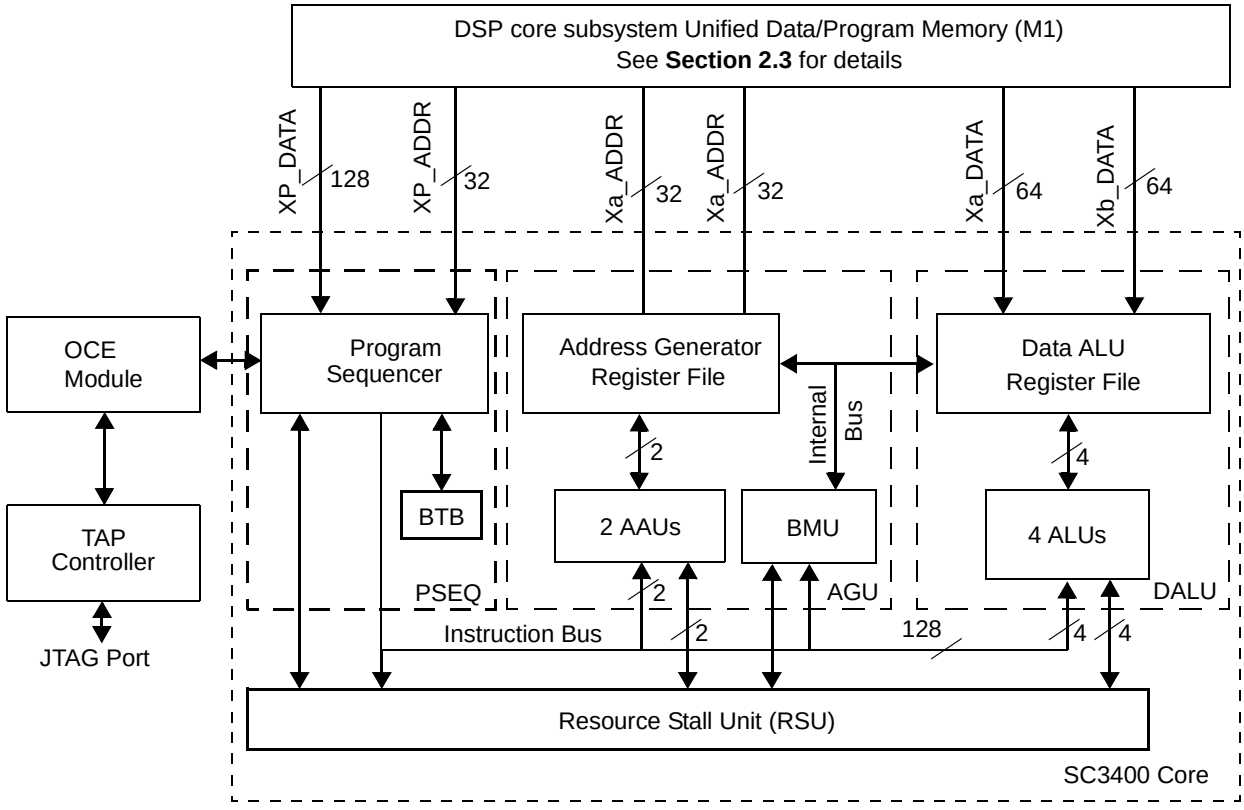


Figure 2-1. Block Diagram of the SC3400 Core in the MSC8144E

2.1.1 Data Arithmetic Logic Unit (Data ALU)

The Data ALU performs arithmetic and logical operations on data operands in the MSC8144E. The data registers can be read or written to memory over the Xa data bus and the Xb data bus as 8-bit, 16-bit, or 32-bit operands. The 64-bit wide data buses Xa and Xb data bus support the transfer of several operands on a single access. The source operands for the Data ALU, which may be 16, 32, or 40 bits, originate either from data registers or from immediate data. The results of all Data ALU operations are stored in the data registers. All Data ALU operations are performed in one clock cycle. Up to four parallel arithmetic operations can be performed in each cycle. The destination of every arithmetic operation can be used as a source operand for the operation immediately following, without any time penalty.

The components of the Data ALU are as follows:

- A bank of sixteen 40-bit registers
- Four parallel ALUs, each ALU containing a MAC unit and a BFU with a 40-bit barrel shifter
- Eight data bus shifter/limiter circuits, to allow limiting four 16-bit fractional words over each of the 64-bit data buses in a single cycle.

All the MAC units and BFUs can access all the Data ALU registers. Each register is partitioned into three portions: two 16-bit registers (low and high portion of the register) and one 8-bit register (extension portion). The 16-bit high and low register portions are typically used as an inputs for arithmetic operations. The full 40-bit register can be used as an input operand, but is generally used as an output operand for most instructions. The two 64-bit wide data buses that connect between the Data ALU register file and the memory enable a very high data bandwidth between memory and registers. Load and store instructions utilize the maximum width of the bus according to the application requirement because there are different versions of the instructions for different bandwidths:

- **move.b** loads or stores bytes (8-bit)
- **move.4b** loads or stores four bytes (32-bit)
- **move.w** or **move.f** loads or stores integer or fractional words (16-bit)
- **move.l** loads or stores long words (32-bit)
- **move.2w or move.2f** loads or stores double-integers and double-fractions, respectively (32-bit)
- **move.4w** or **move.4f** loads or stores quad-integers and quad-fractions respectively (64-bit)
- **move.2l** loads or stores two long words (64-bits total)

With the ability to execute any two **MOVE** instructions in parallel every clock cycle, a maximum data throughput of 12.8 GBps/16.0 GBps (at 800/1000 MHz) can be achieved between the memory and the register file.

2.1.1.1 Data Registers

The Data ALU registers are read or written over the data buses (Xa and Xb). The source operands for Data ALU arithmetic instructions always originate from Data ALU registers. All the Data ALU operations are performed in one clock cycle so that a new instruction can be initiated in every clock, yielding a rate of up to four Data ALU instructions per clock cycle. The destination of every arithmetic operation can be used as a source operand for the operation immediately following.

2.1.1.2 Multiply-Accumulate (MAC) Unit

The MAC unit comprises the main arithmetic processing unit of each SC3400 core and performs all the calculations on data operands. The MAC unit outputs one 40-bit result in the form of [Extension:Most Significant Portion:Least Significant Portion] (EXT:MSP:LSP). The multiplier executes 16-bit × 16-bit fractional or integer multiplication between two's complement signed, unsigned, or mixed operands. The 32-bit product is right-justified and added to the 40-bit contents of one of the sixteen data registers.

2.1.1.3 Bit-Field Unit (BFU)

The BFU contains a 40-bit parallel bidirectional shifter with a 40-bit input and a 40-bit output, mask generation unit, and logic unit. The BFU is used in the following operations:

- Multi-bit left/right shift (arithmetic or logical)
- One-bit rotate (right or left)
- Bit-field insert and extract
- Count leading bits
- Logical operations
- Sign or zero extension operations

2.1.1.4 Back Trace Registers (BTR)

The Back Trace Registers (BTR), BTR0 and BTR1, optimize the implementation of Viterbi encoder and decoder algorithms.

2.1.2 Address Generation Unit (AGU)

The AGU is one of the execution units in the SC3400 core. The AGU performs effective address calculations using the integer arithmetic necessary to address data operands in memory, and it contains the registers to generate the addresses. It performs four types of arithmetic: linear, modulo, multiple wrap-around modulo, and reverse-carry. The AGU operates in parallel with other chip resources to minimize address generation overhead. The AGU also generates change-of-flow program addresses and manages the stack pointer (SP). The major components of the AGU are as follows:

- Eight address registers (R[0–7])
- Eight alternative address registers (R[8–15]) or eight base address registers (B[0–7])
- Two stack pointers (NSP, ESP), only one of which is active at a time (SP)
- Four offset registers (N[0–3])
- Four modifier registers (M[0–3])
- A Modifier Control Register (MCTL)
- Two Address Arithmetic Units (AAU)
- One Bit Mask Unit (BMU)

Figure 2-2 shows a block diagram of the AGU.

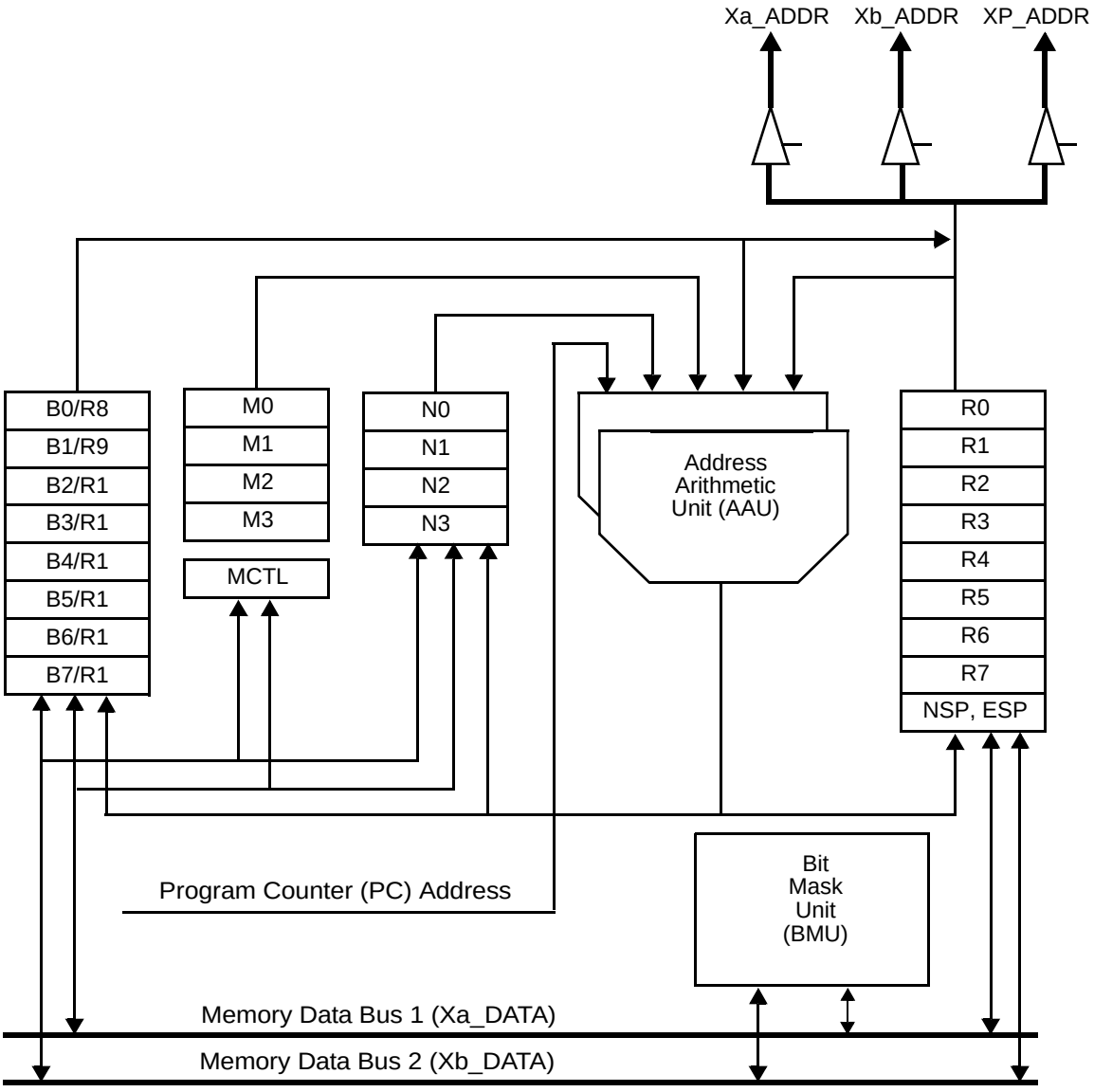


Figure 2-2. AGU Block Diagram

The two AAUs are identical. Each contains a 32-bit full adder called an offset adder and a 32-bit full adder called a modulo adder. The offset adder performs the following operations:

- Add or subtract an AGU registers or PC to/from an AGU register
- Add or subtract an immediate value to/from an AGU register
- Compare to or test an AGU register
- Logical and arithmetic shift operations on AGU registers
- Sign or zero-extend an AGU register
- Add with reverse carry

The offset values added in this adder are pre-shifted by 1, 2, or 3, according to the access width. In reverse-carry mode, the carry propagates in the opposite direction. The modulo adder adds the summed result of the first full adder to a modulo value, M or minus M, where M is stored in the selected modifier register. In modulo mode, the modulo comparator tests whether the result is inside the buffer by comparing the results to the B register and chooses the correct result from between the offset adder and the modulo adder.

2.1.2.1 Stack Pointer Registers

To facilitate use of a software stack, two special registers with special addressing modes are assigned to the AGU: the Normal Mode Stack Pointer (NSP) and the Exception Mode Stack Pointer (ESP). Both the ESP and the NSP are 32-bit read/write address registers with predecrement and post-increment updates, as well as offset with immediate values to allow random access to the software stack. Stack instructions use the ESP when the MSC8144E is in the Exception mode of operation, which it enters when exceptions occur. The NSP is used in Normal mode, while not servicing an exception. The two stack pointers make it easier to support multitasking systems and optimizes stack usage for these systems.

2.1.2.2 Bit Mask Unit (BMU)

The BMU performs bit mask operations, such as setting, clearing, changing, or testing a destination, according to an immediate mask operand. Data is loaded to the BMU over the data memory buses Xa or Xb. The result is written back over the Xa data bus or Xb data bus to the destinations in the next cycle. All bit mask instructions typically execute in two cycles and work on 16-bit data. This data can be a memory location, or a portion (high or low) of a register. The BMU supports a set of bit mask instructions that operate on:

- All AGU pointers (R[0–15])
- All Data ALU registers (D[0–15])
- All control registers (EMR, VBA, SR, MCTL)
- Memory locations

Only a single bit mask instruction is allowed in any single execution set, since only one execution unit exists for these instructions. A subset of the bit mask instructions (BMTSET) allows support for software semaphores.

2.1.3 Program Sequencer Unit (PSEQ)

The PSEQ fetches and dispatches instructions, controls hardware loops, and controls exception processing. The PSEQ implements three out of the twelve stages of the pipeline and controls the different processing states of the MSC8144E core. It consists of three hardware blocks:

- *Program address generator (PAG)*. Generates the program counter (PC) for instruction fetch operations and controls the hardware loop functionality.
- *Program dispatch unit (PDU)*. Detects the execution set out of the fetch set and dispatches the various instructions of the execution set to their appropriate execution units.
- *Program control unit (PCU)*. Controls the overall pipeline behavior of the program flow.
- *Branch target buffer (BTB)*. Reduces change-of-flow (COF) cycle latency by predicting COF resolution based on a dynamic history of previous executions of the same COF.

The PSEQ implements its functions using the following registers:

- Program Counter Register (PC)
- Status Register (SR)
- Four Loop Start Address Registers (SA[0–3])
- Four Loop Counter Registers (LC[0–3])
- Exception and Mode Register (EMR)
- Vector Base Address Register (VBA)

2.1.4 Resource Stall Unit (RSU)

The RSU block is the hardware interlock controller. It collects information from the instruction bus, holds the status for all resources in the core and resolves conflicts and hazards in the pipeline. The SC3400 RSU covers all new hazards that are a result of the deeper pipeline and maintains backward compatibility with SC1000-family core legacy code.

2.1.5 On-Chip Emulator (OCE)

The OCE module allows nonintrusive interaction with the MSC8144E and its peripherals so that you can examine registers, memory, or on-chip peripherals, define various breakpoints, and read the trace-FIFO. These interactions facilitate hardware and software development on the MSC8144E processor. The OCE module interfaces with the debugging system through on-chip JTAG TAP controller signals.

2.2 Programming Model

The three main units of the SC3400 DSP core programming model are the Address Generation Unit (AGU), the Data Arithmetic Logic Unit (Data ALU), and the PSEQ (see **Figure 2-3**). This section gives a brief overview of each of these units.

2.2.1 AGU Programming Model

The address registers can be programmed for linear, modulo (regular or multiple wrap-around), and bit-reverse addressing. Automatic updating of address registers is available when address register indirect addressing is used.

- *Address Registers (R[0–15])*. The sixteen 32-bit address registers R[0–15] contain addresses or general-purpose data. These are 32-bit read/write registers. The 32-bit address in a selected address register is used in calculating the effective address of an operand. The contents of an address register point directly to memory or are used as an offset. R[0–15] are composed of two separate banks, a lower bank (R[0–7]) and an upper bank (R[8–15]). The lower bank registers can be used for linear, modulo, or bit reverse addressing. An upper bank register can be used in linear addressing modes only if the respective register in the lower bank is not using modulo addressing mode. In modulo addressing mode, each lower bank register Rn is assigned a corresponding base address register Bn. Registers B[0–7] and R[8–15] are mapped to the same physical register, respectively. Therefore, for example, R8 is available only if R0 is not being used in modulo addressing, since this requires the base address register B0. See **Section 2.2.2**, *Data Arithmetic Logic Programming Model*, on page 2-11 for further information.

  If an address register is updated, one of the modifier control registers (MCTL) specifies the type of update arithmetic. Offset registers (Ni) are used for post-addition and indexing by offset. The address register modification is performed by either of the two AAUs.
- *Stack Pointer Registers (NSP, ESP)*. The MSC8144E has two stack pointer registers: the Normal Stack Pointer (NSP) and the Exception Stack Pointer (ESP). These 32-bit registers are used implicitly in all PUSH and POP instructions. Only one stack pointer is active at a time, according to the mode:
  - In Normal mode, the NSP is used.
  - In Exception mode, the ESP is used.

  The Status Register EXP bit determines the active mode. The active stack pointer (SP) is used explicitly for memory references in the address register indirect modes. The stack pointers point to the next unoccupied location in the stacks. They are post-incremented on all the implicit PUSH operations and pre-decremented on all the implicit POP operations.

Note: You must explicitly initialize both stack pointer registers after reset.

- *Offset Registers (N[0–3])*. The 32-bit read/write offset registers N[0–3] contain offset values to increment or decrement address registers in address register update calculations. These registers are also used for 32-bit general-purpose storage. For example, the contents of an offset register specify the offset into a table or the base of the table for indexed addressing. An offset register can be used to step through a table at a specified rate—such as five locations per step for waveform generation. Each address register can be used with each offset register. For example, R0 can be used with N0, N1, N2, or N3 for offset address calculation.
- *Base Address Registers (B[0–7])*. The 32-bit read/write base address registers B[0–7] are used in modulo calculations. Each B register is associated with an R register (B0 with R0, and so on). When the modulo addressing mode is activated, the B register contains the lower boundary value of the modulo buffer. The upper boundary of the modulo buffer is calculated by B+M-1, where M is the modifier register associated with the register used. When not used for modulo accessing, these registers can function as alternative address registers (R[8–15]). Both Rx and B_{X-8} share the same physical register. For example, if R0 is not programmed for modulo addressing, the base address register B0 can serve as an additional address register R8.
- *Modifier Registers (M[0–3])*. The 32-bit read/write modifier registers M[0–3] contain the value of the modulus modifier. These registers are also used for general-purpose storage. The address arithmetic unit (AAU) supports linear, modulo, multiple wrap-around modulo, and reverse-carry arithmetic types for most address register indirect addressing modes. When the modulo arithmetic is activated, the contents of Mj specify the modulus. Each address register can be used with each modifier register, as programmed in the MCTL register.
- *Modifier Control Register (MCTL)*. The 32-bit read/write register to program the address mode (AM) for each of the eight address registers (R[0–7]). The addressing mode of the upper address register file (R[8–15]) cannot be programmed and functions in linear mode only.

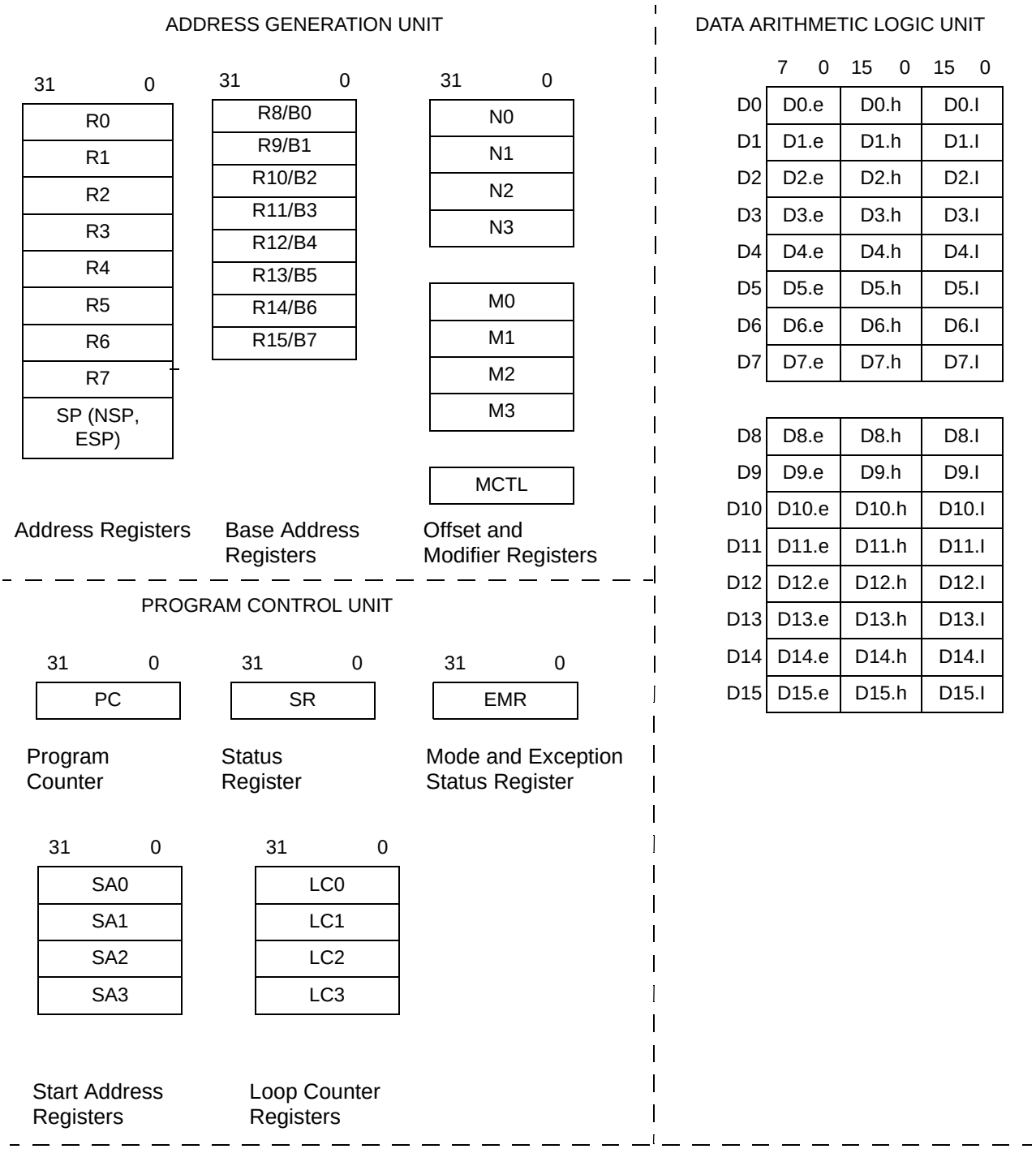


Figure 2-3. SC3400 Programming Model

2.2.2 Data Arithmetic Logic Programming Model

The Data ALU programming model is shown in **Figure 2-3**. Register D0 refers to the entire 40-bit register, whereas D0.e, D0.h, D0.l refer to the extension, most significant and least significant portions of the D0 register, respectively. The D[0–15] data registers, referred to as D*x*, give maximum flexibility, since they are used as source operands, destination storage, or accumulators.The registers serve as input buffer registers between the Xa data bus or Xb data bus and the ALUs. They are used as Data ALU source operands, allowing new operands to be loaded for the next instruction while the register contents are used by the current arithmetic instruction.

Each data register Dx has an additional associated flag bit, the limit tag bit Lx, to signify that limiting could occur when reading Dx over the Xa data bus and Xb data bus. For saving and restoring, the limit tag bit Lx is coupled with the extension portion Dx.e, to form a 9-bit operand. The limit tag bit Lx is updated when a result is written from the ALU to the Dx register.

The data registers are accessed with three types of data width:

- A long-word type access, writing or reading 32-bit operands
- A word type access, writing or reading 16-bit operands
- A byte type access, writing or reading 8-bit operands

Fractional data in Dx registers that is transferred to memory over the Xa data bus and Xb data bus is replaced by a limiting constant if the value cannot be represented by the number of bits in the access width. The contents of Dx are not affected if limiting occurs. Only the value transferred over Xa data bus or Xb data bus is limited. This process is commonly referred to as transfer saturation, and it should not be confused with the arithmetic saturation mode. The overflow protection is performed after the contents of the register are shifted according to the scaling mode. Shifting and limiting are performed only when a fractional operand is specified as the source for a data move over Xa data bus or Xb data bus. When an integer operand is specified as the source for a data move, shifting and limiting are not performed.

Automatic sign extension or zero extension of the data values into the 40-bit registers is provided when an operand is transferred from memory to a data register. If a fractional word operand is to be written to a data register, the MSP portion of the register is written with the word operand, the LSP portion is zero-extended, and the EXT portion is sign-extended from MSP. When an integer operand is to be written to a data register, the LSP portion of the register is written with the word operand, and the MSP portion and EXT are either zero-extended or sign-extended from the LSP. Long-word operands are written into the MSP:LSP portions of the register, and the EXT portion is either zero- or sign-extended.

When a byte operand is to be written to a data register, the register’s first eight bit portion of the LSP (Dx.1[7–0]) is written with the byte operand, and the remaining bits are either zero-extended or sign-extended from the LSP lower byte.

2.2.3 Program Control Unit Programming Model

The Program Control Unit (PCU) is part of the Program Sequencer Unit (PSEQ). The PCU controls the overall pipeline behavior of the program flow. The PCU implements its functions using the following registers:

- Program Counter Register (PC)
- Status Register (SR)
- Four Start Address Registers (SA[0–3])
- Four Loop Counter Registers (LC[0–3])
- Exception and Mode Register (EMR). The EMR reflects and controls exception situations in the core. It contains bits that reflect memory configuration, servicing of a non-maskable interrupt, and the following exception conditions: Data ALU overflow, illegal execution set, and illegal instruction flow.

The EMR GP[0–6] and BEM fields are initialized at reset as described in **Table 2-1**.

Table 2-1. EMR GP[6–0] and BEM Field Reset Values

Field	Reset Value[1]
BEM[2]	1
GP0	EE1
GP1	0
GP2	ISBSEL2 from Hard Reset Configuration Word (HRCW) bit 15
GP3	ISBSEL1 from HRCW bit 14
GP4	ISBSEL0 from HRCW bit 13[3]
GP5	0
GP6	0

Notes:
1. All of the specified bits are read-only and cannot be changed by the user except as indicated above by the input signal or HCRW bit names.
2. The device design selects big-endian mode for core operation, as indicated by BEM = 1.
3. GP4 equals the inversion of the HRCW bit 13.

2.3 Instruction Set Overview

The SC3000 instruction set is organized into the following instruction types:

- *DALU Instructions*. Perform operations on the data registers D[0–15] using the DALU execution units (MAC and BFU). The DALU instructions are further divided into the following subtypes:
  - Data arithmetic instructions are listed in **Table 2-2**.
  - Multiply instructions are listed in **Table 2-3**.
  - Multiply accumulate instructions are listed in **Table 2-4**.
  - Logical (including bit-field) instructions are listed in **Table 2-5**.
  - Barrel-shifter instructions are listed in **Table 2-6**.
- *AGU Instructions*. Perform operations using the AGU execution units (AAU and BMU) and the program sequencer unit. The AGU instructions are further divided into the following subtypes:
  - Address arithmetic instructions (AAU) are listed in **Table 2-7**.
  - Move instructions are listed in **Table 2-8**.
  - Stack support instructions are listed in **Table 2-9**.
  - Cache instructions are listed in **Table 2-10**.
  - Bit-mask (BMU) instructions are listed in **Table 2-11**.
  - Non-loop change-of-flow (non-loop COF) instructions are listed in **Table 2-12**.
  - Loop control and loop change-of-flow instructions are listed in **Table 2-13**.
  - Program control instructions are listed in **Table 2-14**.
- *PREFIX Instructions*. Support conditional execution of other instructions and NOP insertion for time and space padding. They have unique properties since their binary form is a prefix encoding. They are decoded by the dispatcher, but are not dispatched to an execution unit. All PREFIX instructions are listed in **Table 2-15**.

Table 2-2. DALU Arithmetic Instructions (DAU)

Instruction	Description
ACS2H	Add Compare Select High Part
ACS2L	Add Compare Select Low Part
ABS	Absolute value
ABS2	Two word absolute values
ADC	Add long with carry
ADD	Add
ADD.W	Add 16-Bit Value
ADD2	Add two words
ADDNC.W	Add without changing the carry bit in the SR
ADR	Add and round
ASL	Arithmetic shift left by one bit

Table 2-2. DALU Arithmetic Instructions (DAU)

Instruction	Description
ASL2	Arithmetic Shift Left by One of Two Word Operands
ASR	Arithmetic shift right by one bit
ASR2	Arithmetic shift right by two bits
AVGU4	Average of Two Byte Operands
CLIP	Clip
CLIP2	Clip two operands
CLR	Clear
CMPEQ	Compare for equal
CMPEQ.L	Integer compare for equal
CMPGT	Compare for greater than
CMPHI	Compare for higher (unsigned)
DECEQ	Decrement a data register and set T if zero
DECGE	Decrement a data register and set T if greater than or equal to zero
DIV	Divide iteration
IADD	Integer addition not affected by saturation
IADDNC.W	Integer addition without changing the carry bit. Not affected by saturation
INC	Increment a data register (as integer data)
INC.F	Increment a data register (as fractional data)
ISUB	Integer Subtraction not affected by saturation
MAX	Transfer maximum signed value
MAX2	Transfer two 16-bit maximum signed values
MAX2VIT	Special MAX2 version for Viterbi kernel
MAXM	Transfer maximum magnitude value
MIN	Transfer minimum signed value
MIN2	Transfer two 16-bit minimum signed values
NEG	Negate
NEG2	Two negate word operations
PACK.2x	Pack Two Integer/Fractional Words to a Register
RND	Round
SAD4	Sum of Absolute Byte Differences
SAT.F	Saturate fractional value in data register to fit in high portion
SAT.L	Saturate value in data register to fit in 32 bits
SATU2.B	Saturate two signed words to fit in unsigned bytes
SAT2.W	Saturate two 20-bit words to fit 16-bit words
SBC	Subtract long with carry
SBR	Subtract and round
SOD2ffcc	Sum Or Difference of Two 16-Bit Values, function & cross

Table 2-2. DALU Arithmetic Instructions (DAU)

Instruction	Description
SUB	Subtract
SUB.W	Subtract 16-Bit Value
SUB2	Subtract two words
SUBL	Shift left and subtract
SUBNC.W	Subtract without changing the carry bit in the status register
TFR	Transfer data register to a data register
TFR.W	Transfer Half Data Register to Half Data Register
TFRF	Conditional data register transfer, if the T bit is clear
TFRT	Conditional data register transfer, if the T bit is set
TSTEQ	Test for equal to zero
TSTEQ.L	Integer test for equal to zero
TSTGE	Test for greater than or equal to zero
TSTGT	Test for greater than zero
VTRACE	Viterbi Trace Back

Table 2-3. DALU Multiply Instructions (MPY)

Instruction	Description
IMPY	Multiply signed integers in data registers
IMPY.W	Multiply signed immediate and signed integer in data register
IMPYHLUU	Multiply unsigned integer and unsigned integer; first source from high portion, second from low portion
IMPYSU	Multiply signed integer and unsigned integer
IMPYSU2	Two multiply of signed by unsigned integer bytes
IMPYUU	Multiply unsigned integer and unsigned integer
MPY	Multiply signed fractions
MPYR	Multiply signed fractions and round
MPYSU	Multiply signed fraction and unsigned fraction
MPYUS	Multiply unsigned fraction and signed fraction
MPYUU	Multiply unsigned fraction and unsigned fraction

Table 2-4. DALU Multiply-Accumulate Instructions (MAC)

Instruction	Description
DMACSS	Multiply signed by signed and accumulate with data register right shifted by word size
DMACSU	Multiply signed by unsigned and accumulate with data register right shifted by word size
IMAC	Multiply-accumulate integers
IMACLHUU	Multiply-accumulate unsigned integers; first source from low portion, second from high portion
IMACSU2	Two multiply-accumulate of signed by unsigned integer bytes

Table 2-4. DALU Multiply-Accumulate Instructions (MAC)

Instruction	Description
IMACUS	Multiply-accumulate unsigned integer and signed integer
MAC	Multiply-accumulate signed fractions
MACR	Multiply-accumulate signed fractions and round
MACSU	Multiply-accumulate signed fraction and unsigned fraction
MACUS	Multiply-accumulate unsigned fraction and signed fraction
MACUU	Multiply-accumulate unsigned fraction and unsigned fraction

Table 2-5. DALU Logical Instructions (DLU)

Instruction	Description
AND	Logical AND
BDEINTRLV	Bit De-Interleave
BREV	Bit Reverse
BINTRLV	Bit Interleave
CLB	Count leading bits (ones or zeros)
DOALIGN	Extracts unaligned four bytes
EOR	Logical exclusive OR
EXTRACT	Extract signed bit field
EXTRACTU	Extract unsigned bit field
INSERT	Insert bit field
NOT	One’s complement (inversion)
OR	Logical inclusive OR
ROL	Rotate one bit left through the carry bit
ROR	Rotate one bit right through the carry bit
SWAP	Swap
SWAPB	Swap the four bytes in a long (for BE8/LE8 support)
SWAPB2	Swap the bytes in each word (for BE8/LE8 support)
SXT.B	Sign extend byte
SXT.L	Sign extend long
SXT.W	Sign extend word
ZXT.B	Zero extend byte
ZXT.L	Zero extend long
ZXT.W	Zero extend word

Table 2-6. DALU Barrel Shifter Instructions (DBS)

Instruction	Description
ASLL	Multiple-bit arithmetic shift left
ASLL2	Multiple-bit arithmetic shift left of two word operands
ASLLS	Multiple-bit arithmetic shift left with saturation
ASLW	Word arithmetic shift left (16 bit shift)
ASRR	Multiple-bit arithmetic shift right
ASRR2	Multiple-bit arithmetic shift right of two word operands
ASRRS	Multiple-bit arithmetic shift right with saturation
ASRW	Word arithmetic shift right (16 bit shift)
LSLL	Multiple-bit logical shift left
LSLL2	Multiple-bit bitwise shift left of two word operands
LSR	Logical shift right by one bit
LSR2	Bitwise shift right one bit of two word operands
LSRR	Multiple-bit logical shift right
LSRR2	Multiple-bit bitwise shift right of two word operands
LSRR.L	Integer multiple-bit shift right
LSRW	Word logical shift right (16-bit shift)

Table 2-7. AGU Arithmetic Instructions (AAU)

Instruction	Description
ADDA	Add (affected by the modifier mode)
ADDL1A	Add with 1-bit left shift of source operand (affected by the modifier mode)
ADDL2A	Add with 2-bit left shift of source operand (affected by the modifier mode)
ASL2A	Arithmetic shift left by 2 bits (32-bit)
ASLA	Arithmetic shift left (32-bit)
ASRA	Arithmetic shift right (32-bit)
CMPEQA	Compare for equal
CMPEQA.W	Compare for Equal
CMPGTA	Compare for greater than
CMPGTA.W	Compare for Greater Than
CMPHIA	Compare for higher (unsigned)
DECA	Decrement register
DECEQA	Decrement and set T if zero
DECGEA	Decrement and set T if equal or greater than zero
INCA	Increment register
LSRA	Logical shift right (32-bit)
SUBA	Subtract (affected by the modifier mode)

Table 2-7. AGU Arithmetic Instructions (AAU)

Instruction	Description
SXTA.B	Sign extend byte
SXTA.W	Sign extend word
TFRA	Register transfer
TFRA (OSP)	Move the “other” stack pointer to/from a register, inversely defined by the exception mode
TSTEQA.L	Test for equal
TSTEQA.W	Test for equal on lower 16 bits
TSTGEA	Test for greater than or equal
TSTGTA	Test for greater than
ZXTA.B	Zero extend byte
ZXTA.W	Zero extend word

Table 2-8. AGU Move Instructions (MOVE)

Instruction	Description
MOVE.2F	Move two fractional words from memory to a register pair
MOVE.2L	Move two longs to/from a register pair
MOVE.2L	Move register pair to a register pair
MOVE.2W	Move two integer words to/from memory and a register pair
MOVE.4B	Move four signed bytes to/from memory and register quad
MOVE.4F	Move four fractional words from memory to a register quad
MOVE.4W	Move four integer words to/from memory and a register quad
MOVE.B	Move byte to/from memory
MOVE.F	Move fractional word to/from memory
MOVE.L	Move long to/from memory
MOVE.W	Move integer word to/from memory, or immediate to register or memory
MOVE.x	Move BTR portions to memory
MOVE2.2B	Move two signed bytes to/from memory to/from a packed register
MOVE2.4B	Move four signed bytes to/from memory to/from a packed register pair
MOVE2.8B	Move eight signed bytes to/from memory to/from a packed register quad
MOVEU2.2B	Move two unsigned bytes from memory to a packed register
MOVEU2.4B	Move four unsigned bytes from memory to a packed register pair
MOVEU2.8B	Move eight unsigned bytes from memory to a packed register quad
MOVEc	Move address register to address register, depending on T bit of SR
MOVES.2F	Move two fractional words to memory with scaling and limiting enabled
MOVES.4F	Move four fractional words to memory with scaling and limiting enabled
MOVES.F	Move fractional word to memory with scaling and limiting enabled
MOVES.L	Move long to memory with scaling and limiting enabled

Table 2-8. AGU Move Instructions (MOVE)

Instruction	Description
MOVEU.4B	Move four unsigned bytes from memory to a register quad
MOVEU.B	Move unsigned byte from memory
MOVEU.L	Move unsigned long from immediate
MOVEU.W	Move unsigned integer word from memory or from immediate
MOVEU.W	Move unsigned integer word from memory to a data register portion
SETALIGN	Set align shift size: special move for the DOALIGN instruction
VSL.2F	Viterbi shift left: special move for Viterbi kernel
VSL.2W	Viterbi shift left: special move for Viterbi kernel
VSL.4F	Viterbi shift left: special move for Viterbi kernel
VSL.4W	Viterbi shift left: special move for Viterbi kernel
UNPACK.2W	Unpack two integer words from a register to a register pair

Table 2-9. AGU Stack Support Instructions (STK)

Instruction	Description
POP	Pop a register from the software stack
POP.2L	Pop two long register pairs from the software stack
POPN	Pop a register from the software stack using the normal stack pointer
POPN.2L	Pop two long register pairs from the software stack using the normal stack pointer
PUSH	Push a register onto the software stack
PUSH.2L	Push two long register pairs onto the software stack
PUSHN	Push a register onto the software stack using the normal stack pointer
PUSHN.2L	Push two long register pairs onto the software stack using the normal stack pointer

Table 2-10. AGU Cache Instructions (ACH)

Instruction	Description
DFETCH	Data cache. Pre-fetch one line (no action if a “hit”, no freeze if a “miss”).
PFETCH	Program cache. Pre-fetch one line (no action if a “hit”, no freeze if a “miss”).

Table 2-11. AGU Bit-Mask Instructions (BMU)

Instruction	Description
AND	Logical AND on a 16-bit operand
BMCHG	Bit-mask change a 16-bit operand
BMCHG.W	Bit-mask change a 16-bit operand in memory
BMCLR	Bit-mask clear a 16-bit operand
BMCLR.W	Bit-mask clear a 16-bit operand in memory

Table 2-11. AGU Bit-Mask Instructions (BMU)

Instruction	Description
BMSET	Bit-mask set a 16-bit operand
BMSET.W	Bit-mask set a 16-bit operand in memory
BMTSET	Bit mask test and set a 16-bit operand
BMTSET.W	Bit mask test and set a 16-bit operand **Note:** When used as part of a reservation atomic operation, this can only be directed toward the M2 memory. It is not supported in any other memory.
BMTSTC	Bit mask test if clear Sets the T-bit, if every bit position that has the value 1 in the mask is 0 in an operand.
BMTSTC.W	Bit mask test if clear in memory. Sets the T-bit, if every bit position that has the value 1 in the mask is 0 in an operand.
BMTSTS	Bit mask test if set. Sets the T-bit, if every bit position that has the value 1 in the mask is 1 in an operand.
BMTSTS.W	Bit mask test if set in memory. Sets the T-bit, if every bit position that has the value 1 in the mask is 1 in an operand.
EOR	Logical exclusive OR on a 16-bit operand
NOT	Binary inversion of a 16-bit operand
OR	Logical OR on a 16-bit operand

Table 2-12. AGU Non-Loop Change-of-Flow Instructions (COF)

Instruction	Description
BF	Branch if false
BFD	Branch if false (delayed)
BRA	Branch
BRAD	Branch (delayed)
BSR	Branch to subroutine
BSRD	Branch to subroutine (delayed)
BT	Branch if true
BTD	Branch if true (delayed)
JF	Jump if false
JFD	Jump if false (delayed)
JMP	Jump
JMPD	Jump (delayed)
JSR	Jump to subroutine
JSRD	Jump to subroutine (delayed)
JT	Jump if true
JTD	Jump if true (delayed)
RTE	Return from exception
RTED	Return from exception (delayed)

Table 2-12. AGU Non-Loop Change-of-Flow Instructions (COF)

Instruction	Description
RTEKRI	Return from exception and keep NMID with RAS invalidated
RTEKRID	Return from exception and keep NMID with RAS invalidated (delayed)
RTERI	Return from exception with RAS invalidated
RTERID	Return from exception with RAS invalidated (delayed)
RTPE	Return from precise exception
RTS	Return from subroutine
RTSD	Return from subroutine (delayed)
RTSTK	Force restore PC from the stack, updating SP
RTSTKD	Force restore PC from the stack, updating SP (delayed)
TRAPn	Execute a precise software exception

Table 2-13. Loop Control (Including Loop COF) Instructions (ALC)

Instruction	Description
BREAK	Terminate the loop and branch to an address
CONT	Jump to the start of the loop to start the next iteration
CONTD	Jump to the start of the loop to start the next iteration (delayed)
DOENn	Do enable - set loop counter n and enable loop n as a long loop
DOENSHn	Do enable short - set loop counter n and enable loop n as a short loop
DOSETUPn	Setup loop start address n
SKIPLS	Test the active LC and skip the loop if LCn is equal or smaller than zero

Table 2-14. AGU Program Control Instructions (APC)

Instruction	Description
DEBUG	Enter debug mode
DEBUGEV	Signal debug event
DI	Disable interrupts (sets the DI bit in the status register)
EI	Enable interrupts (clears the DI bit in the status register)
FBTB	Flush branch target buffer
ILLEGAL	Trigger a precise illegal instruction exception
MARK	Push the PC into the trace buffer
STOP	Stop processing (lowest power stand-by)
SYNCIO	Synchronize I/O
SYNCM	Synchronize Memory
WAIT	Wait for interrupt (low power stand-by)

Table 2-15. Prefix Instructions (PRE)

Instruction	Description
IFA	Execute current execution set or subgroup unconditionally
IFF	Execute current execution set or subgroup if the T bit is clear
IFT	Execute current execution set or subgroup if the T bit is set
NOP	No operation

2.4 Additional Programming Considerations

- All code sections must end with an unconditional **cof** instruction. This can be either a **cof** or a delayed **cof** instruction.

External Signals 3

The MSC8144E 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. MSC8144E Functional Signal Groupings

Functional Group	Detailed Description
Power and ground	**Table 3-3** on **page 3-5**
Clock	**Table 3-4** on **page 3-6**
Reset and Configuration	**Table 3-5** on **page 3-7**
DDR Memory Controller	**Table 3-6** on **page 3-11**
Serial RapidIO Interface	**Table 3-7** on **page 3-12**
PCI	**Table 3-8** on **page 3-14**
Ethernet Controllers	**Table 3-9** on **page 3-27**
ATM/UTOPIA	**Table 3-10** on **page 3-32**
Serial peripheral interface (SPI)	**Table 3-11** on **page 3-41**
GPIOs and maskable Interrupts	**Table 3-12** on **page 3-42**
Timers	**Table 3-13** on **page 3-48**
UART	**Table 3-14** on **page 3-49**
I^2C	**Table 3-15** on **page 3-50**
TDM[7–0]	**Table 3-16** on **page 3-50**
$\overline{\text{NMI}}$/$\overline{\text{INT_OUT}}$/$\overline{\text{NMI_OUT}}$	**Table 3-17** on **page 3-58**
OCE module and JTAG Test Access Port	**Table 3-18** on **page 3-58**

Power and ground, clocks, reset input lines, DDR memory controller, Serial RapidIO, I^2C, and four of the TDM interfaces lines are always supported. Some signals are only sampled during the power-on reset sequence; most of these signals are multiplexed with other signals. Signals to support the remaining modules are multiplexed across the remaining signal lines.

Signal multiplexing occurs at two levels. The high-level I/O multiplexing is selected during power-on reset by the PIN_MUX bits in the high part of the Reset Configuration Word (see **Table 5-7** on page 5-18 for details). The high-level I/O multiplexing permits sharing of signal lines among the PCI interface, the two Ethernet controllers, the ATM/UTOPIA interface, and the GPIO ports. **Table 3-2** lists the signal groups supported by each of the eight available multiplexing modes.

Table 3-2. Interface Multiplexing by Mode

Interface	Supported Interfaces by I/O Multiplexing Mode							
	0 (000)	**1 (001)**	**2 (010)**	**3 (011)**	**4 (100)**	**5 (101)**	**6 (110)**	**7 (111)**
TDM	TDM[0–7]	TDM[0–7]	TDM[0–6]	TDM[0–3]	TDM[0–3]	TDM[0–6]	TDM[0–6]	TDM[0–6]
PCI	—	—	PCI	PCI	PCI (no error support)	—	—	
ATM	UTOPIA8 UTOPIA16	UTOPIA8	—	UTOPIA8	UTOPIA8 UTOPIA16	UTOPIA8 UTOPIA16	UTOPIA8	UTOPIA8 UTOPIA16 POS Master mode
Ethernet 1	SGMII	MII/SMII/RMII/ RGMII/SGMII	SMII/RMII/ RGMII/SGMII	SGMII	SGMII	SGMII	MII/SMII/ RMII/ RGMII/ SGMII	SGMII
Ethernet 2	SMII/RMII/ SGMII	SMII/RMII/ SGMII	SMII/RMII/ SGMII	SMII/RMII/ SGMII	SGMII	SMII/RMII/ RGMII/ SGMII	SMII/RMII/ RGMII/ SGMII	SMII/RMII/ SGMII
Clocks, JTAG, I^2C, Reset, DDR, RapidIO, and UART	Supported in all modes							
Total GPIO	31	30	28	17	14	31	30	27

Note: The GE_MDC and GE_MDIO pins reference supply is V_{DDGE2}. These pin are used for Ethernet PHY management for both ethernet controllers. Therefore, the following limitations exist with regard to signal multiplexing:

1. When using pin multiplexing mode 6, if the application requires the RGMII port with MDC and MDIO management done by the MSC8144, then use Ethernet controller 2. If two RGMII ports are needed (for pin multiplexing mode 6), then use both Ethernet controllers.
2. When using pin multiplexing mode 1, if the application requires the RGMII port with MDC and MDIO management done by the MSC8144, then tie V_{DDGE1} and V_{DDGE2} to a 2.5 V supply; the following limitations exist:
   a. Ethernet controller 2 does not support RMII and SMII (they are 3.3 V protocols).
   b. TDM7 is not supported (only 7 TDM ports are available).
3. When using pin multiplexing mode 2, if the application requires the RGMII port with MDC and MDIO management done by the MSC8144E, then tie V_{DDGE1} and V_{DDGE2} to a 2.5 V supply; the following limitations exist:
   a. Ethernet controller 2 does not support RMII and SMII (they are 3.3 V protocols).
   b. PCI is not supported (this is a 3.3 V protocol).

   One way to overcome limitations 2 and 3 is to implement the management (MDC/MDIO) using an external host. Then, you connect the MSC8144E V_{DDGE2} to 3.3 V supply, which removes the limitations.

If the GPIO function is selected by the high-level I/O multiplexing, additional multiplexing selections can be made through the GPIO configuration registers (see **Chapter 23**, *GPIO* for details) that permit sharing of signals among the GPIO ports, maskable interrupt inputs, timers, UART, I^2C, and some TDM interfaces. **Figure 3-1** shows MSC8144E external signals organized by function.

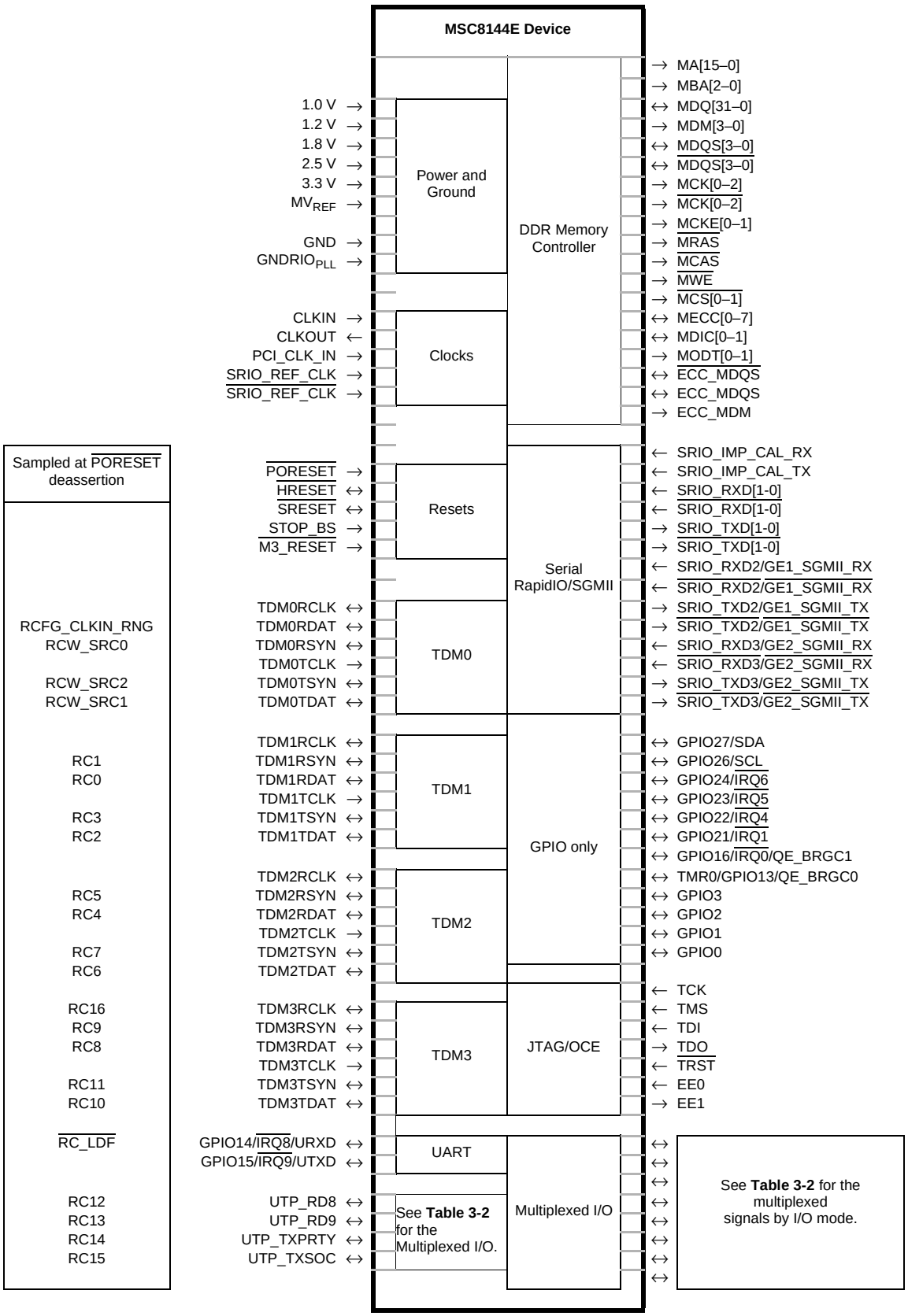


Figure 3-1. MSC8144E External Signals

3.1 Power Signals

Table 3-3. Power and Ground Inputs

Signal Name	Description
V_{DD}	**Internal Logic Power (1.0 V)** A dedicated well-regulated power source for the internal logic. Provide an extremely low impedance path to the V_{DD} power rail and adequate external decoupling capacitors. The external decoupling capacitors recommendations are listed in the *MSC8144E Technical Data Sheet*.
V_{DDDDR}	**SSTL IO Driver Power (2.5 V or 1.8 V)** A dedicated power source for the DDR DRAM interface buffers. Provide adequate external decoupling capacitors. The external decoupling capacitors recommendations are listed in the *MSC8144E Technical Data Sheet*.
MV_{REF}	**SSTL Reference Power** A reference power level for the SSTL2 memory interface.
V_{DDM3}	**M3 Internal Logic (1.2 V)** A dedicated well-regulated power source for the M3 internal logic. Provide an extremely low impedance path to the power rail. The external decoupling capacitors recommendations are listed in the *MSC8144E Technical Data Sheet*.
V_{DDM3IO}	**M3 I/O Power (2.5 V)** A dedicated well-regulated power source for the M3 I/O signals. Provide an extremely low impedance path to the power rail.
V_{25M3}	**M3 Charge Pump Power (2.5 V)** A dedicated well-regulated power source for the M3 EDRAM charge pump.
V_{DDIO}	**Input/Output Power (3.3 V)** The power source for the external non-Ethernet I/O signal lines. Provide adequate external decoupling capacitors. The external decoupling capacitors recommendations are listed in the *MSC8144E Technical Data Sheet*.
V_{DDGE1}	**Ethernet 1 Input/Output Power (2.5 V or 3.3 V)** The power source for the Ethernet signal lines. Provide adequate external decoupling capacitors. The external decoupling capacitors recommendations are listed in the *MSC8144E Technical Data Sheet*.
V_{DDGE2}	**Ethernet 2 Input/Output Power (2.5 V or 3.3 V)** The power source for the Ethernet signal lines. Provide adequate external decoupling capacitors. The external decoupling capacitors recommendations are listed in the *MSC8144E Technical Data Sheet*.
V_{DDPLL0}	**System PLL 0 Power (1.0 V)** A dedicated well-regulated power for the system Phase Lock Loops (PLLs). Provide an extremely low impedance path to the V_{DDPLL} power rail.
V_{DDPLL1}	**System PLL 1 Power (1.0 V)** A dedicated well-regulated power for the system Phase Lock Loops (PLLs). Provide an extremely low impedance path to the V_{DDPLL} power rail.
V_{DDPLL2}	**System PLL 2 Power (1.0 V)** A dedicated well-regulated power for the system Phase Lock Loops (PLLs). Provide an extremely low impedance path to the V_{DDPLL} power rail.
$V_{DDRIOPLL}$	**RapidIO PLL Power (1.0 V)** A dedicated well-regulated power for the RapidIO Phase Lock Loops (PLL). Provide an extremely low impedance path to the $V_{DDRIOPLL}$ power rail.
V_{DDSXC}	**RapidIO C Power (1.0 V)** A dedicated well-regulated power for the RapidIO C circuitry. Provide an extremely low impedance path to the V_{DDSXC} power rail.

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

Signal Name	Description
V_{DDSXP}	**RapidIO P Power (1.0 V)** A dedicated well-regulated power for the RapidIO P circuitry. Provide an extremely low impedance path to the V_{DDSXP} power rail.
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}.
GND_{RIOPLL}	**RapidIO PLL Ground** Ground dedicated for RapidIO PLL use. The connection should be provided with an extremely low-impedance path to ground.
GND_{SXC}	**RapidIO C Ground** A ground for the RapidIO C circuitry. Provide an extremely low impedance path to the ground plane.
GND_{SXP}	**RapidIO P Ground** A ground for the RapidIO P circuitry. Provide an extremely low impedance path to the ground plane.

3.2 Clock Signals

Table 3-4. Clock Signals

Signal Name	Type	Signal Description
CLKIN	Input	**Clock In** Primary clock input to the MSC8144E PLLs.
CLKOUT	Output	**Clock Out** The bus clock output.
PCI_CLK_IN	Input	**PCI Input Clock** Input clock for the PCI module.

3.3 Reset and Configuration Signals

Table 3-5. Reset and Configuration Signals

Signal Name	Type	Signal Description
PORESET	Input	**Power-On Reset** When asserted, this line causes the MSC8144E to enter power-on reset state. Internally, this signal also resets the TAP and debugging modules. The power-on reset flow resets the MSC8144E device, configures various device attributes including its clock modes, and drives HRESET and SRESET as open-drain outputs.
HRESET	Input/ Output	**Hard Reset** When asserted as an input, this signal causes the MSC8144E 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 and SRESET as open-drain outputs. This signal requires an external pull-up resistor. The signal is tri-stated after the hard reset flow is complete.
SRESET	Input/ Output	**Soft Reset** When asserted as an input, this signal causes the MSC8144E to enter the soft reset state, about all current internal transactions, configure most registers with their default values, and cause the cores to enter their reset state. The signal does not affect I/O signal functionality or direction or memory controller operations. While the device is in the soft reset state, it drives the SRESET as an open-drain output. This signal requires an external pull-up resistor. The signal is tri-stated after the soft reset flow is complete.
M3_RESET	Input	**M3 Memory Reset** When asserted, this line causes the M3 memory to enter the reset state. When using the M3 memory, connect to 2.5 V using the same logic as PORESET.
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 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*).
RC_LDF	Output	**Reset Configuration Word Load Failure** Used by the core to signal when the reset configuration word fails to load from an I^2C EEPROM when using the boot sequencer. It indicates whether the boot sequencer failed, which can be due to an incorrect data structure or an I^2C bus failure. The signal can be asserted anytime during HRESET assertion. Once asserted, the device holds HRESET low until the PORESET is restarted.
GPIO14	Input/ Output	**General-Purpose Input Output 14** One of 32 GPIO pins used as GPIO or as a dedicated input or output. For details, see **Chapter 23**, *GPIO*.
IRQ8	Input	**Interrupt Request 8** One of the sixteen external lines that can request a service routine, via the internal interrupt controller, from the SC3400 cores.
URXD	Input/ Output	**UART Receive Data** For details, see **Chapter 21**, *UART*.

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

Signal Name	Type	Signal Description
RCFG_CLKIN_RNG	Input	**Reset Configuration CLKIN Range** This signal is sampled at the deassertion of PORESET to identify the range of the CLKIN input. If this pin is pulled low, the CLKIN frequency is less than or equal to 66 MHz. If this pin is pulled high, the CLKIN frequency is above 66 MHz. The required signal level must be maintained as long as HRESET is asserted.
TDM0RDAT	Input/ Output	**TDM0 Serial Receiver Data** The receive data signal for TDM 0. As an input, this can be the DATA_A data signal for TDM 0. For configuration details, see **Chapter 20**, *TDM Interface*.
RCW_SRC0	Input	**Reset Configuration Word Source 0** Along with the RCW_SRC[1–2], this signal is sampled at the deassertion of PORESET to identify the source of the reset configuration word. The required signal level must be maintained as long as HRESET is asserted.
TDM0RSYN	Input/ Output	**TDM0 Receive Frame Sync** The receive sync signal for TDM 0. As an input, this can be the DATA_B data signal for TDM 0. For configuration details, see **Chapter 20**, *TDM Interface*.
RCW_SRC1	Input	**Reset Configuration Word Source 1** Along with the RCW_SRC[0, 2], this signal is sampled at the deassertion of PORESET to identify the source of the reset configuration word. The required signal level must be maintained as long as HRESET is asserted.
TDM0TDAT	Input/ Output	**TDM0 Serial Transmitter Data** The transmit data signal for TDM 0. As an output, this can be the DATA_D data signal for TDM 0. For configuration details, see **Chapter 20**, *TDM Interface*.
RCW_SRC2	Input	**Reset Configuration Word Source 2** Along with the RCW_SRC[0–1], this signal is sampled at the deassertion of PORESET to identify the source of the reset configuration word. The required signal level must be maintained as long as HRESET is asserted.
TDM0TSYN	Input/ Output	**TDM0 Transmit frame Sync** Transmit Frame Sync for TDM 0. For configuration details, see **Chapter 20**, *TDM Interface*.
RC0	Input	**Reset Configuration Word Bit 0** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
TDM1RDAT	Input/ Output	**TDM1 Serial Receiver Data** The receive data signal for TDM 1. As an input, this can be the DATA_A data signal for TDM 1. For configuration details, see **Chapter 20**, *TDM Interface*.
RC1	Input	**Reset Configuration Word Bit 1** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
TDM1RSYN	Input/ Output	**TDM1 Receive Frame Sync** The receive sync signal for TDM 1. As an input, this can be the DATA_B data signal for TDM 1. For configuration details, see **Chapter 20**, *TDM Interface*.
RC2	Input	**Reset Configuration Word Bit 2** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
TDM1TDAT	Input/ Output	**TDM1 Serial Transmitter Data** The transmit data signal for TDM 1. As an output, this can be the DATA_D data signal for TDM 1. For configuration details, see **Chapter 20**, *TDM Interface*.

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

Signal Name	Type	Signal Description
RC3	Input	**Reset Configuration Word Bit 3** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
TDM1TSYN	Input/ Output	**TDM1 Transmit Frame Sync** Transmit frame sync for TDM 1. For configuration details, see **Chapter 20**, *TDM Interface*.
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.
TDM2RDAT	Input/ Output	**TDM2 Serial Receiver Data** The receive data signal for TDM 2. As an input, this can be the DATA_A data signal for TDM 2. For configuration details, see **Chapter 20**, *TDM Interface*.
RC5	Input	**Reset Configuration Word Bit 5** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
TDM2RSYN	Input/ Output	**TDM2 Receive Frame Sync** The receive sync signal for TDM 2. As an input, this can be the DATA_B data signal for TDM 2. For configuration details, see **Chapter 20**, *TDM Interface*.
RC6	Input	**Reset Configuration Word Bit 6** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
TDM2TDAT	Input/ Output	**TDM2 Serial Transmitter Data** The transmit data signal for TDM 2. As an output, this can be the DATA_D data signal for TDM 2. For configuration details, see **Chapter 20**, *TDM Interface*.
RC7	Input	**Reset Configuration Word Bit 7** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
TDM2TSYN	Input/ Output	**TDM2 Transmit frame Sync** Transmit frame sync for TDM 2. For configuration details, see **Chapter 20**, *TDM Interface*.
RC8	Input	**Reset Configuration Word Bit 8** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
TDM3RDAT	Input/ Output	**TDM3 Serial Receiver Data** The receive data signal for TDM 3. As an input, this can be the DATA_A data signal for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.
RC9	Input	**Reset Configuration Word Bit 9** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
TDM3RSYN	Input/ Output	**TDM3 Receive Frame Sync** The receive sync signal for TDM 3. As an input, this can be the DATA_B data signal for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.

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

Signal Name	Type	Signal Description
RC10	Input	**Reset Configuration Word Bit 10** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
TDM3TDAT	Input/ Output	**TDM3 Serial Transmitter Data** The serial transmit data signal for TDM 3. As an output, it provides the DATA_D signal for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.
RC11	Input	**Reset Configuration Word Bit 11** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
TDM3TSYN	Input/ Output	**TDM3 Transmit frame Sync** Transmit frame sync for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.
RC12	Input	**Reset Configuration Word Bit 12** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
UTP_RD8	Input	**ATM UTOPIA Receive Data 8** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.
RC13	Input	**Reset Configuration Word Bit 13** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
UTP_RD9	Input	**ATM UTOPIA Receive Data 9** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.
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.
UTP_TPRTY	Output	**ATM UTOPIA Transmit Parity** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) 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.
UTP_TSOC	Output	**ATM UTOPIA Transmit Start of Cell** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) 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.
TDM3RCLK	Input/ Output	**TDM3 Receive Clock** The receive clock signal for TDM 3. As an output, this can be the DATA_C data signal for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.
Note: RC[0–16] are valid only for driving a reduced external reset configuration word value. The signals are sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers. The required signal levels must be maintained as long as HRESET is asserted. All other signal drivers connected to these inputs must be tri-stated while HRESET is asserted.		

3.4 Memory Controller

Refer to the memory controller chapter for details on configuring these signals. To support DDR DRAM external memory, the memory controller uses SSTL2+ signal levels.

Table 3-6. 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[31–0]	Input/ Output	**Data Bus** The MSC8144E device drives the bus during write cycles and the external memory drives the bus during read cycles.
MDM[3–0]	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 MDM3 corresponds to the LSB.
MDQS[3–0]	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.
$\overline{\text{MDQS[3–0]}}$	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.
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.
$\overline{\text{MCS[0–1]}}$	Output	**Chip Select 0–1** Enables specific memory devices or peripherals connected to the bus.
MECC[7–0]	Input/Output	**Error Checking and Correction Codes** As outputs, represent the state of ECC driven by the DDR controller on writes. As inputs, represents the ECC driven by the DDR SDRAMs on reads.

Table 3-6. Memory Controller Signals (Continued)

Signal Name	Type	Description
ECC_MDM	Output	**ECC Data Output Mask** Masks ECC 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.
ECC_MDQS	Input/Output	**ECC Data Strobe** Strobe for ECC data capture. The signal is driven as an input by the DDR SRAM with read data and driven by the DDR controller as an output with write data. The strobes may be single-ended or differential.
$\overline{\text{ECC_MDQS}}$	Input/Output	**ECC DQS Complement** Complement strobe for byte-lane data capture. The signal is driven as an input by the DDR SRAM with read data and driven by the DDR controller as an output with write data. The strobes may be single-ended or differential.
MDIC[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.

3.5 Serial RapidIO Signals

Refer to **Chapter 16**, *Serial RapidIO® Controller* for configuration information.

Table 3-7. Serial RapidIO Signals

Signal Name	Type	Description
SRIO_IMP_CAL_RX	Input	**SRIO Receiver Impedance Control Signal** Receiver impedance calibration control signal.
SRIO_IMP_CAL_TX	Input	**SRIO Transmitter Impedance Control Signal** Transmitter impedance calibration control signal.
SRIO_RXD0	Input	**SRIO Receive Data 0** Serial data input for a 1x or 4x link. Each signal is part of a differential pair.
$\overline{\text{SRIO_RXD0}}$	Input	**SRIO Receive Data 0 Inverted** Inverted serial data input for a 1x or 4x link. Each signal is part of a differential pair.
SRIO_RXD1	Input	**SRIO Receive Data 1** Serial data input for a 4x link. Each signal is part of a differential pair.
$\overline{\text{SRIO_RXD1}}$	Input	**SRIO Receive Data 1 Inverted** Inverted serial data input for a 4x link. Each signal is part of a differential pair.
SRIO_RXD2 GE1_SGMII_RX	Input Input	**SRIO Receive Data 2** Serial data input for a 4x link. Each signal is part of a differential pair. **Ethernet 1 SGMII Receive Data** Part of the Ethernet signals. For details, see **Chapter 20**, *Ethernet Controller*.
SRIO_RXD3 GE2_SGMII_RX	Input Input	**SRIO Receive Data 3** Serial data input for a 4x link. Each signal is part of a differential pair. **Ethernet 2 SGMII Receive Data** Part of the Ethernet signals. For details, see **Chapter 20**, *Ethernet Controller*.

Table 3-7. Serial RapidIO Signals (Continued)

Signal Name	Type	Description
$\overline{\text{SRIO_RXD2}}$	Input	**SRIO Receive Data 2 Inverted** Inverted serial data input for a 4x link. Each signal is part of a differential pair.
$\overline{\text{GE1_SGMII_RX}}$	Input	**Ethernet 1 SGMII Receive Data Inverted** Part of the Ethernet signals. For details, see **Chapter 20**, *Ethernet Controller*.
$\overline{\text{SRIO_RXD3}}$	Input	**SRIO Receive Data 3 Inverted** Inverted serial data input for a 4x link. Each signal is part of a differential pair.
$\overline{\text{GE2_SGMII_RX}}$	Input	**Ethernet 1 SGMII Receive Data Inverted** Part of the Ethernet signals. For details, see **Chapter 20, *Ethernet Controller***.
SRIO_TXD0	Output	**SRIO Transmit Data 0** Serial data output for a 1x or 4x link. Each signal is part of a differential pair.
$\overline{\text{SRIO_TXD0}}$	Output	**SRIO Transmit Data 0 Inverted** Inverted serial data output for a 1x or 4x link. Each signal is part of a differential pair.
SRIO_TXD1	Output	**SRIO Transmit Data 1** Serial data output for a 4x link. Each signal is part of a differential pair.
$\overline{\text{SRIO_TXD1}}$	Output	**SRIO Transmit Data 1 Inverted** Inverted serial data output for a 4x link. Each signal is part of a differential pair.
SRIO_TXD2	Output	**SRIO Transmit Data 2** Serial data output for a 4x link. Each signal is part of a differential pair.
GE1_SGMII_TX	Output	**Ethernet 1 SGMII Transmit Data** Part of the Ethernet signals. For details, see **Chapter 20**, *Ethernet Controller*.
SRIO_TXD3	Output	**SRIO Transmit Data 3** Serial data output for a 4x link. Each signal is part of a differential pair.
GE2_SGMII_TX	Output	**Ethernet 2 SGMII Transmit Data** Part of the Ethernet signals. For details, see **Chapter 20**, *Ethernet Controller*.
$\overline{\text{SRIO_TXD2}}$	Output	**SRIO Transmit Data 2 Inverted** Inverted serial data output for a 4x link. Each signal is part of a differential pair.
$\overline{\text{GE1_SGMII_TX}}$	Output	**Ethernet 1 SGMII Transmit Data Inverted** Part of the Ethernet signals. For details, see **Chapter 20**, *Ethernet Controller*.
$\overline{\text{SRIO_TXD3}}$	Output	**SRIO Transmit Data 3 Inverted** Inverted serial data output for a 4x link. Each signal is part of a differential pair.
$\overline{\text{GE2_SGMII_TX}}$	Output	**Ethernet 2 SGMII Transmit Data Inverted** Part of the Ethernet signals. For details, see **Chapter 20, *Ethernet Controller***.
SRIO_REF_CLK	Input	**SRIO Reference Clock** Reference clock signal. Each signal is part of a differential pair. For SGMII operation, this provides the input clock.
$\overline{\text{SRIO_REF_CLK}}$	Input	**SRIO Reference Clock Inverted** Inverted reference clock signal. Each signal is part of a differential pair. For SGMII operation, this provides the input clock.
Note: For proper definition of serial RapidIO modes (1x/4x) and SGMII, configure the interfaces using the Reset Configuration Word settings. For details, see **Chapter 5**, *Reset*.		

3.6 PCI Signals

Table 3-8 describes the signals in this group.

Table 3-8. PCI Signals

Signal Name	Type	Description	I/O Mode
PCI_AD31	Input/ Output	**PCI Address/Data Line 31** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD9	Output	**ATM UTOPIA Transmit Data 9** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD31	Input/ Output	**PCI Address/Data Line 31** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_TX_CLK	Input	**Ethernet 1 Transmit Clock** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_RD0	Input	**ATM UTOPIA Receive Data 0** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_AD31	Input/ Output	**PCI Address/Data Line 31** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_RX_ER	Input	**Ethernet 2 Receive Error** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6
PCI_AD30	Input/ Output	**PCI Address/Data Line 30** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD8	Output	**ATM UTOPIA Transmit Data 8** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD30	Input/ Output	**PCI Address/Data Line 30** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_TD3	Output	**Ethernet 1 Transmit Data 3** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_TD5	Output	**ATM UTOPIA Transmit Data 5** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_AD30	Input/ Output	**PCI Address/Data Line 30** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_RX_DV	Input	**Ethernet 2 Receive Data Valid** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6
PCI_AD29	Input/ Output	**PCI Address/Data Line 29** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TCLK	Input	**ATM UTOPIA Transmit Clock** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD29	Input/ Output	**PCI Address/Data Line 29** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_TD2	Output	**Ethernet 1 Transmit Data 2** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_TD4	Output	**ATM UTOPIA Transmit Data 4** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7

Table 3-8. PCI Signals (Continued)

Signal Name	Type	Description	I/O Mode
PCI_AD29	Input/ Output	**PCI Address/Data Line 29** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_RX_CLK	Input	**Ethernet 2 Receive Clock** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6
PCI_AD28	Input/ Output	**PCI Address/Data Line 28** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TCLAV	Input/ Output	**UTOPIA Transmit Cell Available** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD28	Input/ Output	**PCI Address/Data Line 28** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_TD1	Output	**Ethernet 1 Transmit Data 1** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_TD3	Output	**UTOPIA Transmit Data 3** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_AD28	Input/ Output	**PCI Address/Data Line 28** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_RD1	Input	**Ethernet 2 Receive Data 1** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6
PCI_AD27	Input/ Output	**PCI Address/Data Line 27** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TADDR4	Input/ Output	**UTOPIA Transmit Address 4** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD27	Input/ Output	**PCI Address/Data Line 27** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_TD0	Output	**Ethernet 1 Transmit Data 0** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_TD2	Output	**UTOPIA Transmit Data 2** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_AD27	Input/ Output	**PCI Address/Data Line 27** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_RD0	Input	**Ethernet 2 Receive Data 0** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6
PCI_AD26	Input/ Output	**PCI Address/Data Line 26** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	All modes
PCI_AD25	Input/ Output	**PCI Address/Data Line 25** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	All modes
PCI_AD24	Input/ Output	**PCI Address/Data Line 24** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TADDR1	Input/ Output	**ATM UTOPIA Transmit Address 1** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7

Table 3-8. PCI Signals (Continued)

Signal Name	Type	Description	I/O Mode
PCI_AD24	Input/ Output	**PCI Address/Data Line 24** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO8	Input/ Output	**General-Purpose Input Output 8** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ14}}$	Input	**Interrupt Request 14** One of sixteen external lines that can request a service routine via the internal interrupt controller. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
TDM6TSYN	Input/ Output	**TDM6 Transmit Frame Sync** The transmit sync signal for TDM 6. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For TDM configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD23	Input/ Output	**PCI Address/Data Line 23** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TADDR0	Input/ Output	**ATM UTOPIA Transmit Address 0** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD23	Input/ Output	**PCI Address/Data Line 23** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO7	Input/ Output	**General-Purpose Input Output 7** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ13}}$	Input	**Interrupt Request 13** One of sixteen external lines that can request a service routine via the internal interrupt controller. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
TDM6TDAT	Input/ Output	**TDM6 Transmit Data** The transmit data signal for TDM 6. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For TDM configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD22	Input/ Output	**PCI Address/Data Line 22** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RSOC	Input	**ATM UTOPIA Receive Start of Cell** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD22	Input/ Output	**PCI Address/Data Line 22** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM6TCLK	Input	**TDM6 Transmit Clock** Transmit Clock for TDM 6. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD21	Input/ Output	**PCI Address/Data Line 21** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RPRTY	Input	**ATM UTOPIA Receive Parity** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7

Table 3-8. PCI Signals (Continued)

Signal Name	Type	Description	I/O Mode
PCI_AD21	Input/ Output	**PCI Address/Data Line 21** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO6	Input/ Output	**General-Purpose Input Output 6** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ12}}$	Input	**Interrupt Request 12** One of sixteen external lines that can request a service routine via the internal interrupt controller. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
TDM6RSYN	Input/ Output	**TDM6 Receive Frame Sync** The receive sync signal for TDM 6. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For TDM configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD20	Input/ Output	**PCI Address/Data Line 20** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
$\overline{\text{UTP_REN}}$	Input/ Output	**ATM UTOPIA Receive Enable** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD20	Input/ Output	**PCI Address/Data Line 20** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO5	Input/ Output	**General-Purpose Input Output 5** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ11}}$	Input	**Interrupt Request 11** One of sixteen external lines that can request a service routine via the internal interrupt controller. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
TDM6RDAT	Input/ Output	**TDM6 Receive Data** The receive data signal for TDM 6. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For TDM configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD19	Input/ Output	**PCI Address/Data Line 19** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RD15	Input	**ATM UTOPIA Receive Data 15** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD19	Input/ Output	**PCI Address/Data Line 19** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO4	Input/ Output	**General-Purpose Input Output 4** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ10}}$	Input	**Interrupt Request 10** One of sixteen external lines that can request a service routine via the internal interrupt controller. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
TDM6RCLK	Input/ Output	**TDM6 Receive Clock** The receive clock signal for TDM 6. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For TDM configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6

Table 3-8. PCI Signals (Continued)

Signal Name	Type	Description	I/O Mode
PCI_AD18	Input/ Output	**PCI Address/Data Line 18** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RD14	Input	**ATM UTOPIA Receive Data 14** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD18	Input/ Output	**PCI Address/Data Line 18** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO12	Input/ Output	**General-Purpose Input Output 12** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
TDM5TSYN	Input/ Output	**TDM5 Transmit Frame Sync** The transmit sync signal for TDM 5. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For TDM configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD17	Input/ Output	**PCI Address/Data Line 17** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RD13	Input	**ATM UTOPIA Receive Data 13** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD17	Input/ Output	**PCI Address/Data Line 17** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO11	Input/ Output	**General-Purpose Input Output 11** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
TDM5TDAT	Input/ Output	**TDM5 Serial Transmitter Data** The transmit data signal for TDM 5. As an output, this can be the DATA_D data signal for TDM 5. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For TDM configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD16	Input/ Output	**PCI Address/Data Line 16** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RD12	Input	**ATM UTOPIA Receive Data 12** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD16	Input/ Output	**PCI Address/Data Line 16** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM5TCLK	Input	**TDM5 Transmit Clock** Transmit Clock for TDM 5. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD15	Input/ Output	**PCI Address/Data Line 15** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RD11	Input	**ATM UTOPIA Receive Data 11** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7

Table 3-8. PCI Signals (Continued)

Signal Name	Type	Description	I/O Mode
PCI_AD15	Input/ Output	**PCI Address/Data Line 15** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO10	Input/ Output	**General-Purpose Input Output 10** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
TDM5RSYN	Input/ Output	**TDM5 Receive Frame Sync** The receive sync signal for TDM 5. As an input, this can be the DATA_B data signal for TDM 5. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For TDM configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD14	Input/ Output	**PCI Address/Data Line 14** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RD10	Input	**ATM UTOPIA Receive Data 10** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD14	Input/Output	**PCI Address/Data Line 14** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO9	Input/ Output	**General-Purpose Input Output 9** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
TDM5RDAT	Input/ Output	**TDM5 Serial Receiver Data** The receive data signal for TDM 5. As an input, this can be the DATA_A data signal for TDM 5. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For TDM configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD13	Input/ Output	**PCI Address/Data Line 13** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RCLK	Input	**ATM UTOPIA Receive Clock** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD13	Input/ Output	**PCI Address/Data Line 13** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO28	Input/ Output	**General-Purpose Input Output 28** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
TDM5RCLK	Input/ Output	**TDM5 Receive Clock** Receive clock for TDM 5. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For TDM configuration details, **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD12	Input/ Output	**PCI Address/Data Line 12** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RCLAV_ PDRPA	Input/ Output	**ATM UTOPIA Receive Cell Available** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD12	Input/ Output	**PCI Address/Data Line 12** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM4TSYN	Input/ Output	**TDM4 Transmit Sync** Transmit sync signal for TDM 4. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For TDM configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6

Table 3-8. PCI Signals (Continued)

Signal Name	Type	Description	I/O Mode
PCI_AD11	Input/ Output	**PCI Address/Data Line 11** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RADDR4	Input/ Output	**ATM UTOPIA Receive Address 4** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD11	Input/ Output	**PCI Address/Data Line 11** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM4TDAT	Input/ Output	**TDM4 Serial Transmitter Data** The serial transmit data signal for TDM 4. As an output, it provides the DATA_D signal for TDM 4. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For TDM configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD10	Input/ Output	**PCI Address/Data Line 10** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RADDR3	Input/ Output	**ATM UTOPIA Receive Address 3** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD10	Input/ Output	**PCI Address/Data Line 10** Part of the PCI address/data bus. For details, see **Chapter 15, *PCI***.	3,4
TDM4TCLK	Input	**TDM4 Transmit Clock** Transmit clock for TDM 4. Selected through GPIO configuration. For details, see **Chapter 23**, *GPIO*. For TDM configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD9	Input/ Output	**PCI Address/Data Line 9** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RADDR2	Input/ Output	**ATM UTOPIA Receive Address 2** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD9	Input/ Output	**PCI Address/Data Line 9** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM4RSYN	Input/ Output	**TDM4 Receive Frame Sync** The receive sync signal for TDM 4. As an input, this can be the DATA_B data signal for TDM 4. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD8	Input/ Output	**PCI Address/Data Line 8** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RADDR1	Input/ Output	**ATM UTOPIA Receive Address 1** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD8	Input/ Output	**PCI Address/Data Line 8** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM4RDAT	Input/ Output	**TDM4 Serial Receiver Data** The receive data signal for TDM 4. As an input, this can be the DATA_A data signal for TDM 4. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD7	Input/ Output	**PCI Address/Data Line 7** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RADDR0	Input/ Output	**ATM UTOPIA Receive Address 0** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7

Table 3-8. PCI Signals (Continued)

Signal Name	Type	Description	I/O Mode
PCI_AD7	Input/ Output	**PCI Address/Data Line 7** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM4RCLK	Input/ Output	**TDM4 Receive Clock** The receive clock signal for TDM 4. As an output, this can be the DATA_C data signal for TDM 4. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD6	Input/ Output	**PCI Address/Data Line 6** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
GPIO25	Input/ Output	**General-Purpose Input Output 25** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,5
$\overline{\text{IRQ15}}$	Input	**Interrupt Request 15** One of sixteen external lines that can request a service routine via the internal interrupt controller. Configured through the GPIO port. For details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,5
GE1_RX_ER	Input	**Ethernet 1 Receive Error** For details, see **Chapter 19**, *Ethernet Controller*.	1,6
PCI_AD5	Input/ Output	**PCI Address/Data Line 5** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	0,2,3,4,5
GE1_CRS	Input	**Ethernet 1 Carrier Sense** For details, see **Chapter 19**, *Ethernet Controller*.	1,6
PCI_AD4	Input/ Output	**PCI Address/Data Line 4** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
TDM7TSYN	Input/ Output	**TDM7 Transmit Frame Sync** Transmit frame sync for TDM 7. See **Chapter 20**, *TDM Interface*.	0,1
UTP_RMOD	Input	**Receive Word Modulo**	7
PCI_AD3	Input/ Output	**PCI Address/Data Line 3** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
TDM7TDAT	Input/ Output	**TDM7 Serial Transmitter Data** The serial transmit data signal for TDM 7. As an output, it provides the DATA_D signal for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
GE2_TD3	Output	**Ethernet 2 Transmit Data 3** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
UTP_TMD	Output	**Transmit Word Modulo**	7
PCI_AD2	Input/ Output	**PCI Address/Data Line 2** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
TDM7RSYN	Input/ Output	**TDM7 Receive Frame Sync** The receive sync signal for TDM 7. As an input, this can be the DATA_B data signal for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
GE2_TD2	Output	**Ethernet 2 Transmit Data 2** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
UTP_TER	Output	**Transmit Error**	7

Table 3-8. PCI Signals (Continued)

Signal Name	Type	Description	I/O Mode
PCI_AD1	Input/ Output	**PCI Address/Data Line 1** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
TDM7RDAT	Input/ Output	**TDM7 Serial Receiver Data** The receive data signal for TDM 7. As an input, this can be the DATA_A data signal for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
GE2_RD3	Input	**Ethernet 2 Receive Data 3** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
UTP_STA	Output	**Transmit Start-of-Packet**	7
PCI_AD0	Input/ Output	**PCI Address/Data Line 0** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
TDM7RCLK	Input/ Output	**TDM7 Receive Clock** The receive clock signal for TDM 7. As an output, this can be the DATA_C data signal for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
GE2_RD2	Input	**Ethernet 2 Receive Data 2** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
UTP_RVL	Input	**Receive Data Valid**	7
$\overline{\text{PCI_CBE0}}$	Input/ Output	**PCI Byte 0 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD10	Output	**ATM UTOPIA Transmit Data 10** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
$\overline{\text{PCI_CBE0}}$	Input/ Output	**PCI Byte 0 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_TX_EN	Output	**Ethernet 1 Transmit Enable** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_TD6	Output	**ATM UTOPIA Transmit Data 6** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
$\overline{\text{PCI_CBE0}}$	Input/ Output	**PCI Byte 0 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_TD0	Output	**Ethernet 2 Transmit Data 0** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6
$\overline{\text{PCI_CBE1}}$	Input/ Output	**PCI Byte 1 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD11	Output	**ATM UTOPIA Transmit Data 11** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
$\overline{\text{PCI_CBE1}}$	Input/ Output	**PCI Byte 1 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_TX_ER	Output	**Ethernet 1 Transmit Error** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_TD7	Output	**ATM UTOPIA Transmit Data 7** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7

Table 3-8. PCI Signals (Continued)

Signal Name	Type	Description	I/O Mode
PCI_CBE1	Input/ Output	**PCI Byte 1 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_TD1	Output	**Ethernet 2 Transmit Data 1** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6
PCI_CBE2	Input/ Output	**PCI Byte 2 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD12	Output	**ATM UTOPIA Transmit Data 12** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_CBE2	Input/ Output	**PCI Byte 2 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_RD0	Input	**Ethernet 1 Receive Data 0** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_RD2	Input	**ATM UTOPIA Receive Data 2** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_CBE2	Input/ Output	**PCI Byte 2 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_TX_EN	Output	**Ethernet 2 Transmit Enable** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6
PCI_CBE3	Input/ Output	**PCI Byte 3 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD13	Output	**ATM UTOPIA Transmit Data 13** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_CBE3	Input/ Output	**PCI Byte 3 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_RD1	Input	**Ethernet 1 Receive Data 1** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_RD3	Input	**ATM UTOPIA Receive Data 3** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_CBE3	Input/ Output	**PCI Byte Enable 3** For details, see **Chapter 15**, *PCI*.	4
GPIO17	Input/ Output	**General-Purpose Input Output 17** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	2
TMR1	Input/ Output	**Timer 1** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For timer functional details, see **Chapter 22**, *Timers*.	2
UTP_IR	Input	**UTOPIA IR** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,5,6,7

Table 3-8. PCI Signals (Continued)

Signal Name	Type	Description	I/O Mode
PCI_IDSL	Input	**PCI IDSL** For details, see **Chapter 15**, *PCI*.	2,3,4
TDM7TCLK	Input	**TDM7 Transmit Clock** Transmit Clock for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
GE2_TCK	Output	**Ethernet 2 Transmit Clock** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
UTP_RER	Input	**Receive Error**	7
PCI_PAR	Input/ Output	**PCI Parity** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
$\overline{\text{UTP_TEN}}$	Input/ Output	**ATM UTOPIA Transmit Enable** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_PAR	Input/ Output	**PCI Parity** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_RX_CLK	Input	**Ethernet 1 Receive Clock** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_RD6	Input	**ATM UTOPIA Receive Data 6** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_PAR	Input/ Output	**PCI Parity** For details, see **Chapter 15, *PCI***.	4
GPIO20	Input/ Output	**General-Purpose Input Output 20** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,3,5,6
TMR4	Input/ Output	**Timer 4** The signal can be configured as an input to the counter or an output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For timer functional details, see **Chapter 22**, *Timers*.	0,1,2,3,5,6
UTP_REOP	Input	**Receive End-of-Packet**	7
$\overline{\text{PCI_FRAME}}$	Input/ Output	**PCI Frame Sync** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD14	Output	**ATM UTOPIA Transmit Data 14** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
$\overline{\text{PCI_FRAME}}$	Input/ Output	**PCI Frame Sync** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_RD2	Input	**Ethernet 1 Receive Data 2** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_RD4	Input	**ATM UTOPIA Receive Data 4** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7

Table 3-8. PCI Signals (Continued)

Signal Name	Type	Description	I/O Mode
PCI_FRAME	Input/ Output	**PCI Frame Sync** For details, see **Chapter 15**, *PCI*.	4
GPIO18	Input/ Output	**General-Purpose Input Output 18** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,3,5,6
TMR2	Input/ Output	**Timer 2** The signal can be configured as an input to the counter or an output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For timer functional details, see **Chapter 22**, *Timers*.	0,1,2,3,5,6
UTP_SRP	Input	**Receive Start-of-Packet**	7
PCI_IRDY	Input/ Output	**PCI Ready** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD15	Output	**ATM UTOPIA Transmit Data 15** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_IRDY	Input/ Output	**PCI Ready** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_RD3	Input	**Ethernet 1 Receive Data 3** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_RD5	Input	**ATM UTOPIA Receive Data 5** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_IRDY	Input/ Output	**PCI Ready** For details, see **Chapter 15**, *PCI*.	4
GPIO19	Input/ Output	**General-Purpose Input Output 19** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,3,5,6
TMR3	Input/ Output	**Timer 3** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 22**, *Timers*.	0,1,2,3,5,6
UTP_TEOP	Output	**Transmit End-of-Packet**	7
PCI_GNT	Input	**PCI Bus Grant** For details, see **Chapter 15**, *PCI*.	2,3,4
GPIO29	Input/ Output	**General-Purpose Input Output 29** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,5,6
IRQ7	Input	**Interrupt Request 7** One of the sixteen external lines that can request a service routine via the internal interrupt controller. Configured as part of the GPIO port. For details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,1,5,6

Table 3-8. PCI Signals (Continued)

Signal Name	Type	Description	I/O Mode
PCI_STOP	Input/ Output	**PCI Stop** For details, see **Chapter 15**, *PCI*.	2,3,4
GPIO30	Input/ Output	**General-Purpose Input Output 30** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,5,6
IRQ2	Input	**Interrupt Request 2** One of the sixteen external lines that can request a service routine via the internal interrupt controller. Configured as part of the GPIO port. For details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,1,5,6
PCI_REQ	Output	**PCI Bus Request** For details, see **Chapter 15**, *PCI*.	All modes
PCI_TRDY	Input/ Output	**PCI Transmit Ready** For details, see **Chapter 15**, *PCI*.	All modes
PCI_DEVSEL	Input/ Output	**PCI Device Select** For details, see **Chapter 15**, *PCI*.	2,3,4
GPIO31	Input/ Output	**General-Purpose Input Output 31** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,5,6
IRQ3	Input	**Interrupt Request 3** One of sixteen external lines that can request a service routine via the internal interrupt controller. Configured as part of the GPIO port. For details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,1,5,6
PCI_PERR	Input/ Output	**PCI Parity Error** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3
UTP_TD1	Output	**ATM UTOPIA Transmit Data 1** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,4,5,6,7
PCI_SERR	Input/ Output	**PCI System Error** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3
UTP_TD0	Output	**ATM UTOPIA Transmit Data 0** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,4,5,6,7

3.7 Ethernet Signals

Table 3-9 describes the signals in this group.

Table 3-9. Ethernet Signals

Signal Name	Type	Description	I/O Mode
GE2_TD3	Output	**Ethernet 2 Transmit Data 3** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
TDM7TDAT	Input/ Output	**TDM7 Serial Transmitter Data** The serial transmit data signal for TDM 7. As an output, it provides the DATA_D signal for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
PCI_AD3	Input/ Output	**PCI Address/Data Line 3** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
UTP_TMD	Output	**Transmit Word Modulo**	7
GE2_TD2	Output	**Ethernet 2 Transmit Data 2** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
TDM7RSYN	Input/ Output	**TDM7 Receive Frame Sync** The receive sync signal for TDM 7. As an input, this can be the DATA_B data signal for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
PCI_AD2	Input/ Output	**PCI Address/Data Line 2** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
UTP_TER	Output	**Transmit Error**	7
GE2_TD1	Output	**Ethernet 2 Transmit Data 1** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6, 7
$\overline{\text{PCI_CBE1}}$	Input/ Output	**PCI Byte 1 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_TD0	Output	**Ethernet 2 Transmit Data 0** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6, 7
$\overline{\text{PCI_CBE0}}$	Input/ Output	**PCI Byte 0 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_RD3	Input	**Ethernet 2 Receive Data 3** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
TDM7RDAT	Input/ Output	**TDM7 Serial Receiver Data** The receive data signal for TDM 7. As an input, this can be the DATA_A data signal for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
PCI_AD1	Input/ Output	**PCI Address/Data Line 1** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
UTP_STA	Output	**Transmit Start-of-Packet**	7

Table 3-9. Ethernet Signals (Continued)

Signal Name	Type	Description	I/O Mode
GE2_RD2	Input	**Ethernet 2 Receive Data 2** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
TDM7RCLK	Input/ Output	**TDM7 Receive Clock** The receive clock signal for TDM 7. As an output, this can be the DATA_C data signal for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
PCI_AD0	Input/ Output	**PCI Address/Data Line 0** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
UTP_RVL	Input	**Receive Data Valid**	7
GE2_RD1	Input	**Ethernet 2 Receive Data 1** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6, 7
PCI_AD28	Input/ Output	**PCI Address/Data Line 28** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_RD0	Input	**Ethernet 2 Receive Data 0** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6, 7
PCI_AD27	Input/ Output	**PCI Address/Data Line 27** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_TX_EN	Output	**Ethernet 2 Transmit Enable** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6, 7
$\overline{\text{PCI_CBE2}}$	Input/ Output	**PCI Byte 2 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_TCK	Output	**Ethernet 2 Transmit Clock** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
TDM7TCLK	Input	**TDM7 Transmit Clock** Transmit Clock for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
PCI_IDSL	Input	**PCI IDSL** For details, see **Chapter 15**, *PCI*.	2,3,4
UTP_RER	Input	**Receive Error**	7
GE2_RX_DV	Input	**Ethernet 2 Receive Data Valid** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6, 7
PCI_AD30	Input/ Output	**PCI Address/Data Line 30** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_RX_ER	Input	**Ethernet 2 Receive Error** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6, 7
PCI_AD31	Input/ Output	**PCI Address/Data Line 31** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4
GE2_RX_CLK	Input	**Ethernet 2 Receive Clock** For details, see **Chapter 19**, *Ethernet Controller*.	0,1,2,3,5,6, 7
PCI_AD29	Input/ Output	**PCI Address/Data Line 29** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	4

Table 3-9. Ethernet Signals (Continued)

Signal Name	Type	Description	I/O Mode
GE_MDC	Output	**Ethernet Management Data Clock** For details, see **Chapter 19**, *Ethernet Controller*.	All modes
GE_MDIO	Input/ Output	**Ethernet Management Data Input/Output** For details, see **Chapter 19**, *Ethernet Controller*.	All modes
GE1_TD3	Output	**Ethernet 1 Transmit Data 3** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
PCI_AD30	Input/ Output	**PCI Address/Data Line 30** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
UTP_TD5	Output	**ATM UTOPIA Transmit Data 5** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
GE1_TD2	Output	**Ethernet 1 Transmit Data 2** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
PCI_AD29	Input/ Output	**PCI Address/Data Line 29** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
UTP_TD4	Output	**ATM UTOPIA Transmit Data 4** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
GE1_TD1	Output	**Ethernet 1 Transmit Data 1** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
PCI_AD28	Input/ Output	**PCI Address/Data Line 28** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
UTP_TD3	Output	**ATM UTOPIA Transmit Data 3** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
GE1_TD0	Output	**Ethernet 1 Transmit Data 0** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
PCI_AD27	Input/ Output	**PCI Address/Data Line 27** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
UTP_TD2	Output	**ATM UTOPIA Transmit Data 2** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
GE1_RD3	Input	**Ethernet 1 Receive Data 3** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
$\overline{\text{PCI_IRDY}}$	Input/ Output	**PCI Ready** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
UTP_RD5	Input	**ATM UTOPIA Receive Data 5** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
GE1_RD2	Input	**Ethernet 1 Receive Data 2** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
$\overline{\text{PCI_FRAME}}$	Input/ Output	**PCI Frame Sync** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
UTP_RD4	Input	**ATM UTOPIA Receive Data 4** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7

Table 3-9. Ethernet Signals (Continued)

Signal Name	Type	Description	I/O Mode
GE1_RD1	Input	**Ethernet 1 Receive Data 1** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
PCI_CBE3	Input/ Output	**PCI Byte 3 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
UTP_RD3	Input	**ATM UTOPIA Receive Data 3** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
GE1_RD0	Input	**Ethernet 1 Receive Data 0** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
PCI_CBE2	Input/ Output	**PCI Byte 2 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
UTP_RD2	Input	**ATM UTOPIA Receive Data 2** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
GE1_TX_EN	Output	**Ethernet 1 Transmit Enable** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
PCI_CBE0	Input/ Output	**PCI Byte 0 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
UTP_TD6	Output	**ATM UTOPIA Transmit Data 6** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
GE1_TX_ER	Output	**Ethernet 1 Transmit Error** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
PCI_CBE1	Input/ Output	**PCI Byte 1 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
UTP_TD7	Output	**ATM UTOPIA Transmit Data 7** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
GE1_TX_CLK	Input	**Ethernet 1 Transmit Clock** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
PCI_AD31	Input/ Output	**PCI Address/Data Line 31** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
UTP_RD0	Input	**ATM UTOPIA Receive Data 0** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
GE1_RX_DV	Input	**Ethernet 1 Receive Data Valid** Part of the Ethernet signals. For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_RD7	Input	**ATM UTOPIA Receive Data 7** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,3,4,5,7

Table 3-9. Ethernet Signals (Continued)

Signal Name	Type	Description	I/O Mode
GE1_RX_ER	Input	**Ethernet 1 Receive Error** For details, see **Chapter 19**, *Ethernet Controller*.	1,6
GPIO25	Input/ Output	**General-Purpose Input Output 25** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,5
$\overline{\text{IRQ15}}$	Input	**Interrupt Request 15** One of sixteen external lines that can request a service routine via the internal interrupt controller. Configured through the GPIO port. For details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,5
PCI_AD6	Input/ Output	**PCI Address/Data Line 6** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
GE1_RX_CLK	Input	**Ethernet 1 Receive Clock** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
PCI_PAR	Input/ Output	**PCI Parity** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
UTP_RD6	Input	**ATM UTOPIA Receive Data 6** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
GE1_COL	Input	**Ethernet 1 Collision** Part of the Ethernet signals. For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_RD1	Input	**ATM UTOPIA Receive Data 1** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,3,4,5,7
GE1_CRS	Input	**Ethernet 1 Carrier Sense** For details, see **Chapter 19**, *Ethernet Controller*.	1,6
PCI_AD5	Input/ Output	**PCI Address/Data Line 5** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	0,2,3,4,5
SRIO_RXD2	Input	**SRIO Receive Data 2** Serial data input for a 4x link. Each signal is part of a differential pair.	All modes
GE1_SGMII_RX	Input	**Ethernet 1 SGMII Receive Data** Part of the Ethernet signals. For details, see **Chapter 20**, *Ethernet Controller*.	
SRIO_RXD3	Input	**SRIO Receive Data 3** Serial data input for a 4x link. Each signal is part of a differential pair.	All modes
GE2_SGMII_RX	Input	**Ethernet 2 SGMII Receive Data** Part of the Ethernet signals. For details, see **Chapter 20**, *Ethernet Controller*.	
$\overline{\text{SRIO_RXD2}}$	Input	**SRIO Receive Data 2 Inverted** Inverted serial data input for a 4x link. Each signal is part of a differential pair.	All modes
$\overline{\text{GE1_SGMII_RX}}$	Input	**Ethernet 1 SGMII Receive Data Inverted** Part of the Ethernet signals. For details, see **Chapter 20**, *Ethernet Controller*.	
$\overline{\text{SRIO_RXD3}}$	Input	**SRIO Receive Data 3 Inverted** Inverted serial data input for a 4x link. Each signal is part of a differential pair.	All modes
$\overline{\text{GE2_SGMII_RX}}$	Input	**Ethernet 2 SGMII Receive Data Inverted** Part of the Ethernet signals. For details, see **Chapter 20, *Ethernet Controller***.	

Table 3-9. Ethernet Signals (Continued)

Signal Name	Type	Description	I/O Mode
SRIO_TXD2	Output	**SRIO Transmit Data 2** Serial data output for a 4x link. Each signal is part of a differential pair.	All modes
GE1_SGMII_TX	Output	**Ethernet 1 SGMII Transmit Data** Part of the Ethernet signals. For details, see **Chapter 20**, *Ethernet Controller*.	
SRIO_TXD3	Output	**SRIO Transmit Data 3** Serial data output for a 4x link. Each signal is part of a differential pair.	All modes
GE2_SGMII_TX	Output	**Ethernet 2 SGMII Transmit Data** Part of the Ethernet signals. For details, see **Chapter 20**, *Ethernet Controller*.	
$\overline{\text{SRIO_TXD2}}$	Output	**SRIO Transmit Data 2 Inverted** Inverted serial data output for a 4x link. Each signal is part of a differential pair.	All modes
$\overline{\text{GE1_SGMII_TX}}$	Output	**Ethernet 1 SGMII Transmit Data Inverted** Part of the Ethernet signals. For details, see **Chapter 20**, *Ethernet Controller*.	
$\overline{\text{SRIO_TXD3}}$	Output	**SRIO Transmit Data 3 Inverted** Inverted serial data output for a 4x link. Each signal is part of a differential pair.	All modes
$\overline{\text{GE2_SGMII_TX}}$	Output	**Ethernet 2 SGMII Transmit Data Inverted** Part of the Ethernet signals. For details, see **Chapter 20, *Ethernet Controller***.	

3.8 ATM UTOPIA Signals

Table 3-10 describes the signals in this group.

Table 3-10. UTOPIA Signals

Signal Name	Type	Description	I/O Mode
UTP_TD15	Output	**ATM UTOPIA Transmit Data 15** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
$\overline{\text{PCI_IRDY}}$	Input/ Output	**PCI Ready** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD14	Output	**ATM UTOPIA Transmit Data 14** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
$\overline{\text{PCI_FRAME}}$	Input/ Output	**PCI Frame Sync** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD13	Output	**ATM UTOPIA Transmit Data 13** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
$\overline{\text{PCI_CBE3}}$	Input/ Output	**PCI Byte 3 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD12	Output	**ATM UTOPIA Transmit Data 12** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
$\overline{\text{PCI_CBE2}}$	Input/ Output	**PCI Byte 2 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2

Table 3-10. UTOPIA Signals (Continued)

Signal Name	Type	Description	I/O Mode
UTP_TD11	Output	**ATM UTOPIA Transmit Data 11** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
$\overline{\text{PCI_CBE1}}$	Input/ Output	**PCI Byte 1 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD10	Output	**ATM UTOPIA Transmit Data 10** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
$\overline{\text{PCI_CBE0}}$	Input/ Output	**PCI Byte 0 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD9	Output	**ATM UTOPIA Transmit Data 9** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD31	Input/ Output	**PCI Address/Data Line 31** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD8	Output	**ATM UTOPIA Transmit Data 8** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD30	Input/ Output	**PCI Address/Data Line 30** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TD7	Output	**ATM UTOPIA Transmit Data 7** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
$\overline{\text{PCI_CBE1}}$	Input/ Output	**PCI Byte 1 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_TX_ER	Output	**Ethernet 1 Transmit Error** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_TD6	Output	**ATM UTOPIA Transmit Data 6** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
$\overline{\text{PCI_CBE0}}$	Input/ Output	**PCI Byte 0 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_TX_EN	Output	**Ethernet 1 Transmit Enable** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_TD5	Output	**ATM UTOPIA Transmit Data 5** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_AD30	Input/ Output	**PCI Address/Data Line 30** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_TD3	Output	**Ethernet 1 Transmit Data 3** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_TD4	Output	**ATM UTOPIA Transmit Data 4** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_AD29	Input/ Output	**PCI Address/Data Line 29** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_TD2	Output	**Ethernet 1 Transmit Data 2** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6

Table 3-10. UTOPIA Signals (Continued)

Signal Name	Type	Description	I/O Mode
UTP_TD3	Output	**ATM UTOPIA Transmit Data 3** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_AD28	Input/ Output	**PCI Address/Data Line 28** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_TD1	Output	**Ethernet 1 Transmit Data 1** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_TD2	Output	**ATM UTOPIA Transmit Data 2** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_AD27	Input/ Output	**PCI Address/Data Line 27** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_TD0	Output	**Ethernet 1 Transmit Data 0** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_TD1	Output	**ATM UTOPIA Transmit Data 1** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,4,5, 6,7
$\overline{\text{PCI_PERR}}$	Input/ Output	**PCI Parity Error** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3
UTP_TD0	Output	**ATM UTOPIA Transmit Data 0** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,4,5,6,7
$\overline{\text{PCI_SERR}}$	Input/ Output	**PCI System Error** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3
UTP_RD15	Input	**ATM UTOPIA Receive Data 15** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD19	Input/ Output	**PCI Address/Data Line 19** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RD14	Input	**ATM UTOPIA Receive Data 14** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD18	Input/ Output	**PCI Address/Data Line 18** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RD13	Input	**ATM UTOPIA Receive Data 13** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD17	Input/ Output	**PCI Address/Data Line 17** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RD12	Input	**ATM UTOPIA Receive Data 12** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD16	Input/ Output	**PCI Address/Data Line 16** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RD11	Input	**ATM UTOPIA Receive Data 11** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD15	Input/ Output	**PCI Address/Data Line 15** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2

Table 3-10. UTOPIA Signals (Continued)

Signal Name	Type	Description	I/O Mode
UTP_RD10	Input	**ATM UTOPIA Receive Data 10** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD14	Input/ Output	**PCI Address/Data Line 14** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RD9	Input	**ATM UTOPIA Receive Data 9** For details, see **Chapter 19**, *Asynchronous Transfer Mode (ATM) Controller*.	All modes
RC13	Input	**Reset Configuration Word Bit 13** Used to load part of the Reset Configuration Word during Reset.	Reset
UTP_RD8	Input	**ATM UTOPIA Receive Data 8** For details, see **Chapter 19**, *Asynchronous Transfer Mode (ATM) Controller*.	All modes
RC12	Input	**Reset Configuration Word Bit 12** Used to load part of the Reset Configuration Word during Reset.	Reset
UTP_RD7	Input	**ATM UTOPIA Receive Data 7** For details, see **Chapter 19**, *Asynchronous Transfer Mode (ATM) Controller*.	0,3,4,5,7
GE1_RX_DV	Input	**Ethernet 1 Receive Data Valid** Part of the Ethernet signals. For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_RD6	Input	**ATM UTOPIA Receive Data 6** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_PAR	Input/ Output	**PCI Parity** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_RX_CLK	Input	**Ethernet 1 Receive Clock** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_RD5	Input	**ATM UTOPIA Receive Data 5** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
$\overline{\text{PCI_IRDY}}$	Input/ Output	**PCI Ready** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_RD3	Input	**Ethernet 1 Receive Data 3** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_RD4	Input	**ATM UTOPIA Receive Data 4** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
$\overline{\text{PCI_FRAME}}$	Input/ Output	**PCI Frame Sync** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_RD2	Input	**Ethernet 1 Receive Data 2** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_RD3	Input	**ATM UTOPIA Receive Data 3** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
$\overline{\text{PCI_CBE3}}$	Input/ Output	**PCI Byte 3 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_RD1	Input	**Ethernet 1 Receive Data 1** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6

Table 3-10. UTOPIA Signals (Continued)

Signal Name	Type	Description	I/O Mode
UTP_RD2	Input	**ATM UTOPIA Receive Data 2** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_CBE2	Input/ Output	**PCI Byte 2 Enable** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_RD0	Input	**Ethernet 1 Receive Data 0** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_RD1	Input	**ATM UTOPIA Receive Data 1** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,3,4,5,7
GE1_COL	Input	**Ethernet 1 Collision** Part of the Ethernet signals. For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_RD0	Input	**ATM UTOPIA Receive Data 0** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,4,5,7
PCI_AD31	Input/ Output	**PCI Address/Data 31** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3
GE1_TX_CLK	Input	**Ethernet 1 Transmit Clock** For details, see **Chapter 19**, *Ethernet Controller*.	1,2,6
UTP_TADDR4	Input/ Output	**ATM UTOPIA Transmit Address 4** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD27	Input/ Output	**PCI Address/Data Line 27** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TADDR3	Input/ Output	**ATM UTOPIA Transmit Address 3** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	All modes
UTP_TADDR2	Input/ Output	**ATM UTOPIA Transmit Address 2** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	All modes
UTP_TADDR1	Input/ Output	**ATM UTOPIA Transmit Address 1** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD24	Input/ Output	**PCI Address/Data Line 24** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TADDR0	Input/ Output	**ATM UTOPIA Transmit Address 0** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD23	Input/ Output	**PCI Address/Data Line 23** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RADDR4	Input/ Output	**ATM UTOPIA Receive Address 4** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD11	Input/ Output	**PCI Address/Data Line 11** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RADDR3	Input/ Output	**ATM UTOPIA Receive Address 3** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD10	Input/ Output	**PCI Address/Data Line 10** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2

Table 3-10. UTOPIA Signals (Continued)

Signal Name	Type	Description	I/O Mode
UTP_RADDR2	Input/ Output	**ATM UTOPIA Receive Address 2** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD9	Input/ Output	**PCI Address/Data Line 9** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RADDR1	Input/ Output	**ATM UTOPIA Receive Address 1** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD8	Input/ Output	**PCI Address/Data Line 8** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RADDR0	Input/ Output	**ATM UTOPIA Receive Address 0** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD7	Input/ Output	**PCI Address/Data Line 7** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_IR	Input	**UTOPIA IR** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,5,6,7
$\overline{\text{PCI_CBE3}}$	Input/ Output	**PCI Byte Enable 3** For details, see **Chapter 15**, *PCI*.	4
GPIO17	Input/ Output	**General-Purpose Input Output 17** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	2
TMR1	Input/ Output	**Timer 1** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For timer functional details, see **Chapter 22**, *Timers*.	2
UTP_TCLK	Input	**ATM UTOPIA Transmit Clock** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD29	Input/ Output	**PCI Address/Data Line 29** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RCLK	Input	**ATM UTOPIA Receive Clock** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD13	Input/ Output	**PCI Address/Data Line 13** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TCLAV	Input/ Output	**ATM UTOPIA Transmit Cell Available** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD28	Input/ Output	**PCI Address/Data Line 28** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_RCLAV_ PDRPA	Input/ Output	**ATM UTOPIA Receive Cell Available** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6, 7
PCI_AD12	Input/ Output	**PCI Address/Data Line 12** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2

Table 3-10. UTOPIA Signals (Continued)

Signal Name	Type	Description	I/O Mode
UTP_TSOC	Output	**ATM UTOPIA Transmit Start of Cell** For details, see **Chapter 19**, *Asynchronous Transfer Mode (ATM) Controller*.	All modes
RC15	Input	**Reset Configuration Word Bit 15** Used to load part of the Reset Configuration Word during Reset.	Reset
UTP_RSOC	Input	**ATM UTOPIA Receive Start of Cell** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD22	Input/ Output	**PCI Address/Data Line 22** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_TPRTY	Output	**ATM UTOPIA Transmit Parity** For details, see **Chapter 19**, *Asynchronous Transfer Mode (ATM) Controller*.	All modes
RC14	Input	**Reset Configuration Word Bit 14** Used to load part of the Reset Configuration Word during Reset.	Reset
UTP_RPRTY	Input	**ATM UTOPIA Receive Parity** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD21	Input/ Output	**PCI Address/Data Line 21** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
$\overline{\text{UTP_TEN}}$	Input/ Output	**ATM UTOPIA Transmit Enable** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_PAR	Input/ Output	**PCI Parity** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
$\overline{\text{UTP_REN}}$	Input/ Output	**ATM UTOPIA Receive Enable** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,4,5,6,7
PCI_AD20	Input/ Output	**PCI Address/Data Line 20** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2
UTP_REOP	Input	**Receive End-of-Packet**	7
TMR4	Input/ Output	**Timer 4** Configured as input to the counter or output from the counter. Selected through the GPIO configuration (see **Chapter 22**, *GPIO*). For timer functional details, see **Chapter 21**, *Timers*.	0,1,2,3,5,6
GPIO20	Input/ Output	**General-Purpose Input Output 20** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.	0,1,2,3,5,6
PCI_PAR	Input/ Output	**PCI Parity** For details, see **Chapter 15**, *PCI*.	4

Table 3-10. UTOPIA Signals (Continued)

Signal Name	Type	Description	I/O Mode
UTP_TEOP	Output	**Transmit End-of-Packet**	7
TMR3	Input/ Output	**Timer 3** Configured as input to the counter or output from the counter. Selected through the GPIO configuration (see **Chapter 22**, *GPIO*). For timer functional details, see **Chapter 21**, *Timers*.	0,1,2,3,5,6
GPIO19	Input/ Output	**General-Purpose Input Output 19** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.	0,1,2,3,5,6
$\overline{\text{PCI_IRDY}}$	Input/ Output	**PCI Ready** For details, see **Chapter 15**, *PCI*.	4
UTP_SRP	Input	**Receive Start-of-Packet**	7
TMR2	Input/ Output	**Timer 2** Configured as input to the counter or output from the counter. Selected through the GPIO configuration (see **Chapter 22**, *GPIO*). For timer functional details, see **Chapter 21**, *Timers*.	0,1,2,3,5,6
GPIO18	Input/ Output	**General-Purpose Input Output 18** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.	0,1,2,3,5,6
$\overline{\text{PCI_FRAME}}$	Input/ Output	**PCI Frame Sync** For details, see **Chapter 15**, *PCI*.	4
UTP_TMD	Output	**Transmit Word Modulo**	7
TDM7TDAT	Input/ Output	**TDM7 Serial Transmitter Data** The serial transmit data signal for TDM 7. As an output, it provides the DATA_D signal for TDM 7.	0,1
PCI_AD3	Input/ Output	**PCI Address/Data Line 3** Part of the PCI address/data bus.	2,3,4
GE2_TD3	Output	**Ethernet 2 Transmit Data 3.**	5,6
UTP_RER	Input	**Receive Error**	7
TDM7TCLK	Input	**TDM7 Transmit Clock** Transmit Clock for TDM 7.	0,1
PCI_IDSL	Input	**PCI IDSL**.	2,3,4
GE2_TCK	Output	**Ethernet 2 Transmit Clock**	5,6

Table 3-10. UTOPIA Signals (Continued)

Signal Name	Type	Description	I/O Mode
UTP_RMOD	Input	**Receive Word Modulo**	7
TDM7TSYN	Input/ Output	**TDM7 Transmit Frame Sync** Transmit frame sync for TDM 7.	0,1
PCI_AD4	Input/ Output	**PCI Address/Data Line 4** Part of the PCI address/data bus.	2,3,4
UTP_STA	Output	**Transmit Start-of-Packet**	7
TDM7RDAT	Input/ Output	**TDM7 Serial Receiver Data** The receive data signal for TDM 7. As an input, this can be the DATA_A data signal for TDM 7.	0,1
PCI_AD1	Input/ Output	**PCI Address/Data Line 1** Part of the PCI address/data bus.	2,3,4
GE2_RD3	Input	**Ethernet 2 Receive Data 3**	5,6
UTP_RVL	Input	**Receive Data Valid**	7
TDM7RCLK	Input/ Output	**TDM7 Receive Clock** The receive clock signal for TDM 7. As an output, this can be the DATA_C data signal for TDM 7.	0,1
PCI_AD0	Input/ Output	**PCI Address/Data Line 0** Part of the PCI address/data bus.	2,3,4
GE2_RD2	Input	**Ethernet 2 Receive Data 2**	5,6
UTP_TER	Output	**Transmit Error**	7
TDM7RSYN	Input/ Output	**TDM7 Receive Frame Sync** The receive sync signal for TDM 7. As an input, this can be the DATA_B data signal for TDM 7.	0,1
PCI_AD2	Input/ Output	**PCI Address/Data Line 2** Part of the PCI address/data bus.	2,3,4
GE2_TD2	Output	**Ethernet 2 Transmit Data 2**	5,6

3.9 Serial Peripheral Interface (SPI) Signal Summary

Table 3-11 summarizes the Serial Peripheral Interface (SPI) signal lines, which are available in all modes.

Table 3-11. SPI Signals

Signal Name	Type	Signal Description
GPIO24	Input/ Output	**General-Purpose Input Output 24** One of 32 GPIOs. For details, see **Chapter 25**, *GPIO*. Valid in all modes.
$\overline{\text{IRQ6}}$	Input	**Interrupt Request 6** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
$\overline{\text{SPISEL}}$	Input	**SPI Select** Enable input to the SPI slave in single master mode. In multi-master environment, $\overline{\text{SPISEL}}$ detects an error when more one master is operating. Assertion of an $\overline{\text{SPISEL}}$, while it is master, causes an error.
GPIO23	Input/ Output	**General-Purpose Input Output 23** One of 32 GPIOs. For details, see **Chapter 25**, *GPIO*. Valid in all modes
$\overline{\text{IRQ5}}$	Input	**Interrupt Request 5** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
SPIMISO	Input/ Output	**SPI Master Input Slave Output** When the SPI is a master, SPICLK is the clock input signal that shifts received data in from SPIMOSI and transmitted data out through SPIMISO.
GPIO22	Input/ Output	**General-Purpose Input Output 22** One of 32 GPIOs. For details, see **Chapter 25**, *GPIO*. Valid in all modes.
$\overline{\text{IRQ4}}$	Input	**Interrupt Request 4** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
SPIMOSI	Input/ Output	**SPI Master Output Slave Input** When the SPI is a master, SPICLK is the clock input signal that shifts received data in from SPIMOSI and transmitted data out through SPIMISO.
GPIO21	Input/ Output	**General-Purpose Input Output 21** One of 32 GPIOs. For details, see **Chapter 25**, *GPIO*. Valid in all modes.
$\overline{\text{IRQ1}}$	Input	**Interrupt Request 1** One of sixteen external lines that can request a service routine via the internal interrupt controller. see **Chapter 13**, *Interrupt Handling*.
SPICLK	Input/ Output	**SPI Clock** Gated clock, active only during data transfers. Four combinations of SPICLK phase and polarity can be configured. When the SPI is a master, SPICLK is the clock output signal that shifts received data in from SPIMOSI and transmitted data out through SPIMISO.

3.10 GPIO/Maskable Interrupt Signal Summary

Some of the GPIO and interrupt lines are multiplexed with other interfaces. Some of the lines are independent. In addition to the hardware interrupt inputs, there are also several signal lines used to reroute interrupts between the cores and an external host processor. **Table 3-18** summarizes the GPIO and interrupt signal lines.

Table 3-12. GPIO and Maskable Interrupt Summary

Signal Name	Type	Description	I/O Mode
GPIO31	Input/ Output	**General-Purpose Input Output 31** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,5,6
$\overline{\text{IRQ3}}$	Input	**Interrupt Request 3** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.	0,1,5,6
$\overline{\text{PCI_DEVSEL}}$	Input/ Output	**PCI Device Select** For details, see **Chapter 15**, *PCI*.	2,3,4
GPIO30	Input/ Output	**General-Purpose Input Output 30** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,5,6
$\overline{\text{IRQ2}}$	Input	**Interrupt Request 2** One of the sixteen external lines that can request a service routine via the internal interrupt controller, from the SC3400 core. For details, see **Chapter 13**, *Interrupt Handling*.	0,1,5,6
$\overline{\text{PCI_STOP}}$	Input/ Output	**PCI Stop** For details, see **Chapter 15**, *PCI*.	2,3,4
GPIO29	Input/ Output	**General-Purpose Input Output 29** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,5,6
$\overline{\text{IRQ7}}$	Input	**Interrupt Request 7** One of the sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.	0,1,5,6
$\overline{\text{PCI_GNT}}$	Input	**PCI Bus Grant** For details, see **Chapter 15**, *PCI*.	2,3,4
GPIO28	Input/ Output	**General-Purpose Input Output 28** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
TDM5RCLK	Input/ Output	**TDM5 Receive Clock** Receive clock for TDM 5. See **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
PCI_AD13	Input/ Output	**PCI Address/Data Line 13** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO27	Input/ Output	**General-Purpose Input Output 27** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes
SDA	Input/ Output	**I^2C-Bus Data Line** This is the data line for the I^2C bus.	All modes

Table 3-12. GPIO and Maskable Interrupt Summary (Continued)

Signal Name	Type	Description	I/O Mode
GPIO26	Input/ Output	**General-Purpose Input/Output 26** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes
SCL	Input/ Output	**I²C-Bus Clock Line** This the clock line for the I²C bus.	All modes
GPIO25	Input/ Output	**General-Purpose Input Output 25** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,5
$\overline{\text{IRQ15}}$	Input	**Interrupt Request 15** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.	0,5
PCI_AD6	Input/ Output	**PCI Address/Data Line 6** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
GE1_RX_ER	Input	**Ethernet 1 Receive Data Error** In MII and RMII modes indicates that a receive data error occurred.	1,6
GPIO24	Input/ Output	**General-Purpose Input Output 24** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes
$\overline{\text{IRQ6}}$	Input	**Interrupt Request 6** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.	All modes
GPIO23	Input/ Output	**General-Purpose Input Output 23** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes
$\overline{\text{IRQ5}}$	Input	**Interrupt Request 5** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.	All modes
GPIO22	Input/ Output	**General-Purpose Input Output 22** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes
$\overline{\text{IRQ4}}$	Input	**Interrupt Request 4** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.	All modes
GPIO21	Input/ Output	**General-Purpose Input Output 21** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes
$\overline{\text{IRQ1}}$	Input	**Interrupt Request 1** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.	All modes
GPIO20	Input/ Output	**General-Purpose Input Output 20** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,3,5,6
TMR4	Input/ Output	**Timer 4** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For timer functional details, see **Chapter 22**, *Timers*.	0,1,2,3,5,6
PCI_PAR	Input/ Output	**PCI Parity** For details, see **Chapter 15**, *PCI*.	4
UTP_REOP	Input	**Receive End-of-Packet**	7

Table 3-12. GPIO and Maskable Interrupt Summary (Continued)

Signal Name	Type	Description	I/O Mode
GPIO19	Input/ Output	**General-Purpose Input Output 19** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,3,5,6
TMR3	Input/ Output	**Timer 3** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For timer functional details, see **Chapter 22**, *Timers*.	0,1,2,3,5,6
$\overline{\text{PCI_IRDY}}$	Input/ Output	**PCI Ready** For details, see **Chapter 15**, *PCI*.	4
UTP_TEOP	Output	**Transmit End-of-Packet**	7
GPIO18	Input/ Output	**General-Purpose Input Output 18** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,3,5,6
TMR2	Input/ Output	**Timer 2** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For timer functional details, see **Chapter 22**, *Timers*.	0,1,2,3,5,6
$\overline{\text{PCI_FRAME}}$	Input/ Output	**PCI Frame Sync** For details, see **Chapter 15**, *PCI*.	4
UTP_SRP	Input	**Receive Start-of-Packet**	7
GPIO17	Input/ Output	**General-Purpose Input Output 17** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	2
TMR1	Input/ Output	**Timer 1** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For timer functional details, see **Chapter 22**, *Timers*.	2
UTP_IR	Input	**UTOPIA IR** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,5,6,7
$\overline{\text{PCI_CBE3}}$	Input/ Output	**PCI Byte Enable 3** For details, see **Chapter 15**, *PCI*.	4
GPIO16	Input/ Output	**General-Purpose Input Output 16** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes
$\overline{\text{IRQ0}}$	Input	**Interrupt Request 0** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.	All modes
QE_BRGC1	Output	**QUICC Engine Baud-Rate Generator Clock UTOPIA 1** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	All modes
GPIO15	Input/ Output	**General-Purpose Input Output 15** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes
$\overline{\text{IRQ9}}$	Input	**Interrupt Request 9** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.	All modes
UTXD	Input/ Output	**UART Transmit Data** For details, see **Chapter 21, *UART***.	All modes

Table 3-12. GPIO and Maskable Interrupt Summary (Continued)

Signal Name	Type	Description	I/O Mode
GPIO14	Input/ Output	**General-Purpose Input Output 14** One of 32 GPIO pins used as GPIO or as a dedicated input or output. For details, see **Chapter 23, *GPIO***.	All modes
$\overline{\text{RC_LDF}}$	Output	**Reset Configuration Word Load Failure** Used by the core to signal when the reset configuration word fails to load from an I^2C EEPROM when using the boot sequencer. It indicates whether the boot sequencer failed, which can be due to an incorrect data structure or an I^2C bus failure. The signal can be asserted anytime during $\overline{\text{HRESET}}$ assertion. Once asserted, the device holds $\overline{\text{HRESET}}$ low until the $\overline{\text{PORESET}}$ is restarted.	Reset
$\overline{\text{IRQ8}}$	Input	**Interrupt Request 8** One of the sixteen external lines that can request a service routine, via the internal interrupt controller, from the SC3400 cores.	All modes
URXD	Input/ Output	**UART Receive Data** For details, see **Chapter 21**, *UART*.	All modes
GPIO13	Input/ Output	**General-Purpose Input Output 13** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes
TMR0	Input/ Output	**Timer 0** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For timer functional details, see **Chapter 22**, *Timers*.	All modes
QE_BRGC0	Output	**QUICC Engine Baud-Rate Generator Clock UTOPIA 0** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	All modes
GPIO12	Input/ Output	**General-Purpose Input Output 12** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
TDM5TSYN	Input/ Output	**TDM5 Transmit Frame Sync** The transmit sync signal for TDM 5. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD18	Input/ Output	**PCI Address/Data Line 18** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO11	Input/ Output	**General-Purpose Input Output 11** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
TDM5TDAT	Input/ Output	**TDM5 Serial Transmitter Data** The transmit data signal for TDM 5. As an output, this can be the DATA_D data signal for TDM 5. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD17	Input/ Output	**PCI Address/Data Line 17** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO10	Input/ Output	**General-Purpose Input Output 10** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
TDM5RSYN	Input/ Output	**TDM5 Receive Frame Sync** The receive sync signal for TDM 5. As an input, this can be the DATA_B data signal for TDM 5. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD15	Input/ Output	**PCI Address/Data Line 15** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4

Table 3-12. GPIO and Maskable Interrupt Summary (Continued)

Signal Name	Type	Description	I/O Mode
GPIO9	Input/ Output	**General-Purpose Input Output 9** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
TDM5RDAT	Input/ Output	**TDM5 Serial Receiver Data** The receive data signal for TDM 5. As an input, this can be the DATA_A data signal for TDM 5. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD14	Input/ Output	**PCI Address/Data Line 14** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO8	Input/ Output	**General-Purpose Input Output 8** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ14}}$	Input	**Interrupt Request 14** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
TDM6TSYN	Input/ Output	**TDM6 Transmit Frame Sync** The transmit sync signal for TDM 6. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD24	Input/ Output	**PCI Address/Data Line 24** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO7	Input/ Output	**General-Purpose Input Output 7** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ13}}$	Input	**Interrupt Request 13** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
TDM6TDAT	Input/ Output	**TDM6 Transmit Data** The transmit data signal for TDM 6. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD23	Input/ Output	**PCI Address/Data Line 23** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO6	Input/ Output	**General-Purpose Input Output 6** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ12}}$	Input	**Interrupt Request 12** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
TDM6RSYN	Input/ Output	**TDM6 Receive Frame Sync** The receive sync signal for TDM 6. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD21	Input/ Output	**PCI Address/Data Line 21** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4

Table 3-12. GPIO and Maskable Interrupt Summary (Continued)

Signal Name	Type	Description	I/O Mode
GPIO5	Input/ Output	**General-Purpose Input Output 5** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ11}}$	Input	**Interrupt Request 11** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
TDM6RDAT	Input/ Output	**TDM6 Receive Data** The receive data signal for TDM 6. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD20	Input/ Output	**PCI Address/Data Line 20** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO4	Input/ Output	**General-Purpose Input Output 4** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ10}}$	Input	**Interrupt Request 10** One of sixteen external lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
TDM6RCLK	Input/ Output	**TDM6 Receive Clock** The receive clock signal for TDM 6. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD19	Input/ Output	**PCI Address/Data Line 19** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
GPIO3	Input/ Output	**General-Purpose Input Output 3** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes
GPIO2	Input/ Output	**General-Purpose Input Output 2** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes
GPIO1	Input/ Output	**General-Purpose Input Output 1** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes
GPIO0	Input/ Output	**General-Purpose Input Output 0** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes

3.11 Timer Signals

Table 3-13 describes the signals in this group.

Table 3-13. Timer Signals

Signal Name	Type	Description	I/O Mode
TMR4	Input/ Output	**Timer 4** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For timer functional details, see **Chapter 22**, *Timers*.	0,1,2,3,5,6
GPIO20	Input/ Output	**General-Purpose Input Output 20** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,3,5,6
PCI_PAR	Input/ Output	**PCI Parity** For details, see **Chapter 15**, *PCI*.	4
UTP_REOP	Input	**Receive End-of-Packet**	7
TMR3	Input/ Output	**Timer 3** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For timer functional details, see **Chapter 22**, *Timers*.	0,1,2,3,5,6
GPIO19	Input/ Output	**General-Purpose Input Output 19** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,3,5,6
$\overline{\text{PCI_IRDY}}$	Input/ Output	**PCI Ready** For details, see **Chapter 15**, *PCI*.	4
UTP_TEOP	Output	**Transmit End-of-Packet**	7
TMR2	Input/ Output	**Timer 2** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For timer functional details, see **Chapter 22**, *Timers*.	0,1,2,3,5,6
GPIO18	Input/ Output	**General-Purpose Input Output 18** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,3,5,6
$\overline{\text{PCI_FRAME}}$	Input/ Output	**PCI Frame Sync** For details, see **Chapter 15**, *PCI*.	4
UTP_SRP	Input	**Receive Start-of-Packet**	7
TMR1	Input/ Output	**Timer 1** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For timer functional details, see **Chapter 22**, *Timers*. V	2
GPIO17	Input/ Output	**General-Purpose Input Output 17** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	2
UTP_IR	Input	**UTOPIA IR** For details, see **Chapter 18**, *Asynchronous Transfer Mode (ATM) Controller*.	0,1,3,5,6,7
$\overline{\text{PCI_CBE3}}$	Input/ Output	**PCI Byte Enable 3** For details, see **Chapter 15**, *PCI*.	4

Table 3-13. Timer Signals (Continued)

Signal Name	Type	Description	I/O Mode
TMR0	Input/ Output	**Timer 0** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For timer functional details, see **Chapter 22**, *Timers*.	All modes
GPIO13	Input/ Output	**General-Purpose Input Output 13** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes

3.12 UART Signals

Table 3-14. UART Signals

Signal Name	Type	Description	I/O Mode
UTXD	Input/ Output	**UART Transmit Data** Selected through the GPIO configuration. For details, see **Chapter 23, *GPIO***. For functional details, see **Chapter 21, *UART***.	All modes
GPIO15	Input/ Output	**General-Purpose Input Output 15** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	All modes
$\overline{\text{IRQ9}}$	Input	**Interrupt Request 9** One of sixteen external lines that can request a service routine via the internal interrupt controller. Selected through the GPIO configuration. For configuration details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	All modes
URXD	Input/ Output	**UART Receive Data** Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 21**, *UART*.	All modes
$\overline{\text{RC_LDF}}$	Output	**Reset Configuration Word Load Failure** Used by the core to signal when the reset configuration word fails to load from an I^2C EEPROM when using the boot sequencer. It indicates whether the boot sequencer failed, which can be due to an incorrect data structure or an I^2C bus failure. The signal can be asserted anytime during $\overline{\text{HRESET}}$ assertion. Once asserted, the device holds $\overline{\text{HRESET}}$ low until the $\overline{\text{PORESET}}$ is restarted. Valid in reset	Reset
GPIO14	Input/ Output	**General-Purpose Input Output 14** One of 32 GPIO pins used as GPIO or as a dedicated input or output. For details, see **Chapter 23**, *GPIO*. Valid in modes 0–6.	All modes
$\overline{\text{IRQ8}}$	Input	**Interrupt Request 8** One of the sixteen external lines that can request a service routine, via the internal interrupt controller. Selected through the GPIO configuration. For configuration details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	All modes

3.13 I²C Signals

Table 3-15. I²C Signals

Signal Name	Type	Description
SDA	Input/ Output	**I²C-Bus Data Line** This is the data line for the I²C bus. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 25**, *I2C*.
GPIO27	Input/ Output	**General-Purpose Input Output 27** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*. Valid in all modes.
SCL	Input/ Output	**I²C-Bus Clock Line** This the clock line for the I²C bus. Selected through the GPIO configuration. For details, see **Chapter 23**, *GPIO*. For functional details, see **Chapter 25**, *I2C*.
GPIO26	Input/ Output	**General-Purpose Input/Output 26** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*. Valid in all modes.

3.14 TDM Signals

Table 3-16. TDM[7–0] Signals

Signal Name	Type	Description	I/O Mode
TDM7TDAT	Input/ Output	**TDM7 Serial Transmitter Data** The serial transmit data signal for TDM 7. As an output, it provides the DATA_D signal for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
PCI_AD3	Input/ Output	**PCI Address/Data Line 3** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
GE2_TD3	Output	**Ethernet 2 Transmit Data 3** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
UTP_TMD	Output	**Transmit Word Modulo**	7
TDM7TCLK	Input	**TDM7 Transmit Clock** Transmit Clock for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
PCI_IDSL	Input	**PCI IDSL** For details, see **Chapter 15**, *PCI*.	2,3,4
GE2_TCK	Output	**Ethernet 2 Transmit Clock** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
UTP_RER	Input	**Receive Error**	7
TDM7TSYN	Input/ Output	**TDM7 Transmit Frame Sync** Transmit frame sync for TDM 7. See **Chapter 20**, *TDM Interface*.	0,1
PCI_AD4	Input/ Output	**PCI Address/Data Line 4** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
UTP_RMOD	Input	**Receive Word Modulo**	7

Table 3-16. TDM[7–0] Signals (Continued)

Signal Name	Type	Description	I/O Mode
TDM7RDAT	Input/ Output	**TDM7 Serial Receiver Data** The receive data signal for TDM 7. As an input, this can be the DATA_A data signal for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
PCI_AD1	Input/ Output	**PCI Address/Data Line 1** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
GE2_RD3	Input	**Ethernet 2 Receive Data 3** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
UTP_STA	Output	**Transmit Start-of-Packet**	7
TDM7RCLK	Input/ Output	**TDM7 Receive Clock** The receive clock signal for TDM 7. As an output, this can be the DATA_C data signal for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
PCI_AD0	Input/ Output	**PCI Address/Data Line 0** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
GE2_RD2	Input	**Ethernet 2 Receive Data 2** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
UTP_RVL	Input	**Receive Data Valid**	7
TDM7RSYN	Input/ Output	**TDM7 Receive Frame Sync** The receive sync signal for TDM 7. As an input, this can be the DATA_B data signal for TDM 7. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1
PCI_AD2	Input/ Output	**PCI Address/Data Line 2** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	2,3,4
GE2_TD2	Output	**Ethernet 2 Transmit Data 2** For details, see **Chapter 19**, *Ethernet Controller*.	5,6
UTP_TER	Output	**Transmit Error**	7
TDM6TDAT	Input/ Output	**TDM6 Transmit Data** The transmit data signal for TDM 6. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
GPIO7	Input/ Output	**General-Purpose Input Output 7** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ13}}$	Input	**Interrupt Request 13** One of sixteen external lines that can request a service routine via the internal interrupt controller. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
PCI_AD23	Input/ Output	**PCI Address/Data Line 23** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM6TCLK	Input	**TDM6 Transmit Clock** Transmit clock for TDM 6. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD22	Input/ Output	**PCI Address/Data Line 22** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4

Table 3-16. TDM[7–0] Signals (Continued)

Signal Name	Type	Description	I/O Mode
TDM6TSYN	Input/ Output	**TDM6 Transmit Frame Sync** The transmit sync signal for TDM 6. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
GPIO8	Input/ Output	**General-Purpose Input Output 8** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ14}}$	Input	**Interrupt Request 14** One of sixteen external lines that can request a service routine via the internal interrupt controller. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
PCI_AD24	Input/ Output	**PCI Address/Data Line 24** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM6RDAT	Input/ Output	**TDM6 Receive Data** The receive data signal for TDM 6. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
GPIO5	Input/ Output	**General-Purpose Input Output 5** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ11}}$	Input	**Interrupt Request 11** One of sixteen external lines that can request a service routine via the internal interrupt controller. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
PCI_AD20	Input/ Output	**PCI Address/Data Line 20** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM6RCLK	Input/ Output	**TDM6 Receive Clock** The receive clock signal for TDM 6. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
GPIO4	Input/ Output	**General-Purpose Input Output 4** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ10}}$	Input	**Interrupt Request 10** One of sixteen external lines that can request a service routine via the internal interrupt controller. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
PCI_AD19	Input/ Output	**PCI Address/Data Line 19** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4

Table 3-16. TDM[7–0] Signals (Continued)

Signal Name	Type	Description	I/O Mode
TDM6RSYN	Input/ Output	**TDM6 Receive Frame Sync** The receive sync signal for TDM 6. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
GPIO6	Input/ Output	**General-Purpose Input Output 6** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
$\overline{\text{IRQ12}}$	Input	**Interrupt Request 12** One of sixteen external lines that can request a service routine via the internal interrupt controller. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.	0,1,2,5,6
PCI_AD21	Input/ Output	**PCI Address/Data Line 21** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM5TDAT	Input/ Output	**TDM5 Serial Transmitter Data** The transmit data signal for TDM 5. As an output, this can be the DATA_D data signal for TDM 5. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
GPIO11	Input/ Output	**General-Purpose Input Output 11** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
PCI_AD17	Input/ Output	**PCI Address/Data Line 17** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM5TCLK	Input	**TDM5 Transmit Clock** Transmit Clock for TDM 5. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD16	Input/ Output	**PCI Address/Data Line 16** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM5TSYN	Input/ Output	**TDM5 Transmit Frame Sync** The transmit sync signal for TDM 5. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
GPIO12	Input/ Output	**General-Purpose Input Output 12** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
PCI_AD18	Input/ Output	**PCI Address/Data Line 18** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM5RDAT	Input/ Output	**TDM5 Serial Receiver Data** The receive data signal for TDM 5. As an input, this can be the DATA_A data signal for TDM 5. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
GPIO9	Input/ Output	**General-Purpose Input Output 9** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
PCI_AD14	Input/ Output	**PCI Address/Data Line 14** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4

Table 3-16. TDM[7–0] Signals (Continued)

Signal Name	Type	Description	I/O Mode
TDM5RCLK	Input/ Output	**TDM5 Receive Clock** Receive clock for TDM 5. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For functional details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
GPIO28	Input/ Output	**General-Purpose Input Output 28** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
PCI_AD13	Input/ Output	**PCI Address/Data Line 13** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM5RSYN	Input/ Output	**TDM5 Receive Frame Sync** The receive sync signal for TDM 5. As an input, this can be the DATA_B data signal for TDM 5. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For functional details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
GPIO10	Input/ Output	**General-Purpose Input Output 10** One of 32 GPIOs. For details, see **Chapter 23**, *GPIO*.	0,1,2,5,6
PCI_AD15	Input/ Output	**PCI Address/Data Line 15** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM4TDAT	Input/ Output	**TDM4 Serial Transmitter Data** The serial transmit data signal for TDM 4. As an output, it provides the DATA_D signal for TDM 4. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For functional details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD11	Input/ Output	**PCI Address/Data Line 11** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM4TCLK	Input	**TDM4 Transmit Clock** Transmit clock for TDM 4. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD10	Input/ Output	**PCI Address/Data Line 10** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM4TSYN	Input/ Output	**TDM4 Transmit Sync** Transmit sync signal for TDM 4. Selected through the GPIO port; see **Chapter 23**, *GPIO*. For functional details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD12	Input/ Output	**PCI Address/Data Line 12** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM4RDAT	Input/ Output	**TDM4 Serial Receiver Data** The receive data signal for TDM 4. As an input, this can be the DATA_A data signal for TDM 4. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD8	Input/ Output	**PCI Address/Data Line 8** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM4RCLK	Input/ Output	**TDM4 Receive Clock** The receive clock signal for TDM 4. As an output, this can be the DATA_C data signal for TDM 4. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD7	Input/ Output	**PCI Address/Data Line 7** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4

Table 3-16. TDM[7–0] Signals (Continued)

Signal Name	Type	Description	I/O Mode
TDM4RSYN	Input/ Output	**TDM4 Receive Frame Sync** The receive sync signal for TDM 4. As an input, this can be the DATA_B data signal for TDM 4. For configuration details, see **Chapter 20**, *TDM Interface*.	0,1,2,5,6
PCI_AD9	Input/ Output	**PCI Address/Data Line 9** Part of the PCI address/data bus. For details, see **Chapter 15**, *PCI*.	3,4
TDM3TDAT	Input/ Output	**TDM3 Serial Transmitter Data** The serial transmit data signal for TDM 3. As an output, it provides the DATA_D signal for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.	All modes
RC10	Input	**Reset Configuration Word Bit 10** Used to load part of the Reset Configuration Word during Reset.	Reset
TDM3TCLK	Input	**TDM3 Transmit Clock** Transmit Clock for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.	All modes
TDM3TSYN	Input/ Output	**TDM3 Transmit Frame Sync** Transmit frame sync for TDM 3. See **Chapter 20, *TDM Interface***.	All modes
RC11	Input	**Reset Configuration Word Bit 11** Used to load part of the Reset Configuration Word during Reset.	Reset
TDM3RDAT	Input/ Output	**TDM3 Serial Receiver Data** The receive data signal for TDM 3. As an input, this can be the DATA_A data signal for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.	All modes
RC8	Input	**Reset Configuration Word Bit 8** Used to load part of the Reset Configuration Word during Reset.	Reset
TDM3RCLK	Input/ Output	**TDM3 Receive Clock** The receive clock signal for TDM 3. As an output, this can be the DATA_C data signal for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.	All modes
RC16	Input	**Reset Configuration Word Bit 16** Used to load part of the Reset Configuration Word during Reset.	Reset
TDM3RSYN	Input/ Output	**TDM3 Receive Frame Sync** The receive sync signal for TDM 3. As an input, this can be the DATA_B data signal for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.	All modes
RC9	Input	**Reset Configuration Word Bit 9** Used to load part of the Reset Configuration Word during Reset.	Reset
TDM2TDAT	Input/ Output	**TDM2 Serial Transmitter Data** The transmit data signal for TDM 2. As an output, this can be the DATA_D data signal for TDM 2. For configuration details, see **Chapter 20, *TDM Interface***.	All modes
RC6	Input	**Reset Configuration Word Bit 6** Used to load part of the Reset Configuration Word during Reset.	Reset
TDM2TCLK	Input	**TDM 2 Transmit Clock** Transmit clock for TDM 2. For configuration details, see **Chapter 20**, *TDM Interface*.	All modes

Table 3-16. TDM[7–0] Signals (Continued)

Signal Name	Type	Description	I/O Mode
TDM2TSYN	Input/ Output	**TDM2 Transmit frame Sync** Transmit frame sync for TDM 2. For configuration details, see **Chapter 20**, *TDM Interface*.	All modes
RC7	Input	**Reset Configuration Word Bit 7** Used to load part of the Reset Configuration Word during Reset.	Reset
TDM2RDAT	Input/ Output	**TDM2 Serial Receiver Data** The receive data signal for TDM 2. As an input, this can be the DATA_A data signal for TDM 2. For configuration details, see **Chapter 20, *TDM Interface***.	All modes
RC4	Input	**Reset Configuration Word Bit 4** Used to load part of the Reset Configuration Word during Reset.	Reset
TDM2RCLK	Input/ Output	**TDM2 Receive Clock** The receive clock signal for TDM 2. As an input, this can be the DATA_C data signal for TDM 2. For configuration details, see **Chapter 20**, *TDM Interface*.	All modes
TDM2RSYN	Input/ Output	**TDM2 Receive Frame Sync** The receive sync signal for TDM 2. As an input, this can be the DATA_B data signal for TDM 2. For configuration details, see **Chapter 20, *TDM Interface***.	All modes
RC5	Input	**Reset Configuration Word Bit 5** Used to load part of the Reset Configuration Word during Reset.	Reset
TDM1TDAT	Input/ Output	**TDM1 Serial Transmitter Data** The transmit data signal for TDM 1. As an output, this can be the DATA_D data signal for TDM 1. For configuration details, see **Chapter 20, *TDM Interface***.	All modes
RC2	Input	**Reset Configuration Word Bit 2** Used to load part of the Reset Configuration Word during Reset.	Reset
TDM1TCLK	Input	**TDM1 Transmit Clock** Transmit clock for TDM 1. For configuration details, see **Chapter 20**, *TDM Interface*.	All modes
TDM1TSYN	Input/ Output	**TDM1 Transmit Frame Sync** Transmit frame sync for TDM 1. For configuration details, see **Chapter 20, *TDM Interface***.	All modes
RC3	Input	**Reset Configuration Word Bit 3** Used to load part of the Reset Configuration Word during Reset.	Reset
TDM1RDAT	Input/ Output	**TDM1 Serial Receiver Data** The receive data signal for TDM 1. As an input, this can be the DATA_A data signal for TDM 1. For configuration details, see **Chapter 20, *TDM Interface***.	All modes
RC0	Input	**Reset Configuration Word Bit 0** Used to load part of the Reset Configuration Word during Reset.	Reset
TDM1RCLK	Input/ Output	**TDM1 Receive Clock** The receive clock signal for TDM 1. As an input, this can be the DATA_C data signal for TDM 1. For configuration details, see **Chapter 20**, *TDM Interface*.	All modes

Table 3-16. TDM[7–0] Signals (Continued)

Signal Name	Type	Description	I/O Mode
TDM1RSYN	Input/ Output	**TDM1 Receive Frame Sync** The receive sync signal for TDM 1. As an input, this can be the DATA_B data signal for TDM 1. For configuration details, see **Chapter 20, *TDM Interface***.	All modes
RC1	Input	**Reset Configuration Word Bit 1** Used to load part of the Reset Configuration Word during Reset.	Reset
TDM0TDAT	Input/ Output	**TDM0 Serial Transmitter Data** The transmit data signal for TDM 0. As an output, this can be the DATA_D data signal for TDM 0. For configuration details, see **Chapter 20**, *TDM Interface*.	All modes
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.	Reset
TDM0TCLK	Input	**TDM 0 Transmit Clock** Transmit Clock for TDM 0. For configuration details, see **Chapter 20**, *TDM Interface*.	All modes
TDM0TSYN	Input/ Output	**TDM0 Transmit Frame Sync** Transmit Frame Sync for TDM 0. For configuration details, see **Chapter 20, *TDM Interface***.	All modes
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.	Reset
TDM0RDAT	Input/ Output	**TDM0 Serial Receiver Data** The receive data signal for TDM 0. As an input, this can be the DATA_A data signal for TDM 0. For configuration details, see **Chapter 20, *TDM Interface***.	All modes
RCFG_CLKIN_RNG	Input	**Reset Configuration CLKIN Range** This signal is sampled at the deassertion of PORESET to identify the range of the CLKIN input.	Reset
TDM0RCLK	Input/ Output	**TDM0 Receive Clock** The receive clock signal for TDM 0. As an input, this can be the DATA_C data signal for TDM 0. For configuration details, see **Chapter 20**, *TDM Interface*.	All modes
TDM0RSYN	Input/ Output	**TDM0 Receive Frame Sync** The receive sync signal for TDM 0. As an input, this can be the DATA_B data signal for TDM 0. For configuration details, see **Chapter 20, *TDM Interface***.	All modes
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.	Reset

3.15 Other Interrupt Signals

Table 3-17 summarizes the other interrupt signal lines.

Table 3-17. Other Interrupt Signals

Signal Name	Type	Description
$\overline{\text{INT_OUT}}$	Output	**Interrupt Output** An open-drain pin driven from the MSC8144E internal interrupt controller. Assertion of this output indicates that an unmasked interrupt is pending in the MSC8144E internal interrupt controller.
$\overline{\text{NMI}}$	Input	**Non-Maskable Interrupt** External device may assert this line to generate a non-maskable interrupt to the MSC8144E device.
$\overline{\text{NMI_OUT}}$	Output	**Non-Maskable Interrupt Output** An open-drain pin driven from the MSC8144E internal interrupt controller. Assertion of this output indicates that a non-maskable interrupt is pending in the MSC8144E internal interrupt controller, waiting to be handled by an external host.

3.16 OCE Event and JTAG Test Access Port Signals

The MSC8144E uses two sets of debugging signals for the two types of internal debugging modules: OCE and the JTAG TAP controller. Each internal SC3400 core has an OCE module, but they are all accessed externally by the same two signals EE0 and EE1. The MSC8144E supports the standard set of test access port (TAP) signals defined by **IEEE**® Std. 1149.1™ Test Access Port and Boundary-Scan Architecture specification and described in **Table 3-18**.

Table 3-18. JTAG TAP Signals

Signal Name	Type	Signal Description
EE0	Input	**OCE Event Bit 0** Used for putting the internal SC3400 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 SC3400 core is in Debug mode. A high output indicates that at least one SC3400 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 connected to $\overline{\text{PORESET}}$.

Chip-Level Arbitration and Switching System (CLASS) 4

The Chip Level Arbitration and Switching System (CLASS) is the central internal interconnect system for the MSC8144E device. The CLASS is a non-blocking, full-fabric crossbar switch 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 400 MHz, and is separate from the SC3400 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 CLASS initiators are:

- Four SC3400 core subsystems
- L2 ICache
- DMA controller
- TDM module
- Serial RapidIO subsystem
- QUICC Engine module
- PCI controller

The CLASS targets are:

- Four M2 memory spaces
- M3 memory
- DDR controller
- PCI controller
- Configuration Control and Status Registers (CCSR)

Note: The CLASS modules also act as internal initiators and targets (see **Figure 4-2**).

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

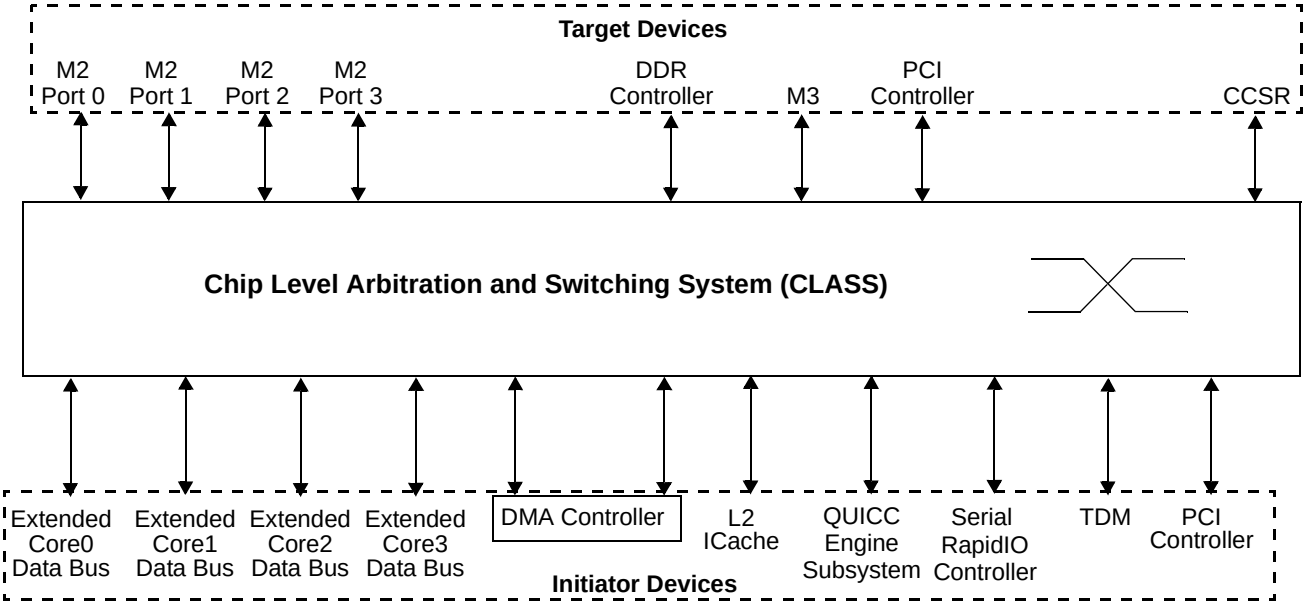


Figure 4-1. CLASS Initiators and Targets in the MSC8144E Device

The CLASS system uses three CLASS modules (CLASS0, CLASS1, and CLASS2) in the MSC8144E device to implement the required interface paths. Each CLASS module provides all the CLASS features.

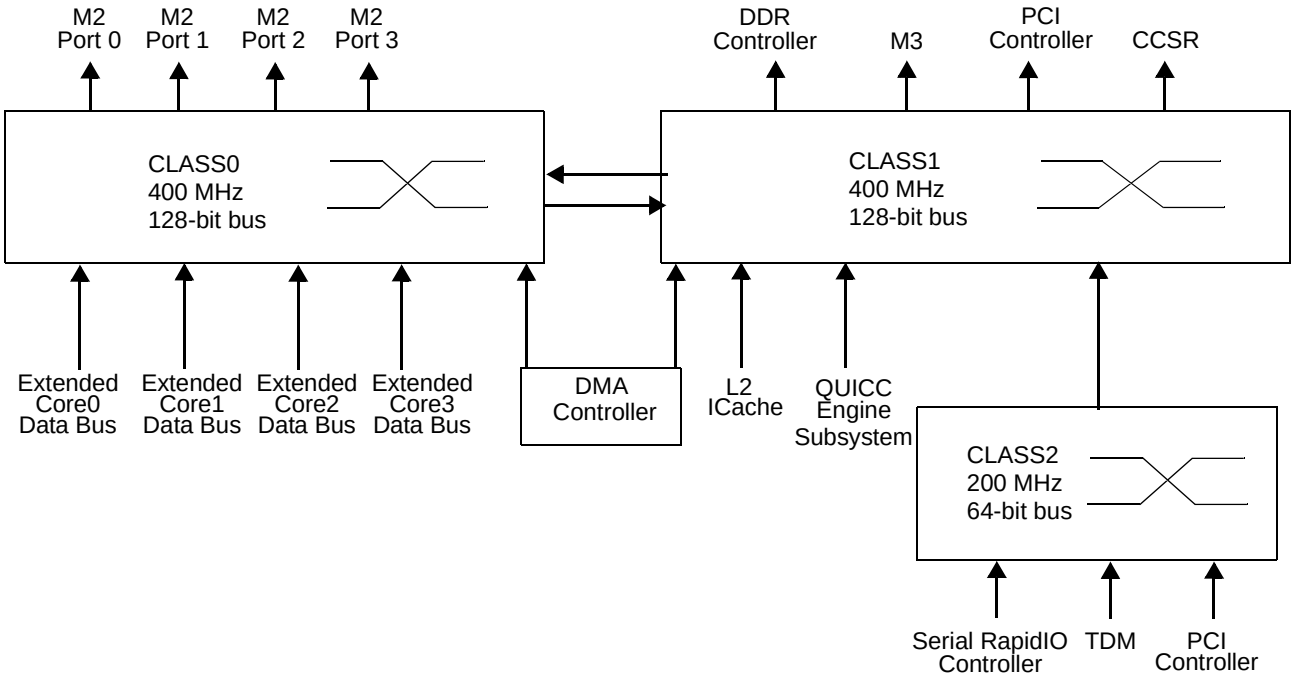


Figure 4-2. CLASS Implementation the MSC8144E Device

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.
- Bank address interleaving is supported for ports 1-4 on CLASS0 (M2 Ports 0–3 are interleaved every 256 bytes, meaning that address bits 9–8 provide access to ports 1–4 of CLASS0)).
- 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.
- Atomic stall unit (ASU) for atomic operations per target (valid for M2 memory only).
- Normalizing mode that splits non-aligned transactions according to the target capabilities (maximal burst size, power-of-2 burst, burst alignment, full size burst, data-beat alignment, wrap size)
- Error detection and handling:
  - The CLASS identifies illegal addresses; addresses that do not belong to any of the address windows or fall inside the negative windows.
  - The CLASS stores the illegal address, reports the error, and generates an interrupt.
- Debug and profiling unit (CDPU) support.

4.2 Functional Description

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

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

The MSC8144E device CLASS modules support different bus widths, numbers of initiator devices, and number of target devices. **Table 4-1** lists the characteristics of the three CLASS modules.

Table 4-1. CLASS Module Parameters

Module	Internal Bus Width	Initiators Supported		Targets Supported	
CLASS0	128-bit	6	DSP core subsystems 0–3, DMA Controller, and CLASS1	5	M2 Ports 0–3 and CLASS1
CLASS1	128-bit	5	L2 ICache, QUICC Engine subsystem, DMA Controller, CLASS0, and CLASS2	5	DDR Controller, M3, PCI Controller, Configuration and Control Registers, and CLASS0
CLASS2	64-bit	3	Serial RapidIO Controller, TDM, PCI Controller	1	CLASS1

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 Atomic Stall Unit (ASU)

The atomic stall unit (ASU) maintains coherency by stalling all read atomic accesses while an atomic operation is open. An atomic operation starts when an initiator performs a read atomic access from a target (and checks to see whether it is appropriate to modify some bits), and ends when the initiator gets the end of transaction signal for write atomic access. The ASU includes an Atomic Open flag which is set when an initiator receives the acknowledge for its read atomic access. While the Atomic Open flag is set, all the read atomic accesses from other initiators are stalled. When the atomic operation ends, the multiplexer clears the Atomic Open flag in the ASU.

Note: Atomic operations are only supported in M2 memory and not in any other memory.

4.2.2.1.1 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.

4.2.2.1.2 Weighted Arbitration

The CLASS arbiter supports limited weighted arbitration, as shown in **Figure 4-3**. 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.

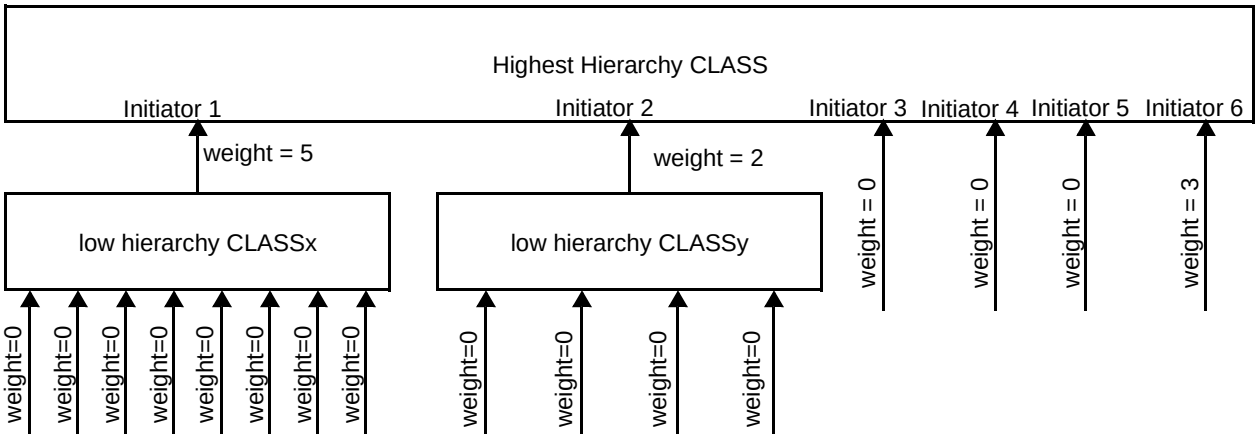


Figure 4-3. Weighted Arbitration Structure (Example)

4.2.2.1.3 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 CnACR (see **Section 4.7.25**, *CLASS Arbitration Control Register (CnACR)*).

4.2.2.1.4 Priority Masking

When CnACR[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.

4.2.2.1.5 Auto Priority Upgrade

This mode is activated by setting the CnPACRx[AUE] bit (see **Section 4.7.4**, *CLASS Priority Auto Upgrade Control Registers (CnPACRx)*). When active, a pending request has its priority upgraded to the next higher priority after a specified number of cycles specified by CnPAVRx[AUV] (see **Section 4.7.3**, *CLASS Priority Auto Upgrade Value Registers (CnPAVRx)*). 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 which 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 will be 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 CnIPCRx and CnTPCRx 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 CnIWPCRx and CnTWPCRx 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 CnPCR.
2. Clear CnIPCRx and CnTPCR.
3. For the time-out mechanism, program CnPTOR and set the CnPCR[TOE] bit.
4. Define the transaction to be monitored by writing the desired configuration to CnWPCR, CnWPACR, CnWPEACR, and CnWPAMR.
5. Enable the watch point units through CnIWPCRx and CnTWPCR.
6. Set the CnWPRCR[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 CnPISR[OVE] bit.
   - In time-out mode, read CnPRCR.
   - If CnPISR[OVE] is set or if CnPRCR is equal to CnPTOR, the results are not valid.
   - Read CnPGCRx 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 4-2** lists the measurement modes, the required configuration settings, and the events measured by the specific CLASS Profiling General Registers for each CLASS module. See **Section 4.7** for the register details.

Table 4-2. CnPGCRx Events Selection

Measurement Mode	Configuration Settings for Each Mode				Events Measured			
	CnWPCR [CE]	CnTPCR [TT]	CnTPCR [PMM]	CnIPCRx [PMM]	CnPGCR0	CnPGCR1	CnPGCR2	CnPGCR3
None selected	0	—	00	00000	—	—	—	—
Initiator Priority and Auto-Upgrade	0	—	00	00001	Number of Initiator requests with Priority 1	Number of Initiator requests with Priority 2	Number of Initiator requests with Priority 3	Initiator Auto-Upgrade

Table 4-2. CnPGCRx Events Selection (Continued)

Measurement Mode	Configuration Settings for Each Mode				Events Measured			
	CnWPCR [CE]	CnTPCR [TT]	CnTPCR [PMM]	CnIPCRx [PMM]	CnPGCR0	CnPGCR1	CnPGCR2	CnPGCR3
Initiator Access Type	0	—	00	00010	Initiator Pending Request	Number of Initiator Read Requests	Number of Initiator Write Requests	Initiator Fast Write
Initiator Stall	0	—	00	00011	Initiator Write After Read	Initiator Write After Read Stall	Initiator Target Switch	Initiator Target Switch Stall
Initiator Priority Upgrade	0	—	00	00100	Initiator Sample 0 Upgrade	Initiator Sample 1 Upgrade	Initiator Any Sample Upgrade	—
Initiator Priority Non-Upgrade	0	—	00	00101	Initiator Sample 0 No Upgrade	Initiator Sample 1 No Upgrade	Initiator Any Sample No Upgrade	—
Initiator Supervisor	0	—	00	00110	Initiator Pending Request	Initiator Supervisor	Initiator Non-Supervisor	—
Initiator Bandwidth	0	—	00	00111	Initiator Read Data Ack.	Initiator Write Data Ack	—	—
Initiator-target Bandwidth	0	—	00	10000 + T	Initiator Target T Read Data Ack	Initiator Target T Write Data Ack	—	—
Arbitration Winner Priority	0	0	01	00000	Target T Win Priority 0	Target T Win Priority 1	Target T Win Priority 2	Target T Win Priority 3
Target Access Splitting	0	1	01	00000	Target T Initiator Access	Target T Target Access	—	—
Arbitration Collision	0	0	10	00000	Number of cycles with more than one request toward Target T (Pending Request)	—	—	—
Target Bandwidth	0	1	10	00000	Target T Read Data Ack	Target T Write Data Ack	—	—
Target Stall	0	—	11	00000	Target T Write After Read	Target T Write After Read Stall	—	—
Watch Point	1	—	00	00000	Watch Point Event	—	—	—

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 **Section 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.
- The CLASS does not allow more than one open atomic access per target. When an atomic access is open toward a particular target, all the other atomic accesses are stalled until the first atomic access toward that target is completed.

Note: Atomic operations are only supported in M2 memory and not in any other memory.

- Arbitration Performance. Only the CLASS1 arbiter has a *wasted cycle* only at transition from high priority to a lower one due to its unique configuration.
- 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.

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 MBus Target Configuration Registers (see page 4-15)
- CLASS Priority Mapping Registers (see page 4-17)
- CLASS Priority Auto Upgrade Value Registers (see page 4-19)
- CLASS Priority Auto Upgrade Control Registers (see page 4-20)
- CLASS 1 Error Address Registers (see page 4-21)
- CLASS 1 Error Extended Address Registers (see page 4-22)
- CLASS Initiator Profiling Configuration Registers (see page 4-24)
- CLASS Initiator Watch Point Control Registers (see page 4-25)
- CLASS Arbitration Weight Registers (see page 4-26)
- CLASS IRQ Status Registers (see page 4-27)
- CLASS IRQ Enable Registers (see page 4-28)
- CLASS Target Profiling Configuration Registers (see page 4-29)
- CLASS Profiling Control Registers (see page 4-31)
- CLASS Watch Point Control Registers (see page 4-32)
- CLASS Watch Point Access Configuration Registers (page 4-34)
- CLASS Watch Point Extended Access Configuration Registers (see page 4-36)
- CLASS Watch Point Address Mask Registers (see page 4-38)
- CLASS Profiling Time Out Registers (see page 4-39)
- CLASS Target Watch Point Control Registers (see page 4-40)
- CLASS Profiling IRQ Status Registers (see page 4-41)
- CLASS Profiling IRQ Enable Registers (see page 4-42)
- CLASS Profiling Reference Counter Registers (see page 4-42)
- CLASS Profiling General Counter Registers (see page 4-43)
- CLASS 2 General Purpose Registers (see page 4-44)
- CLASS Arbitration Control Registers (see page 4-45)
- CLASS1 Start Address Decoder x (see page 4-46)
- CLASS1 End Address Decoder x (see page 4-47)
- CLASS1 Attributes Decoder x (see page 4-48)

Note: The base addresses for addressing registers are as follows:

- CLASS0 = 0xFFF18000
- CLASS1 = 0xFFF19000
- CLASS2 = 0xFFF1A000

4.7.1 CLASS MBus Target Configuration Registers (CnMTCRx)

C0MTCR[0–4] CLASS MBus Target Configuration Registers Offset x*0x020
C1MTCR[0–3]
C2MTCR0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—				IGN	FCE	WCP	OPT	—	BA	FSB	PB	—	MBS		
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

CnMTCRx control the timing of how a transaction is reported as finished by the CLASS normalizer. When all options are disabled, the normalizer works in sampler mode. In addition, this register allows configuration of write confirmation creation. During reset, default values are sampled into the CnMTCR and the bus is disabled. This feature is important because it allows a deep pipeline for most data blocks (fast) while maintaining coherency for the last data in a block (real). **Table 4-3** lists the CnMTCRx bit field descriptions.

Note: The MSC8144E boot program configures these registers to maintain coherency. Reconfiguration by the user is not recommended.

Table 4-3. CnMTCRx Bit Descriptions

Name	Reset	Description	Settings
— 31–28	0	Reserved. Write to 0 for future compatibility.	
IGN 27	0	**Ignore Write Confirmation** Indicates whether to use real confirmation or ignore confirmation (fast confirmation).	These bits are used together as follows: 000 Initiator sends specified confirmation and target uses real confirmation mode. 001 Initiator sends specified confirmation and target uses specified confirmation mode. 010 Initiator sends specified confirmation and target uses fast confirmation mode. 011 Initiator sends specified confirmation and target uses specified confirmation mode. 100 Initiator and target use real confirmation mode. 101 Initiator uses real confirmation mode and initiator uses specified confirmation mode. 110 Initiator and target use fast confirmation mode. 111 Initiator uses fast confirmation mode and target uses real confirmation mode.
FCE 26	0	**Fast Confirmation Enable** Sets the target device confirmation state (real or fast).	
WCP 25	0	**Write with Confirm Propagation Mode** Indicates whether to send the initiator write confirmation. If IGN and FCE are both set, the CLASS module ignores the initiator signal and sends a fast confirmation. **Note:** These bits are not used for atomic accesses and real confirmation mode is used.	
OPT 24	0	**Optimizer Transmit** Determines whether to optimize transmissions. **Note:** This bit is not implemented in CLASS2 and is reserved.	0 Normal. 1 Optimize.

Table 4-3. CnMTCRx Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 23	0	Reserved. Write to 0 for future compatibility.	
BA 22	0	**Burst Alignment** Controls burst alignment.	These bits are used together as follows: 000 No restrictions on burst size or alignment. 001 The byte count in burst access is a power of two, the burst size cannot be bigger than the maximum burst size defined by MBS. 01x Burst size can only be equal to the maximum burst size defined in MBS; there is no alignment restriction. The only other option for access is single bytes. 100 No access crosses the maximum burst size alignment border. 101 The number of datums in the burst access is a power of two and not bigger than the maximum burst size defined by MBS. The address is aligned to the selected burst size, that is, the burst is self-aligned. and will not cross the burst alignment border. 11x Burst size can be equal to the maximum burst size defined in MBS, The burst is aligned to the maximum burst size. The only other option for access is single bytes.
FSB 21	0	**Full Size Burst** Controls full burst size.	
PB 20	0	**Power Burst** Controls power burst (power of two). **Notes:** **1.** The normalizer only generates accesses to the target that meet the burst requirements specified by BA, FSB, and PB values. **2.** BA, FSB, and PB are not implemented for CLASS2 and these bits are reserved in the CLASS2 registers.	
— 19	0	Reserved. Write to 0 for future compatibility.	
MBS 18–16	0	**Max Burst Size** Represents the maximum byte count the target supports, which determines the maximal burst size supported by the target. Other fields in the CnMTCR also use this maximum value. **Note:** The byte count cannot be lower than the datum size, so. for a 128-bit wide data bus, for example, MBS values of 1,2 and 3 use 16 bytes.	000 512 001 4 010 8 011 16 100 32 101 64 110 128 111 256
— 15–13	0	Reserved. Write to 0 for future compatibility.	
DA 12	0	**Datum Alignment** Define how the normalizer segments the access to fit datum limits in the target by controlling the alignment of a generated access when only part of the datum is requested (for example, 3 bytes or 17 bytes). When the target requires a burst request to be aligned to the datum set the DA bit.	0 Partial datum requests permitted. 1 Partial datum requests transferred as single bytes.
— 11	0	Reserved. Write to 0 for future compatibility.	

Table 4-3. CnMTCRx Bit Descriptions (Continued)

Name	Reset	Description	Settings
WS 10–8	0	**Wrap Size** These bits define how the normalizer segments the access when the target expect wrapping access. The normalizer can segment the access to avoid wrapping when these bits are set. **Notes:** **1.** These bits are unimplemented in CLASS 2 and are reserved. **2.** The number of bytes cannot exceed 512 bytes. Therefore, if the transaction uses a data bus width of 128 bits, configuring WS of 5, 6, or 7 effectively selects the value of 32 datums.	000 Disabled. 001 2 datums. 010 4 datums. 011 8 datums. 100 16 datums. 101 32 datums. 110 64 datums. 111 128 datums.
— 7–4	0	Reserved. Write to 0 for future compatibility.	
RC 3–0	0	**Response Control** Because accesses can be segmented and have a few target accesses, the end-of-transmission (EOT) attributes are accumulated. The normalizer has four bits that defined the accumulated EOT attributes. This bit defines the methodology to use to evaluate the EOT accumulation. **Note:** This bit is not implemented in CLASS2 and is reserved.	0 Set the segment bit if the relevant EOT attribute is high. 1 Clear the segment bit if the relevant EOT attribute is low.
Note: When MBS, BA, FSP, PB, WS, and DA are all 0, the normalizer works in sampler mode.			

4.7.2 CLASS Priority Mapping Registers (CnPMRx)

C0PMR[0–5]
C1PMR[0–5]
C2PMR[0–3]

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

CnPMRx 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 4-4 lists the CnPMRx bit field descriptions.

Table 4-4. CnPMRx 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.	
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

4.7.3 CLASS Priority Auto Upgrade Value Registers (CnPAVRx)

C0PAVR[0–5] CLASS Priority Auto Upgrade Value Registers Offset 0x840 + x*0x04
C1PAVR[0–5]
C2PAVR[0–3]

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

CnPAVRx 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 CnPACRx. Therefore, always update the AUV field in the CnPAVRx before you set the AUE bit.

Table 4-5 lists the CnPAVRx bit field descriptions.

Table 4-5. CnPAVRx 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.

4.7.4 CLASS Priority Auto Upgrade Control Registers (CnPACRx)

C0PACR[0–5] CLASS Priority Auto Upgrade Control Registers Offset 0x880 + x*0x04
C1PACR[0–5]
C2PACR[0–3]

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

CnPACRx controls the priority auto-upgrade mechanism.

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

Table 4-6 lists the CnPACRx bit field descriptions.

Table 4-6. CnPACRx 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.

4.7.5 CLASS 1 Error Address Registers (C1EARx)

C1EAR[0–4] CLASS 1 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 X1EARx 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 4-7 lists the C1EARx bit field descriptions.

Table 4-7. 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 one of the 4 cores on the data bus or DMA port 0.
- C1EAR1 = Address generated by DMA port 1.
- C1EAR2 = Address generated by one of the 4 cores on the instruction bus or the L2 ICache.
- C1EAR3 = Address generated by a QUICC Engine peripheral.
- C1EAR4 = Address generated by the RapidIO controller, TDM, or PCI inbound.

4.7.6 CLASS 1 Error Extended Address Registers (C1EEARx)

C1EEAR[0–4] CLASS 1 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	—	AA	—			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 stored 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 4-8 lists the C1EEARx bit field descriptions.

Table 4-8. 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	0	Reserved. Write to 0 for future compatibility.	
AA 12	0	**Atomic Access** This field indicates whether the transaction that caused the error was an atomic access.	0 Not atomic access. 1 Atomic access.
— 11–9	0	Reserved. Write to 0 for future compatibility.	

Table 4-8. C1EEARx 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 0 0x01 Core 1 0x02 Core 2 0x03 Core 3 0x04 L2 ICache Core 0 0x05 L2 ICache Core 1 0x06 L2 ICache Core 2 0x07 L2 ICache Core 3 0x08 DMA Port 0 0x09— 0x10 reserved 0x11 QUICC Engine subsystem 0x12 RapidIO interface 0x13 reserved 0x14 TDM interface 0x15 PCI controller 0x16– 0x17 reserved 0x18 DMA port 1 0x19– 0x1F 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.	

4.7.7 CLASS Initiator Profiling Configuration Registers (CnIPCRx)

C0IPCR[0–5] CLASS Initiator Profiling Configuration Registers‘Offset 0xA00 + x*0x04
C1IPCR[0–5]
C2IPCR[0, 2–3]

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

CnIPCRx controls the CLASS initiator profiling measurements. Each initiator has a dedicated CnIPCR 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 CnIPCR and make sure all the other CnIPCR and the CnTPCR for that CLASS are cleared.

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

Table 4-9 lists the CnIPCRx bit field descriptions.

Table 4-9. CnIPCRx 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. ... 10101 Target 5 bandwidth. 10110–11111 reserved

Table 4-10. Initiator Numbers by CLASS Module

Initiator Number	Initiator Module		
	CLASS0	CLASS1	CLASS2
0	DSP core subsystem 0	CLASS0	Serial RapidIO
1	DSP core subsystem 1	DMA Controller	PCI Controller
2	DSP core subsystem 2	L2 ICache	TDM
3	DSP core subsystem 3	QUICC Engine subsystem	—
4	DMA Controller	CLASS2	—
5	CLASS1	—	—

4.7.8 CLASS Initiator Watch Point Control Registers (CnIWPCRx)

C0IWPCR[0–5] CLASS Initiator Watch Point Control Registers Offset 0xA40 + x*0x04
C1IWPCR[0–5]
C2IWPCR[0–3]

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

CnIWPCRx controls the Watch Point Unit operation for the associated initiator. The Watch Point Unit monitors a specific access defined by the CnWPCRx, CnWPACRx, CnWPEACRx, and CnWPAMR. Each initiator can be enabled/disabled to monitor the selected access. You can write to this register while there are open CLASS transactions.

Note: For each CLASS module, only one WPEN field can be set among all CnIWPCRx and CnTWPCRx when snooping watch point events.

Table 4-11 lists the CnIWPCRx bit field descriptions.

Table 4-11. CnIWPCRx 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.

4.7.9 CLASS Arbitration Weight Registers (CnAWRx)

C0AWR[0–5] CLASS Arbitration Weight Registers Offset 0xA80 + x*0x04
C1AWR[0–5]
C2AWR[0–3]

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

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

The value in CnAWRx 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 4-12 lists the CnAWRx bit field descriptions.

Table 4-12. CnAWRx 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.	

4.7.10 CLASS IRQ Status Register (CnISR)

C0ISR CLASS IRQ Status Register Offset 0xD80
C1ISR
C2ISR

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 CnISR 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 4-13 lists the CnISR bit field descriptions.

Table 4-13. CnISR Bit Descriptions

Name	Reset	Description	Settings
— 31–6	0	Reserved. Write to 0 for future compatibility.	
AEI[5–0] 5–0	0	**Address Error Interrupt 5–0** 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. **Note:** The actual number of supported bits varies depending on the number of initiators supported by the class module: • CLASS0 supports six bits. • CLASS1 supports five bits. • CLASS2 supports three bits. The remaining bits are reserved.	0 No error. 1 Error detected.

4.7.11 CLASS IRQ Enable Register (CnIER)

C0IER CLASS IRQ Enable Register Offset 0xDC0
C1IER
C2IER

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

Table 4-14 lists the CnIER bit field descriptions.

Table 4-14. CnIER Bit Descriptions

Name	Reset	Description	Settings
— 31–6	0	Reserved. Write to 0 for future compatibility.	
AEIE[5–0] 5–0	0	**Address Error Interrupt Enable** Used to enable/disable the address error interrupt for an initiator. **Note:** The actual number of supported bits varies depending on the number of initiators supported by the class module: • CLASS0 supports six bits. • CLASS1 supports five bits. • CLASS2 supports three bits. The remaining bits are reserved.	0 Interrupt masked. 1 Interrupt enabled.

4.7.12 CLASS Target Profiling Configuration Register (CnTPCR)

C0TPCR CLASS Target Profiling Configuration Register Offset 0xE00
C1TPCR
C2TPCR

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

CnTPCR 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 CnIPCRx are cleared.

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

Table 4-15 lists the CnTPCR bit field descriptions.

Table 4-15. CnTPCR Bit Descriptions

Name	Reset	Description	Settings
— 31–13	0	Reserved. Write to 0 for future compatibility.	
TT 12	0	**Target Type** Selects the module used for target profiling. Used with PMM. See PMM settings.	0 Arbiter. 1 Normalizer.
— 11	0	Reserved. Write to 0 for future compatibility.	

Table 4-15. CnTPCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
TN 10–8	0	**Target Number** Indicates the number of selected target.	*For CLASS0:* 000 CLASS1 001 M2 Port 0 010 M2 Port 1 011 M2 Port 2 100 M2 Port 3 101– 111 Reserved *For CLASS1:* 000 Configuration and Status Registers 001 DDR Controller 010 M3 011 PCI Controller 100 CLASS0 101– 111 Reserved *For CLASS2:* 000 CLASS1 001– 111 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.

4.7.13 CLASS Profiling Control Register (CnPCR)

C0PCR CLASS Profiling Control Register Offset 0xE04
C1PCR
C2PCR

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

CnPCR controls the CLASS profiling operation. The register is reset only by a hardware reset. **Table 4-16** lists the CnPCR bit field descriptions.

Table 4-16. CnPCR 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.

4.7.14 CLASS Watch Point Control Registers (CnWPCR)

C0WPCR CLASS Watch Point Control Registers Offset 0xE08
C1WPCR
C2WPCR

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

CnWPCR 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 4-17** lists the CnWPCR bit field descriptions.

Table 4-17. CnWPCR 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 CnWPEACR 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 CnWPEACR enabled.
EATE 14	0	**EOT Attributes Compare Enable** Enables/disables the EOT attributes comparison.	0 EOT attributes compare disabled. 1 EOT attributes compare with CnWPEACR enabled.
ATE 13	0	**Attributes Compare Enable** Enables/disables the attributes comparison.	0 Attributes compare disabled. 1 Attributes compare with CnWPEACR enabled.
SIE 12	0	**Source ID Compare Enable** Enables/disables the source ID comparison.	0 Source ID compare disabled. 1 Source ID compare with CnWPEACR enabled.
PRE 11	0	**Priority Level Compare Enable** Enables/disables the priority level comparison.	0 Priority level compare disabled. 1 Priority level compare with CnWPEACR 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 CnWPEACR 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 CnWPACR 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 CnWPACR enabled.

Table 4-17. CnWPCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
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 CnWPACR enabled.
SNE 6	0	**Snoop Compare Enable** Enables/disables the snoop type comparison.	0 Snoop type compare disabled. 1 Snoop type compare with CnWPACR enabled.
OPE 5	0	**Optimize Compare Enable** Enables/disables the optimize type comparison.	0 Optimize type compare disabled. 1 Optimize type compare with CnWPACR enabled.
TSTE 4	0	**Test Access Compare Enable** Enables/disables the test type comparison.	0 Test type compare disabled. 1 Test type compare with CnWRACR enabled.
SPVE 3	0	**Supervisor Access Compare Enable** Enables/disables supervisor access comparison.	0 Supervisor type compare disabled. 1 Supervisor type compare with CnWRACR 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 CnWPACR enabled.
AE 1	0	**Address Compare Enable** Enables/disables comparison of the access address.	0 Address compare disabled. 1 Address compare with CnWPACR 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.

4.7.15 CLASS Watch Point Access Configuration Register (CnWPACR)

C0WPACR CLASS Watch Point Access Configuration Registers Offset 0xE08
C1WPACR
C2WPACR

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

CnWPACR, along with CnWPEACR, configures the selected access to monitor. The watch point monitoring occurs only if the respective function is enabled in the CnWPCR. The register is reset only by a hardware reset. **Table 4-18** lists the CnWPACR bit field descriptions.

Table 4-18. CnWPACR 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.

Table 4-18. CnWPACR Bit Descriptions (Continued)

Name	Reset	Description	Settings
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 CnWPAWMR, defines the start and range of the addresses the watch point unit monitors. **Note:** For every bit in CnWPAMR[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.	

4.7.16 CLASS Watch Point Extended Access Configuration Register (CnWPEACR)

C0WPEACR CLASS Watch Point Extended Access Configuration Registers Offset 0xE10
C1WPEACR
C2WPEACR

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

CnWPEACR, along with CnWPACR, configures the selected access to monitor. The watch point monitoring occurs only if the respective function is enabled in the CnWPCR. The register is reset only by a hardware reset. **Table 4-19** lists the CnWPEACR bit field descriptions.

Table 4-19. CnWPEACR 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.	*For CLASS0:* 0x8 M2 port 0 0x9 M2 port 1 0xA M2 port 2 0xB M2 port 3 *For CLASS1:* 0xC M3 controller 0xD DDR controller 0xE CCSR 0xF PCI controller *For CLASS2:* 0x8 M2 port 0 0x9 M2 port 1 0xA M2 port 2 0xB M2 port 3 0xC M3 controller 0xD DDR controller 0xE CCSR 0xF PCI controller

Table 4-19. CnWPEACR Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 23–16	0	Reserved. Write to 0 for future compatibility.	
SI 15–11	0	**Source** Defines the source ID to monitor.	*For CL:ASS0:* 0x00 Core0 0x01 Core1 0x02 Core2 0x03 Core3 0x08 DMA port 0 Other values equal to the initiator source ID point to CLASS1 *For CLASS1:* 0x04 L2 ICache Core0 0x05 L2 ICache Core1 0x06 L2 ICache Core2 0x07 L2 ICache Core3 0x11 QUICC Engine module 0x18 DMA port 1 Other values equal to the initiator source ID point to CLASS2 *For CLASS2:* 0x12 RapidIO interface 0x14 TDM interface 0x15 PCI controller
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.

4.7.17 CLASS Watch Point Address Mask Registers (CnWPAMR)

C0WPAMR CLASS Watch Point Address Mask Registers Offset 0xE14
C1WPAMR
C2WPAMR

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

CnWPAMR controls the address range monitored by the watch point unit. The register is reset only by a hardware reset. **Table 4-20** lists the CnWPAMR bit field descriptions.

Table 4-20. CnWPAMR 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 CnWPACR[ADDR]. **Note:** For every bit in ADDM that is cleared, make sure the corresponding bit is cleared in the CnWPACR.	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.

4.7.18 CLASS Profiling Time-Out Registers (CnPTOR)

C0PTOR CLASS Profiling Time-Out Registers Offset 0xE18
C1PTOR
C2PTOR

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

CnPTOR is used to stop the profiling unit operation. When the CnPRCR reaches the value stored in CnPTOR and CnPCR[TOE] is set, the CLASS clears the CnPCR[PE] bit to disable the profiling unit. When CnPCR[PE] clears, the CLASS stops all profiling counters. The register is reset only by a hardware reset. **Table 4-21** lists the CnPTOR bit field descriptions.

Table 4-21. CnPTOR 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.

4.7.19 CLASS Target Watch Point Control Registers (CnTWPCR)

C0TWPCR CLASS Target Watch Point Control Registers Offset 0xE1C
C1TWPCR
C2TWPCR

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
	—										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 CnTWPCR controls the watch point unit operation for CLASS targets. The watch point unit monitors a specific access defined in CnWPCR, CnWPACR, CnWPEACR, and CnWPAMR. Each target can be enabled/disabled for monitoring the specified access type. The register is reset by a hard reset only.

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

Table 4-13 lists the CnTWPCR bit field descriptions.

Table 4-22. CnTWPCR Bit Descriptions

Name	Reset	Description	Settings
— 31–6	0	Reserved. Write to 0 for future compatibility.	
WPEN[5–0] 5–0	0	**Watch Point Enable 5–0** 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.
Note: The actual number of bits implemented in this register depends on the number of targets supported by the CLASS module. The bits are implements from lsb up to the number of targets – 1. The remaining msbs are not implemented and are reserved.			

The targets of the three CLASS modules are as follows:

- CLASS0
  - Target 1 is M2 port 0
  - Target 2 is M2 port 1
  - Target 3 is M2 port 2
  - Target 4 is M2 port 3

- CLASS1
  - Target 0 is the CCSR
  - Target 1 is the DDR controller
  - Target 2 is the M3 memory
  - Target 3 is the PCI
- CLASS2
  - Target 0 is CLASS1

4.7.20 CLASS Profiling IRQ Status Register (CnPISR)

C0PISR CLASS Profiling IRQ Status Registers Offset 0xE20
C1PISR
C2PISR

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

CnPISR indicates that a watch point event occurred or that the CnPRCR overflowed. An interrupt is generated if the status bit is set and the corresponding bit in CnPIERx 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 only reset by a hardware reset or by setting the appropriate CnCPCR[PE] bit. **Table 4-23** lists the CnPISR bit field descriptions.

Table 4-23. CnPISR 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 CnPRCR overflowed (reached 0xFFFFFFFF) during the last measurement.

4.7.21 CLASS Profiling IRQ Enable Register (CnPIER)

C0PIER CLASS Profiling IRQ Enable Registers Offset 0xE24
C1PIER
C2PIER

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

CnPIER enables/disables the generation of interrupts by the debug profiling unit. You can write to the register at any time. The register is only reset by a hardware reset or by setting the appropriate CnCPCR[PE] bit. **Table 4-24** lists the CnPIER bit field descriptions.

Table 4-24. CnPIER 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.

4.7.22 CLASS Profiling Reference Counter Register (CnPRCR)

C0PRCR CLASS Profiling Reference Counter Registers Offset 0xE40
C1PRCR
C2PRCR

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

CnPRCR 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 CnPRCR stops when the profiling unit is disabled, or when the CnPRCR reaches the value stored in CnPTOR and CnCPCR[TOE] is set, which causes the CLASS to clear the CnCPCR[PE] bit to disable the profiling unit. When CnCPCR[PE] clears, the CLASS stops all profiling counters. The register is reset only by a hardware reset only. **Table 4-25** lists the CnPRCR bit field descriptions.

Table 4-25. CnPRCR Bit Descriptions

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

4.7.23 CLASS Profiling General Counter Registers (CnPGCRx)

C0PGCR[0–3] CLASS Profiling General Counter Registers Offset 0x044 + x*0x04
C1PGCR[0–3]
C2PGCR[0–3]

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

CnPGCRx 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 CnPRCR stops when the profiling unit is disabled, or when the CnPRCR reaches the value stored in CnPTOR and CnCPCR[TOE] is set, which causes the CLASS to clear the CnCPCR[PE] bit to disable the profiling unit. When CnCPCR[PE] clears, the CLASS stops all profiling counters. The register is reset only by a hardware reset. **Table 4-26** lists the CnPGCR bit field descriptions.

Table 4-26. CnPGCR Bit Descriptions

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

4.7.24 CLASS General Purpose Register (CnGPR)

C2GPR CLASS2 General Purpose Register Offset 0xF80

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

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

The C2GPR is used to select the PCI read command type and to control the pipeline in the PCI subsystem. The register is reset by a hard reset only. **Table 4-27** lists the C2GPR bit field descriptions.

Table 4-27. CnGPR Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0x3FC00000	Reserved. Always write the reset value to these bits for future compatibility.	
PMDRD 2	0	**PCI Master Delayed Read Disable** Selects the method of outbound read transactions. When delayed reads are enabled, the PCI controller issues new read transactions only after the previous read is completed on the bus. If set together with the PPE bit, the PCI controller can cascade read transactions to a single stream. The bit can only be set when the PCI controller is working in initiator-only mode.	0 PCI delayed read enabled. 1 PCI delayed read disabled.
PRCS 1	0	**PCI Read Command Select** Selects the PCI read command used for 32-byte PCI outbound read transactions.	0 PCI read line is used. 1 PCI read multiple is used.
PPE 0	0	**PCI Pipeline Enable** Controls the PCI subsystem internal pipeline. PPE can be enabled to improve bus utilization in cases in which the PCI functions as an initiator only (inbound windows disabled and PCICCR[MEM] = 0).	0 PCI subsystem internal pipeline disabled. 1 PCI subsystem internal pipeline enabled.

4.7.25 CLASS Arbitration Control Register (CnACR)

C0ACR CLASS Arbitration Control Registers Offset 0xFC0
C1ACR
C2ACR

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—										LA5	LA4	LA3	LA2	LA1	LA0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CnACR 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 4-27** lists the CnACR bit field descriptions.

Table 4-28. CnACR Bit Descriptions

Name	Reset	Description	Settings
— 31–29	0	Reserved. Write to 0 for future compatibility.	
PME **28**	0	**Priority M1ask Enable** Enables/disables the operation of the priority mask unit for starvation elimination.	0 Priority mask disabled. 1 Priority mask enabled.
— 27–6	0	Reserved. Write to 0 for future compatibility.	
LA[5–0] 5–0	0	**Late Arbitration 5–0** Enables/disables late arbitration mode for the associated arbiter.	0 Late arbitration disabled. 1 Late arbitration enabled.
Note: The actual number of the bits implement in each register depends on the number of targets supported by the CLASS module. Implementation begins with the lsb and continues to the number of targets supported. The remaining bits are not implemented and are reserved.			

The targets of the three CLASS modules are as follows:

- CLASS0
  - Target 1 is M2 port 0
  - Target 2 is M2 port 1
  - Target 3 is M2 port 2
  - Target 4 is M2 port 3
- CLASS1
  - Target 0 is the CCSR
  - Target 1 is the DDR controller
  - Target 2 is the M3 memory

— Target 3 is the PCI

- CLASS2
  — Target 0 is CLASS1

4.7.26 CLASS1 Start Address Decoder x (C1SADx)

C1SAD1 CLASS1 Start Address Decoders Offset 0xC00 +x*0x04
C1SAD2

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	1	1	0	1

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

C1SADx configure the address decoding of CLASS1 toward the DDR controller (C1SAD1) and the M3 memory (C1SAD2). 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 C1SADx.

These registers are reset by a hardware reset only. **Table 4-23** lists the C1SADx bit field descriptions.

Table 4-29. C1SADx Bit Descriptions

Name	Reset	Description
— 31–24	0	Reserved. Write to 0 for future compatibility.
SA[35–12] 23–0	Port 1 = 0x040000 Port 2 = 0x0D0000	**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.

4.7.27 CLASS1 End Address Decoder x (C1EADx)

C1EAD1 CLASS1 End Address Decoders Offset 0xC40 + x*0x04
C1EAD2

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	0	1	0	1
EAD2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EA27	EA26	EA25	EA24	EA23	EA22	EA21	EA20	EA19	EA18	EA17	EA16	EA15	EA14	EA13	EA12
Type	R/W															
Reset																
EAD1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
EAD2	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1	1

C1EADx configure the address decoding of CLASS1 toward the DDR controller (C1EAD1) and the M3 memory (C1EAD2). 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 C1EADx.

These registers are reset by a hardware reset only. **Table 4-23** lists the C1EADx bit field descriptions.

Table 4-30. C1EADx Bit Descriptions

Name	Reset	Description
— 31–24	0	Reserved. Write to 0 for future compatibility.
EA[35–12] 23–0	Port 1 = 0x05FFFF Port 2 = 0x0D09FF	**End Address 35–12** The 24 msb of the end address of the specified port window. The lsbs are all zeros. You must make sure that this value is greater than or equal to the start address for the same window. If the end address is equal to the start address, the window size is 4 Kbytes.

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

4.7.28 CLASS1 Attributes Decoder x (C1ATDx)

C1ATD1 CLASS1 End Address Decoders Offset 0xC80 + X*0x04
C1ATD2

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

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

C1ATDx controls the functionality of each specific decoder for CLASS1 toward the DDR controller (C1ATD1) and the M3 memory (C1ATD2). 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 enabled the specific decoder before the start and end addresses are specified in the associated C1SADx and C1EADx.

These registers are reset by a hardware reset only. **Table 4-23** lists the C1ATDx bit field descriptions.

Table 4-31. C1ATDx Bit Descriptions

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

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

Reset 5

The reset and control signals provide many options for MSC8144E operation by configuring various modes and features during power-on reset. Most of these features are configured by loading reset configuration words to the MSC8144E 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 MSC8144E device.

5.1 Reset Operations

The MSC8144E has several inputs to the reset logic:

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

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

5.1.1 Reset Sources

Table 5-1 describes reset sources.

Table 5-1. Reset Sources

Name	Description
Power-on reset (PORESET)	Input pin. Asserting this pin initiates the power-on reset flow that resets all the device and configures various attributes of the device including its clock modes.
Hard reset (HRESET)	This is a bidirectional I/O pin. The MSC8144E can detect an external assertion of HRESET only if it occurs while the MSC8144E is not asserting reset. During HRESET, SRESET is asserted. HRESET is an open-drain pin.
Soft reset (SRESET)	Bidirectional I/O pin. The MSC8144E can only detect an external assertion of SRESET if it occurs while the MSC8144E is not asserting reset. SRESET is an open-drain pin.
Software watchdog reset	After the MSC8144E watchdog timer counts to zero, a software watchdog reset is generated. The enabled software watchdog event then generates an internal hard reset sequence.
RapidIO reset	When the RapidIO logic asserts the RapidIO hard reset signal, an internal hard reset sequence is generated.
JTAG reset	When JTAG logic asserts the JTAG soft reset signal, an internal soft reset sequence is generated.
Software hard reset	A hard reset sequence can be initialized by writing to a memory mapped register (RCR)
Software soft reset	A soft reset sequence can be initialized by writing to a memory mapped register (RCR)

5.1.2 Reset Actions

The MSC8144E reset control logic determines the cause of reset, synchronizes it if necessary, and resets the appropriate internal hardware. Each reset flow has different impact on the device logic. Power-on reset has the greatest impact, resetting the entire device, including clock logic and error capture registers. Hard reset resets the entire device excluding clock logic and error capture registers, while Soft reset initializes the internal logic while maintaining the system configuration. All reset types generate a reset to the cores. The memory controllers, system protection logic, interrupt controller, and I/O pins are initialized only on hard reset. Soft reset initializes the internal logic while maintaining the system configuration. Asserting external SRESET generates a soft reset to the DSP cores and to the remainder of the device. **Table 5-2** identifies reset actions for each reset source.

Table 5-2. Reset Actions for Each Reset Source

Reset Source	Clocks and PLLs Reset Logic Error Capture Registers	Performance Monitor CLASS (most registers, see Section 4.7, *Programming Model*, on page 4-14 for details)	Reset Configuration Words Loaded	Other Internal Logic	$\overline{\text{HRESET}}$ Driven	$\overline{\text{SRESET}}$ Driven and Soft Reset to Cores
• Power-on reset	Yes	Yes	Yes	Yes	Yes	Yes
• External hard reset • Software watchdog reset • RapidIO reset • Software hard reset	No	Yes	No	Yes	Yes	Yes
• External soft reset • JTAG reset • Software soft reset	No	No	No	Yes	No	Yes

5.1.3 Power-On Reset Flow

Assertion of the external $\overline{\text{PORESET}}$ signal initiates the power-on reset flow. $\overline{\text{PORESET}}$ should be asserted externally for at least 32 input clock cycles after stable external power is applied to the MSC8144E device. When $\overline{\text{PORESET}}$ is deasserted, the MSC8144E starts the configuration process. The MSC8144E asserts $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ throughout the power-on reset sequence, including during configuration. Configuration time varies according to the configuration source and CLKIN frequency. Initially, the reset configuration inputs are sampled to determine the configuration source and the input clock division mode. Next, the MSC8144E starts loading the reset configuration words. When the clock mode values in the reset configuration word low load, the system PLL (PLL0) begins to lock. When the system PLL (PLL0) is locked, the clock unit starts distributing clock signals in the device. When all clocks are locked and the reset configuration words are loaded, $\overline{\text{HRESET}}$ is released. $\overline{\text{SRESET}}$ is released sixteen clocks later.

Note: The $\overline{\text{M3_RESET}}$ signal should use the $\overline{\text{PORESET}}$ signal timing. External reset logic should deassert $\overline{\text{M3_RESET}}$ and $\overline{\text{PORESET}}$ together. However, note that the $\overline{\text{M3_RESET}}$ signal must be pulled up to $V_{CCMM3IO}$ (2.5 V).

5.1.4 Detailed Power-On Reset Flow

The detailed power-on reset ($\overline{\text{PORESET}}$) flow for the MSC8144E is as follows:

1. Power is applied to meet the specifications in the *MSC8144E Data Sheet*.
2. $\overline{\text{PORESET}}$ (and optionally $\overline{\text{HRESET}}$) are asserted 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).
3. Apply a stable CLKIN signal and stable reset configuration inputs.
4. The device samples the reset configuration input signals, to determine the clock input range and the reset configuration words source.
5. The device starts loading the reset configuration words. Loading time depends on the reset configuration word source.
6. Once Reset Configuration Word Low is loaded, the system PLL (PLL0) begins to lock.
7. The device keeps driving $\overline{\text{HRESET}}$ low until all PLLs are locked and the reset configuration words are loaded.
8. Deassert the optional $\overline{\text{HRESET}}$, if not done earlier. There is no need to assert $\overline{\text{SRESET}}$ when $\overline{\text{HRESET}}$ is asserted.
9. The internal core reset and the remaining logic is deasserted. I/O drivers are enabled.
10. The PCI and RapidIO interfaces are ready to accept external requests, if enabled, and the core boot vector fetch can proceed, if enabled. The MSC8144E is now in its ready state.

Figure 5-1 shows a timing diagram of the power-on reset flow. If the reset flow is stopped when loading from the I^2C, the $\overline{\text{RC_LDF}}$ signal is asserted by the MSC8144E and resumes the power-on reset flow as shown in **Figure 5-2**.

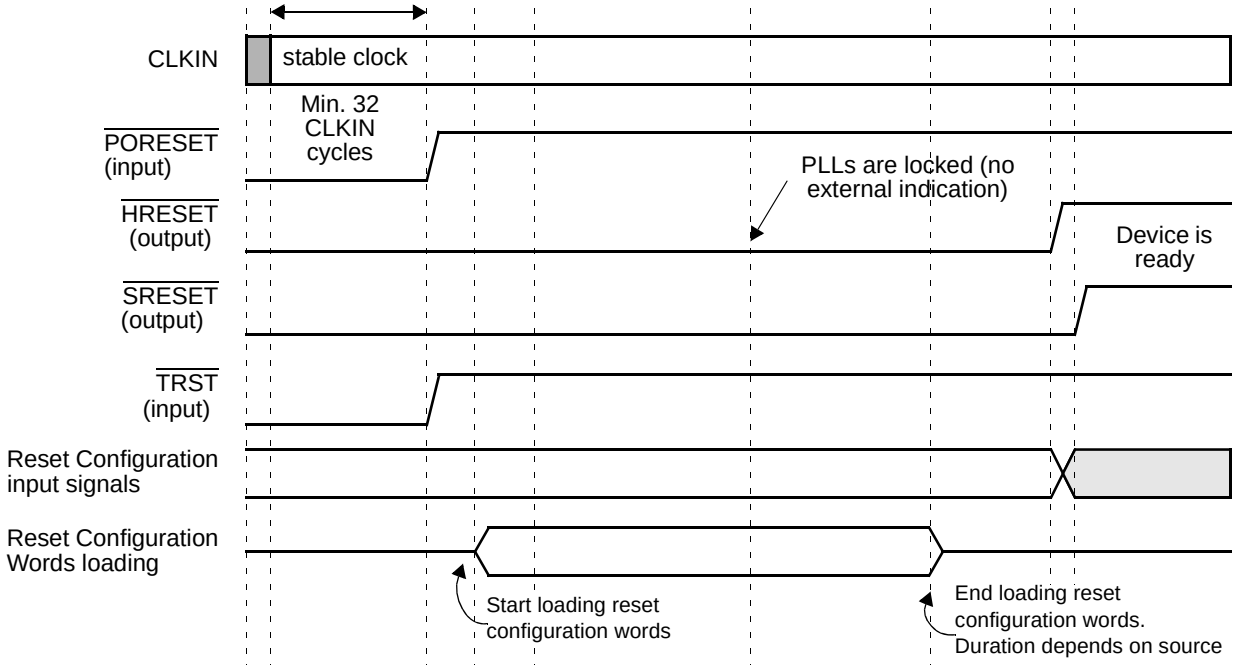


Figure 5-1. Power-On Reset Flow

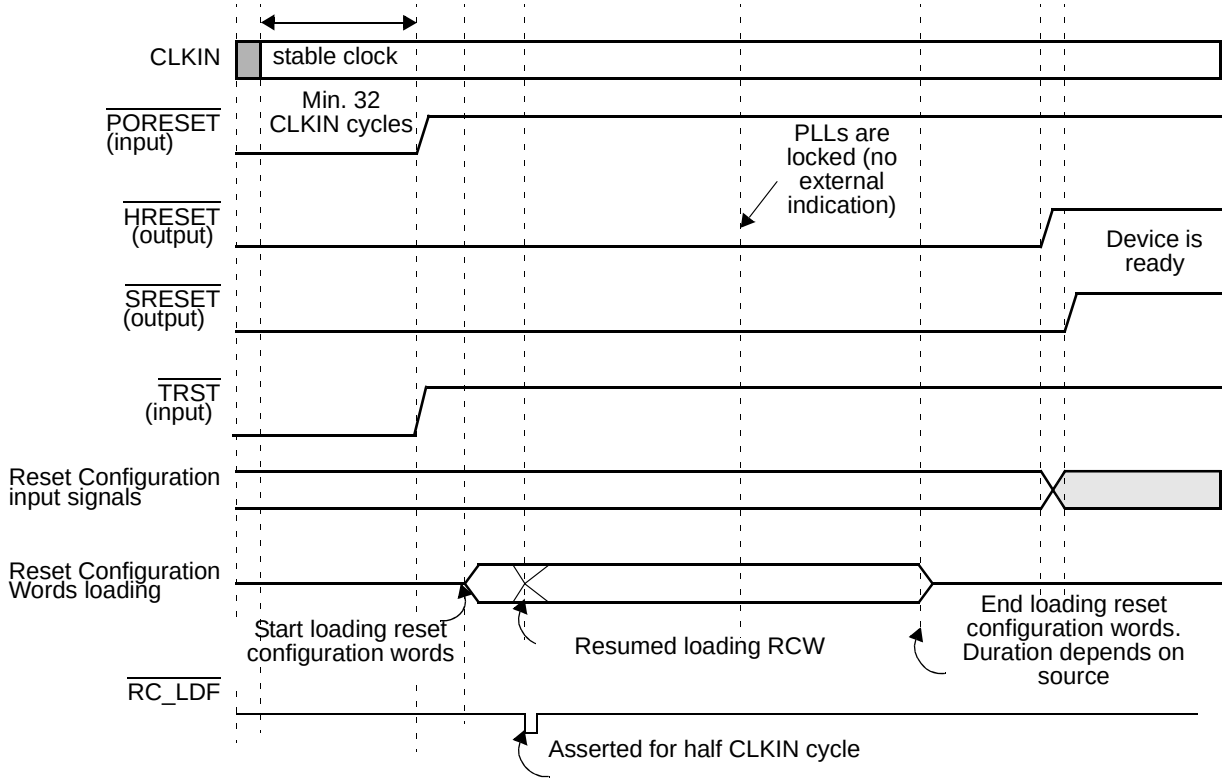


Figure 5-2. Resumed Power-On Reset Flow after $\overline{\text{RC_LDF}}$ Is Asserted

5.1.5 HRESET Flow

The HRESET flow may be initiated externally by asserting HRESET or internally when the MSC8144E detects a reason to assert HRESET. In both cases, the device continues asserting HRESET and SRESET throughout the HRESET flow. The hard reset sequence time varies according to the configuration source and CLKIN frequency. The reset configuration source, the reset configuration words, and the input clock division mode are not affected by hard reset so the MSC8144E immediately configures the device. After the configuration sequence completes, the MSC8144E releases both HRESET and SRESET signals and exits the HRESET flow. Use an external pull-up resistor to deassert the signals. After deassertion is detected, the device waits for a 16-cycle period before testing the presence of an external (hard/soft) reset. **Figure 5-3** shows a timing diagram of the hard reset flow.

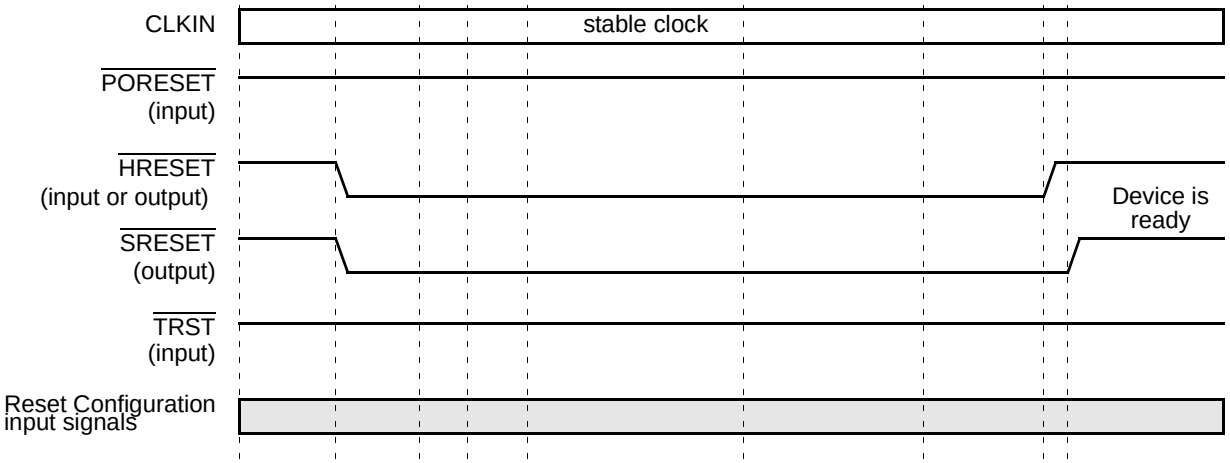


Figure 5-3. Hard Reset Flow

Note: Because the MSC8144E does not sample the reset configuration signals during a hard reset flow, changing the levels of these signals from the values samples during a HRESET sequence has no effect.

5.1.6 SRESET Flow

The SRESET flow is initiated externally by asserting SRESET or internally when the MSC8144E detects a cause to assert SRESET. In both cases, the MSC8144E asserts SRESET for 512 CLKIN clock cycles, after which the MSC8144E releases SRESET and exits the SRESET state. An external pull-up resistor should be used to deassert SRESET; after deassertion is detected, the device waits for a 16-cycle period before testing for the presence of an external (hard/soft) reset. When SRESET is asserted, internal hardware is reset, but the hard reset configuration does not change.

5.2 Reset Configuration

The MSC8144E is initialized using two complementary methods. Initially, a small number of input signals are sampled during the assertion of $\overline{\text{PORESET}}$. These signals determine whether a reset configuration word is required and the device source interface from which it will be loaded. Depending on the configuration signal values, the MSC8144E 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 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 *MSC8144E 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 5.3.3**. They are used to configure the device operation.

5.2.2 Reset Configuration Words Source

The reset configuration words source (RCW_SRC[0–2]) options permit the MSC8144E 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.

- RCW_SRC = 000. Reserved.
- RCW_SRC = 001. Load from an I^2C EEPROM using a frequency specified by the value of RCFG_CLKIN_RNG (0 is less than or equal to 44 MHz; 1 is a range of 66–100 MHz).
- RCW_SRC = 010. Load from an I2C EEPROM using a frequency specified by the value of RCFG_CLKIN_RNG (0 is less than or equal to 66 MHz; 1 is greater than 100 MHz).
- RCW_SRC = 011. Some bits of the reset configuration word are loaded from external signals (RC[0–16]) and others use the default value.
- RCW_SRC = 100. Hard-coded option 1. The reset configuration is loaded from an internal hard-coded option 1.
- RCW_SRC = 101. Hard-coded option 2. The reset configuration is loaded from an internal hard-coded option 2.
- RCW_SRC = 110. Hard-coded option 3. The reset configuration is loaded from an internal hard-coded option 3.
- RCW_SRC = 111. Hard-coded option 4. The reset configuration is loaded from an internal hard-coded option 4.

Note: The value of the reset configuration signals affects the duration of power-on and hard reset sequences, but the duration of the reset sequence will not exceed 4 ms.

Table 5-3 for input signal details.

5.2.3 CLKIN Frequency Range Signal

The CLKIN frequency range reset configuration input (CFG_CLKIN_RNG) indicates whether the CLKIN is less than or more than 66 MHz. See **Table 5-3** for input signal details.

5.2.4 Reset Configuration Load Fail Signal

The reset configuration load fail ($\overline{\text{RC_LDF}}$) output is valid only when the reset configuration words are loaded from an I^2C EEPROM using the boot sequencer and indicates that the boot sequencer failed due to an error. The failure can be caused by an incorrect EEPROM data structure or an I^2C bus problem (see **Chapter 24**, *I^2C* for details. This output signal is valid while $\overline{\text{HRESET}}$ is asserted. It may be asserted any time between deassertion of $\overline{\text{PORESET}}$ and deassertion of $\overline{\text{HRESET}}$. If a failure occurs, the MSC8144E does not deassert $\overline{\text{HRESET}}$ and stays in the reset state. The MSC8144E resumes reading the RCW; the flow may repeat itself until no error id is detected. This signal can help debug reset issues.

5.2.5 Selecting Reset Configuration Input Signals

Table 5-3 shows how to pull down (0) or pull up (1) the reset configuration input signals for various configurations. The reset sequence duration is measured from the deassertion of $\overline{\text{PORESET}}$ to the deassertion of $\overline{\text{SRESET}}$.

Note: When loading the RCW from I^2C, RCWHR[ER] must be set (1).

Table 5-3. Selecting Reset Configuration Input Signals

Configuration Words on I^2C EEPROM	CLKIN Frequency	CFG_CLKIN_RNG, RCW_SRC[0–2]	Reset Sequence Duration in CLKIN Cycles	Duration in µs
No	33 MHz	0, 011–111 (not I^2C EEPROM)	15385	466
No	67 MHz	1, 011–111 (not I^2C EEPROM)	34841	520
Yes	33 MHz	0, 001 (I^2C EEPROM, low clock in frequency)	92561	2805
Yes	66 MHz	0, 010 (I^2C EEPROM, mid clock in frequency)	107451	1628
Yes	67 MHz	1, 001 (I^2C EEPROM, high clock in frequency)	124224	1854
Yes	133 MHz	1, 010 (I^2C EEPROM, high clock in frequency)	157896	1187

Notes:
1. Do not extend the external assertion of the $\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$ signals to the input pins for more than 0.36 ms after the deassertion of the MSC8144 $\overline{\text{PORESET}}$ signal. If the timing of the $\overline{\text{PORESET}}$ circuits is not accurate enough to guarantee this limitation, do not extend the assertion of $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ beyond the deassertion of $\overline{\text{PORESET}}$.
2. When loading the RCW from I^2C, the duration values assume STOP_BS is held low during the $\overline{\text{PORESET}}$ sequence and that SDA is not stuck.

5.2.6 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 MSC8144E decides which interface is used according to reset configuration input signals, as described in **Section 5.2.1**.

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

- Reset Configuration Word Low Register (RCWLR)
- Reset Configuration Word High Register (RCWHR)
- Reset Status Register (RSR)

Note: See **Section 5.3** for register details.

5.2.7 Loading The Reset Configuration Words

The MSC8144E 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 5.2.1**.

5.2.7.1 Loading From an I^2C EEPROM

When a MSC8144E is configured by the reset input signals to load the reset configuration words from an EEPROM via the I^2C interface (RCW_SRC[0–2] = 001 or 010), it uses the I^2C unit boot sequencer in a special mode. In this mode, the I^2C 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).

5.2.7.1.1 Using The Boot Sequencer For Reset Configuration

Note: For detailed description about the I^2C interface and the boot sequencer, refer to **Chapter 24**, *I^2C*.

When used to load the reset configuration words, the I^2C 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 I^2C module enters its reset state until $\overline{\text{HRESET}}$ is deasserted. There should be no other I^2C traffic when the boot sequencer is active. After $\overline{\text{HRESET}}$ is deasserted, the boot sequencer mode is disabled.

Note: I^2C SCL clock stretching is not allowed during the reset sequence. The SCL signal should have minimum rise/fall times.

5.2.7.1.2 EEPROM Addressing

A reset initiator MSC8144E is selected by holding the STOP_BS signal low during the power-on reset flow. The reset initiator 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 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 5.2.7.2**, *Loading Multiple Devices From a Single I^2C EEPROM*, on page 5-11.

5.2.7.1.3 EEPROM Data Format In Reset Configuration Mode

The I^2C 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 I^2C 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 6.5.2.3**, *Boot File Format*, on page 6-17. 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 I^2C bus error is detected, the device stops processing and remains in a hard reset state with $\overline{\text{HRESET}}$ asserted. The MSC8144E drives $\overline{\text{RC_LDF}}$ low (asserted) for half a CLKIN cycle to indicate an error. Then the MSC8144E resumes reading the RCW. This flow may repeat until no error id is detected.

5.2.7.2 Loading Multiple Devices From a Single I²C EEPROM

When the MSC8144E device shares the I²C EEPROM device with other MSC8144E devices to load the reset configuration words, one device must be a reset initiator and the rest must be reset targets. The definition of reset target or reset initiator is latched internally during power-on reset sequence. The reset configuration implementation involves a software and glue logic. The hardware connection is shown in **Figure 5-4**. The STOP_BS signal input to the reset initiator must be driven low during the power on reset sequence while all the targets inputs must be driven high. During the power on reset assertion, the initiator cannot drive the STOP_BS output bus because its role as initiator is not enabled. Pull-ups are required; refer to the *MSC8144E Technical Data sheet* for appropriate resistor values to pull the target STOP_BS signal inputs high.

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

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

In the last stage, the reset initiator releases the STOP_BS for each target in a known order. The released reset target accesses the I²C bus to read from target address 0b1010111. The order of reading the target RCWs is the order for their connection.

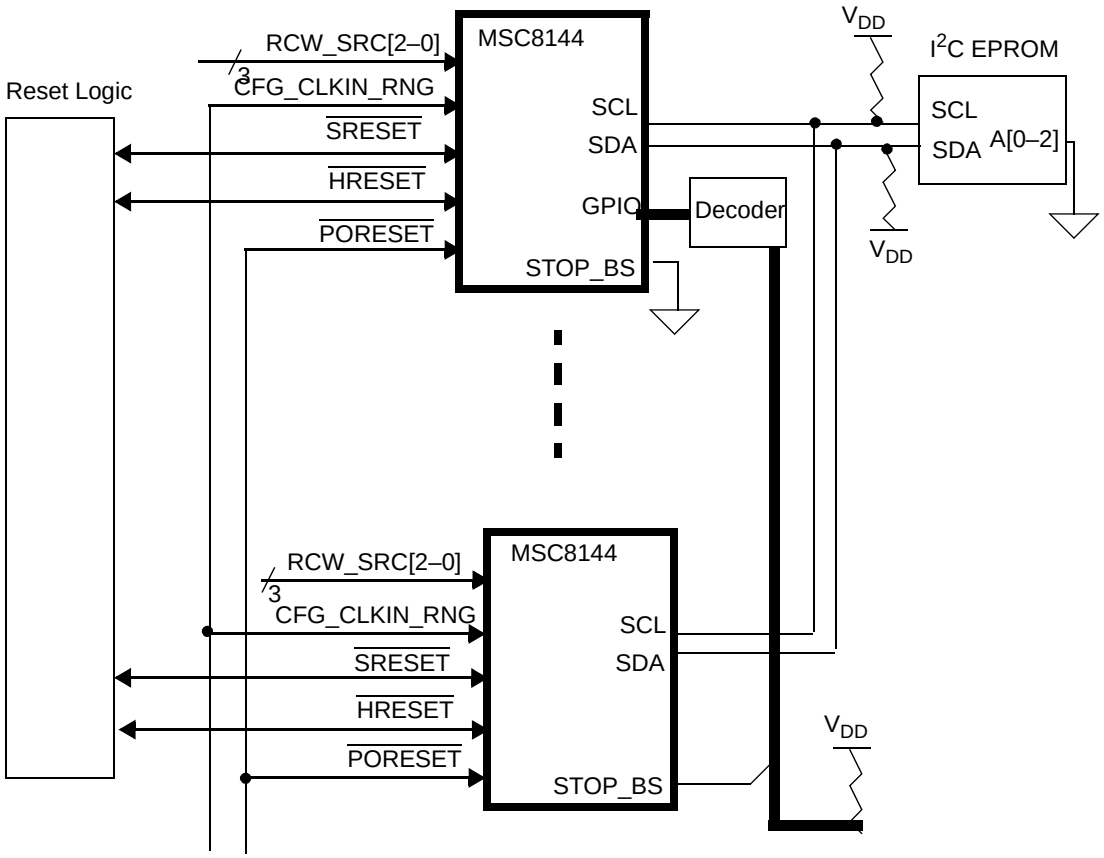


Figure 5-4. Multi Device I^{2}C Reset Configuration Hardware

5.2.7.2.1 Multiple Device External Reset Logic

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.

5.2.7.2.2 Multiple Device ROM Code

The multiple device scheme allows ROM code to support a number of devices loading from the same serial EEPROM target. The routine flow is described in **Section 6.4.1**, *Multi Device Support for the I^{2}C Bus*, on page 6-5.

5.2.7.3 Single Device Loading From I²C EEPROM

The MSC8144E 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 5-5**.

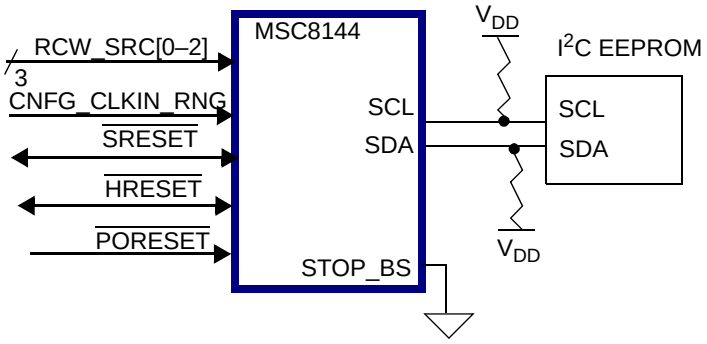


Figure 5-5. Single Device I²C Reset Configuration

5.2.7.4 Loading Reduced RCW From External Pins

When the MSC8144E device is configured to use the reduced RCW (RCW_SRC[2–0] = 011), the MSC8144E latches some bits of the reset configuration word from external pins (RC[0–16]). The other bits of the RCWs are loaded from default hard coded word. The hardware connection is shown in **Figure 5-6**.

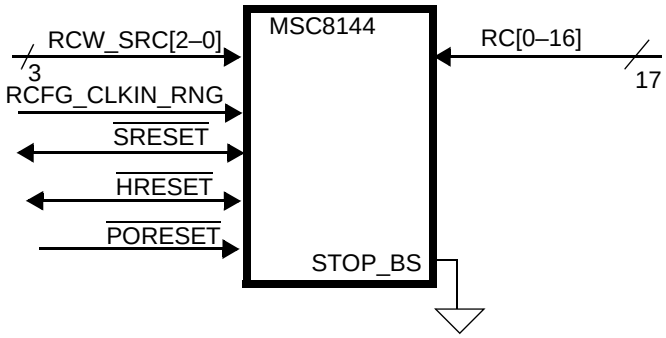


Figure 5-6. External Pins Reduced Reset Configuration

Table 5-2 defines the Hard Coded Reset Configuration Word Low fields values, according to RCW_SRC[0–2].

Table 5-4. Hard Coded Reset Configuration Word Low Field Values

Bits	RCW_SRC[0–2] = 1xx	Meaning
31–30	00	Select clock number 2 for CLK0
29–26	0000	Reserved.
25	0	SerDes digital filter BW is 200 ppm.

Table 5-4. Hard Coded Reset Configuration Word Low Field Values (Continued)

Bits	RCW_SRC[0–2] = 1xx	Meaning
24	0	Reserved.
23	0	RapidIO V_{DD} is 1 V.
22–20	001	RapidIO/SGMII reference clock is 100 MHz, SerDes is 1.25 GHz.
19	1	RapidIO interface is Enabled
18	1	RapidIO interface uses 1x protocol
17	1	SGMII1 is Enabled on SerDes
16	1	SGMII2 is Enabled on SerDes
15–13	000	Reserved.
12	1	Select system PLL (PLL0) for PCI
11	1	Select system PLL (PLL0) for DDR
10	0	Select global PLL (PLL2) for M3
9	0	Reserved.
8	0	Enable global PLL (PLL2)
7	0	Enable core PLL (PLL1)
6	0	Enable system PLL (PLL0)
5–0	001011 for RCW_SRC = 100 001011 for RCW_SRC = 101 110101 for RCW_SRC = 110 110101 for RCW_SRC = 111	MODCK[5–0].

5.2.7.5 Hard Coded Reset Configuration Word High Fields Values

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

Table 5-5. Hard Coded Reset Configuration Word High Fields Values

Bits	RCW_SRC[0:2] = 1xx	Meaning
31	0	Reserved
30	0	No reset targets to be configured
29	0	Watch Dog Timer is disabled.
28-23	000010 for RCW_SRC = 1x0 001000 for RCW_SRC = 1x1	000010 Boot port is PCI 001000 Boot port is I^2C
22	1	Reserved.
21	1	Rapid IO host access enabled.
20-15	011000	RapidIO prescale timer enable. OCeaN clock is 200 MHz.
14	0	Reserved.
13-10	0010 for RCW_SRC = 100 0001 for RCW_SRC = 101 0010 for RCW_SRC = 110 0001 for RCW_SRC = 111	Pin multiplexing.
9-4	000000	Device ID
3	0	No reset extension.

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

Bits	RCW_SRC[0:2] = 1xx	Meaning
2-1	00	No loopback mode on SerDes.
0	1	Common transport type is Large System.

5.2.7.6 External Reset Configuration Word Low

Table 5-5 defines the combined External and Hard Coded Reset Configuration Word Low fields values, according to the value of RCW_SRC[0–2]. For a detailed description of the reset configuration word, see **Section 5.3.1**, *Reset Configuration Word Low Register (RCWLR)*, on page 5-17 and **Section 5.3.2**, *Reset Configuration Word High Register (RCWHR)*, on page 5-19.

Table 5-6. External Reset Configuration Word Low

Bits	RCW_SRC[0:2] = 011	Meaning
31–30	00	Select clock number 2 for CLK0
29–26	0000	Reserved.
25	0	SerDes filter select.
24	0	Reserved.
23	0	RapidIO V_{DD} is 1 V.
22-20	001	RapidIO/SGMII reference clock is 100 MHz, SerDes 1.25 GHz.
19	RC[16] || RC[3]	0 RapidIO disabled on SerDes 1 RapidIO enabled on SerDes
18	RC[16]	0 RapidIO 4x protocol. 1 RapidIO 1x protocol
17	RC[16]	0 Disable SGMII1 on SerDes 1 Enable SGMII1 on SerDes
16	RC[16] && RC[3]	0 Disable SGMII2 on SerDes 1 Enable SGMII2 on SerDes
15–13	000	reserved
12	1	Select system PLL (PLL0) for PCI.
11	1	Select system PLL (PLL0) for DDR
10	0	Select global PLL (PLL2) for M3
9	0	reserved.
8	0	Enable global PLL (PLL2).
7	0	Enable core PLL (PLL1)
6	0	Enable system PLL (PLL0)
5-3	000	MODCK[5–3].
2-0	RC[2–0]	MODCK[2–0].

5.2.7.7 External Reset Configuration Word High Fields Values

Table 5-7 defines the External and Hard Coded Reset Configuration Word High fields values.

Table 5-7. External Reset Configuration Word High Fields Values

Bits	RCW_SRC[0–2] = 011	Meaning
31	0	reserved
30	0	No reset targets to be configured
29	0	Watch Dog Timer is disabled.
28	0	Boot port select [5].
27–26	RC[15–14]	Boot port select [4–3].
25	0	Boot port select[2].
24–23	RC[13–12]	Boot port select[1–0]. See **Table 5-10**.
22	1	Reserved.
21	1	Rapid IO host access enabled.
20–15	01_1000	RapidIO prescale timer enabled. OCeaN clock is 200 MHz.
14	0	Reserved.
13-12	00	Pins multiplexing[3–2].
11–10	RC[11–10]	Pins multiplexing[1–0].
9–4	RC[9–4]	Device ID.
3	0	No reset extension.
2–1	00	No loopback mode on SerDes.
0	1	Common transport type is Large System.

5.3 Reset Programming Model

This section describes the following reset registers in detail:

- Reset Configuration Word Low Register (RCWLR), **page 5-17**.
- Reset Configuration Word High Register (RCWHR), **page 5-19**.
- Reset Status Register (RSR), **page 5-21**.
- Reset Protection Register (RPR), **page 5-23**.
- Reset Control Register (RCR), **page 5-24**.
- Reset Control Enable Register (RCER), **page 5-25**.

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		—				SF	—	RV	SCLK			RIOE	1x	SGMII1	SGMII2
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
				SPCI	SDDR	SM3	—	GPD	CPD	SPD	MODCK					
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The RCWLR is a read-only register set according to the reset configuration word low loaded during the reset flow. **Table 5-8** defines the RCWLR bit fields.

Table 5-8. RCWLR Bit Descriptions

Name	Reset	Description	Settings
CLKO 31–30	0	**CLKOUT Source** This field selects the source for CLKOUT. See **Chapter 7**, *Clocks* for source clock definitions.	00 Source is Clock2. 01 Source is Clock9. 10 Source is Clock10. 11 CLKOUT is always low.
— 29–26	0	Reserved. Write to zero for future compatibility.	
SF 25	0	**SerDes Filter** Selects the SerDes filter.	0 200 ppm SerDes digital filter bandwidth 1 600 ppm SerDes digital filter bandwidth
— 24	0	Reserved. Write to zero for future compatibility.	
RV 23	0	**RapidIO V_{DD} Select**	0 1 V 1 1.2 V

Table 5-8. RCWLR Bit Descriptions (Continued)

Name	Reset	Description	Settings
SCLK **22–20**	0	**SerDes Clock Mode** This selects the SerDes reference clock and frequency.	000 Ref. Clock = 100 MHz, RapidIO = 2.5 GHz, SGMII = 1.25 GHz 001 Ref. Clock = 100 MHz, RapidIO/SGMII = 1.25 GHz 010 Ref. Clock = 125 MHz, RapidIO = 2.5 GHz, SGMII = 1.25 GHz 011 Ref. Clock = 125 MHz, RapidIO/SGMII = 1.25 GHz 100 Ref. Clock = 125 MHz, RapidIO = 3.125 GHz, SGMII is not functional 101 Ref. Clock = 156.25 MHz, RapidIO = 2.5 GHz, SGMII = 1.25 GHz 110 Ref. Clock = 156.25 MHz, RapidIO/SGMII = 1.25 GHz 111 Ref. Clock = 156.25 MHz, RapidIO = 3.125 GHz, SGMII is not functional
RIOE 19	0	**RapidIO Enable** Enables or disables the RapidIO controller.	0 Power is disabled on RapidIO SerDes lanes. 1 Power is enabled on RapidIO SerDes lanes.
1x 18	0	**RapidIO 1x Select**	0 RapidIO 4x mode is selected on SerDes. 1 RapidIO 1x mode is selected on SerDes.
SGMII1 17	0	**SGMII 1 Enable**	0 SGMII 1 is disabled on SerDes. 1 SGMII 1 is enabled on SerDes.
SGMII2 16	0	**SGMII 2 Enable**	0 SGMII 2 is disabled on SerDes. 1 SGMII 2 is enabled on SerDes.
— 15–13	0	Reserved. Write to zero for future compatibility.	
SPCI 12	0	**Select System PLL (PLL0) for PCI Clock**	0 Select global PLL (PLL2) for PCI. 1 Select system PLL (PLL0) for PCI.
SDDR 11	0	**Select System PLL (PLL0) for DDR Clock**	0 Select global PLL (PLL2) for DDR. 1 Select system PLL (PLL0) for DDR.
SM3 10	0	**Select System PLL (PLL0) for M3 Clock**	0 Select global PLL (PLL2) for M3. 1 Select system PLL (PLL0) for M3.
— 9	0	Reserved. Write to zero for future compatibility.	
GPD 8	0	**Global PLL (PLL2) Disable**	0 Enable global PLL (PLL2). 1 Disable global PLL (PLL2).
CPD 7	0	**Core PLL (PLL1) Disable**	0 Enable core PLL (PLL1)s. 1 Disable core PLL (PLL1)s.
SPD 6	0	**System PLL (PLL0) Disable**	0 Enable system PLL (PLL0). 1 Disable system PLL (PLL0).
MODCK 5–0	0	**Clock Mode** Defines the clock operating mode.	See **Chapter 7**, *Clocks*.

5.3.2 Reset Configuration Word High Register (RCWHR)

RCWHR Reset Configuration Word High Register Offset 0x04

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—	RM	EWDT	BPRT						—	RIO	PTE				
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
	PTE		—	PIN_MUX				DEVID					ER	SLP		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 5-9** defines the RCWHR bit fields.

Table 5-9. RCWHR Bit Descriptions

Name	Description	Settings
— 31	Reserved. Write to zero for future compatibility.	
RM 30	**Reset Initiator Configure Targets** This bit must be set for single MSC8144E device loading of the RCW from an I^2C EEPROM and BPRT is I^2C. See **Chapter 6**, *Boot Program*. The number of reset targets is defined externally.	0 Reset target. 1 Reset initiator.
EWDT 29	**Enable Watchdog Timer** Selects the status of the software watchdog when coming out of reset. The user can override this value by writing a 1 to the System Watchdog Control Register (SWCRR[SWEN]) during system initialization.	0 Watchdog timer initially disabled. 1 Watchdog timer initially enabled.
BPRT 28–23	**Boot Port Select** Defines the boot port interface configuration.	See **Table 5-10**.
— 22	Reserved. Write to one for future compatibility.	
RIO 21	**RapidIO Host Access Enable** Enables RapidIO access to internal memory after boot.	0 RapidIO access to internal memory disabled. 1 RapidIO access to internal memory enabled.
PTE 20–15	**RapidIO Prescale Timer Enable** Compute the value using: (OCeaN clock/8 MHz) –1, rounded to the nearest whole value.	
— 14	Reserved. Write to zero for future compatibility.	
PIN_MUX1 13–10	**Pin Multiplexing** Stores the value of the signals sampled during reset. This selects the I/O multiplexing mode.	See **Chapter 3** *External Signals*.
DEVID 9–4	**Device ID** Stores the value of the signals sampled during reset.	00000 Initiator device/Device 0. 00001– 11111 Target device number (from 1 to 31).

Table 5-9. RCWHR Bit Descriptions (Continued)

Name	Description	Settings
ER 3	**Extend Reset** This bit must be set when loading the RCW from I^2C.	0 Normal reset duration. 1 Extended reset duration.
SLP 2–1	**SerDes Loopback**	00 Normal operation 01 Digital loop mode 10 Analog loop mode 11 reserved
CTLS 0	**Common Transport Large System**	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 may 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.

Table 5-10. Boot Port Select

Field Name	Boot Port	Value (Binary)	Description
BPRT	PCI	000000	PCI with no DDR
		000001	PCI using single DDR 32 Mbytes
		000010	PCI (default) - DDR 256 Mbytes
		000011	PCI using single DDR 64 Mbytes
		000100	PCI using single DDR 128 Mbytes
		000101	PCI using DDR 512 Mbytes
		000110	Reserved
		000111	Reserved
	I^2C	001000	I^2C
	RapidIO interface	001001	RapidIO interface without I^2C
		001010	RapidIO interface with I^2C
	SPI	001011	SPI with Flash memory
		001100– 001110	Reserved
	Ethernet1 - No I^2C	001111	SMII
		010000	RMII
		010001	RGMII
		010010	MII
		010011	SGMII
	Ethernet1 and I^2C	010100	SMII
		010101	RMII
		010110	RGMII
		010111	MII
		011000	SGMII (default)
		011001... 111111	Reserved

Note: When SGMII is selected, make sure that RCWLR[1x] is cleared (0).

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
	RSTSRC			—					RIO	SW1	SW2	SW3	—			BSF
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—		SWSR	SWHR	—			JS	—				SW4	SW0	SRS	HRS
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The Reset Status Register accumulates reset events. For example, because software watchdog expiration results in a hard reset, which in turn results in a soft reset, RSR[SWRS], RSR[SRS] and RSR[HRS] are all set after a software watchdog reset. This register returns to its reset value only when power-on reset occurs, but not when a JTAG power-on reset occurs. JTAG reset status bits are not accumulative. **Table 5-11** defines the RSR bit fields.

Table 5-11. RSR Bit Descriptions

Name	Reset	Description	Settings
RSTSRC 31–29	0	**Reset Configuration Word Source** Stores the value of the RCW_SRC[0–2] signals sampled during reset. See **5.2.2**, *Reset Configuration Words Source*.	000 Reserved. 001 I^2C EEPROM with CLKIN less than 44 MHz or from 66–100 MHz 010 I^2C EEPROM with CLKIN of 44–66 MHz or more than 100 MHz. 011 Input pins and default settings. 100 Hard coded option 1. 101 Hard coded option 2. 110 Hard coded option 3. 111 Hard coded option 4.
— 28–24	0	Reserved. Write to zero for future compatibility.	
RIO 23	0	**Hard Reset from RapidIO** Indicates whether the RapidIO interface generated a hard reset.	0 No hard reset. 1 Hard reset generated by the RapidIO interface.
SW1 22	0	**Software Watchdog Timer 1** Indicates whether watchdog timer 1 expired.	0 Software watchdog timer 1 not expired. 1 Software watchdog timer 1 expired.
SW2 21	0	**Software Watchdog Timer 2** Indicates whether watchdog timer 2 expired.	0 Software watchdog timer 2 not expired. 1 Software watchdog timer 2 expired.
SW3 20	0	**Software Watchdog Timer 3** Indicates whether watchdog timer 3 expired.	0 Software watchdog timer 3 not expired. 1 Software watchdog timer 3 expired.
— 19–17	0	Reserved. Write to zero for future compatibility.	
BSF 16	0	**Boot Sequencer Fail** Indicates whether the I^2C boot sequencer failed while loading the reset configuration words. The bit is cleared by writing a 1 to it; writing zero has no effect.	0 No boot sequencer failure. 1 Boot sequencer failed.

Table 5-11. RSR Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 15–14	0	Reserved. Write to zero for future compatibility.	
SWSR 13	0	**Software Soft Reset** Indicates whether a software soft reset has occurred. This bit is cleared by writing a 1 to it; writing zero has no effect.	0 No software soft reset. 1 Software soft reset initiated.
SWHR 12	0	**Software Hard Reset** Indicates whether a software hard reset has occurred. This bit is cleared by writing a 1 to it; writing zero has no effect.	0 No software hard reset. 1 Software hard reset initiated.
— 11–9	0	Reserved. Write to zero for future compatibility.	
JS 8	0	**JTAG Soft Reset** Indicates whether a software hard reset has occurred. This bit is cleared by writing a 1 to it; writing zero has no effect.	0 No JTAG soft reset requested. 1 JTAG soft reset requested.
— 7–4	0	Reserved. Write to zero for future compatibility.	
SW4 3	0	**Software Watchdog Timer 4** Indicates whether software watchdog timer 4 has expired.	0 Software watchdog timer 4 not expired. 1 Software watchdog timer 4 expired.
SW0 2	0	**Software Watchdog Timer 0** Indicates whether software watchdog timer 0 has expired.	0 Software watchdog timer 0 not expired. 1 Software watchdog timer 0 expired.
SRS 1	0	**Soft Reset Status** When an external or internal soft reset event is detected, SRS is set and remains that way until software clears it. SRS is cleared by writing a 1 to it; writing zero has no effect.	0 No soft reset event. 1 A soft reset event was detected.
HRS 0	0	**Hard Reset Status** When an external or internal hard reset event is detected, HRS is set and it remains set until software clears it. HRS is cleared by writing a 1 to it; writing zero has no effect.	0 No hard reset event. 1 A hard reset event was detected.

5.3.4 Reset Protection Register (RPR)

RPR Reset Protection Register Offset 0x18

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

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

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

Table 5-12. RPR Bit Descriptions

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

5.3.5 Reset Control Register (RCR)

RCR Reset Control Register Offset 0x1C

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

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

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

Table 5-13. RCR Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.	
SHR 2	0	**Soft Hard Reset** Setting this bit cause the MSC8144E to convert all hard reset flows to soft reset flows. This feature can be used for debug. This bit returns to its reset state during the reset sequence, so reading it always returns a 0.	0 Normal hard reset flow. 1 Hard reset flow converted to soft reset flow.
SWHR 1	0	**Software Hard Reset** Setting this bit cause the MSC8144E to begin a hard reset flow. This bit returns to its reset state during the reset sequence, so reading it always returns a 0.	0 Normal operation. 1 Initiates a hard reset.
SWSR 0	0	**Software Soft Reset** Setting this bit cause the MSC8144E to begin a soft reset flow. This bit returns to its reset state during the reset sequence, so reading it always returns a 0.	0 Normal operation. 1 Initiates a soft reset.

5.3.6 Reset Control Enable Register (RCER)

RCER Reset Control Enable Register Offset 0x20

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

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

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

Table 5-14. RCER Bit Descriptions

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

t

Boot Program 6

The boot program initializes the MSC8144E after it completes a reset sequence. The MSC8144E can boot from an external host through the PCI or 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 they finish the reset sequence, all cores 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 M2 memory at 0xC007B000–0xC007FFFF (20 KB). Do not write to this area during boot loading.

When they finish the boot sequence, all cores 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, EE1 is configured as a debug acknowledge output and the core writes an indication of the root error cause to address 0xC007B004 in M2 memory (see **Section 6.8**, *Boot Errors*, on page 6-22 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 6.4.1**. 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. The identification of Core 0 and the other cores is done using M_PIR[PNS].
- *Shared configuration (core 0)*. Includes general configuration of internal CLASS, I^2C, RapidIO, QUICC Engine subsystem, and the DSP core subsystem blocks and registers.
- *M3 delay (core 0)*. The M3 needs additional time to complete initialization. Core 0 generates a read transaction towards M3, uses the transaction completion as the end of the M3 initialization, and then allows the bootloader to place code in the M3 memory.
- *Boot mode select (core 0)*. This part includes downloading of code from one of the MSC8144E bootable ports as defined by the RCWHR[BPRT] field.
- *Boot completion*. All cores complete the boot operation and jump to a user-specified address.

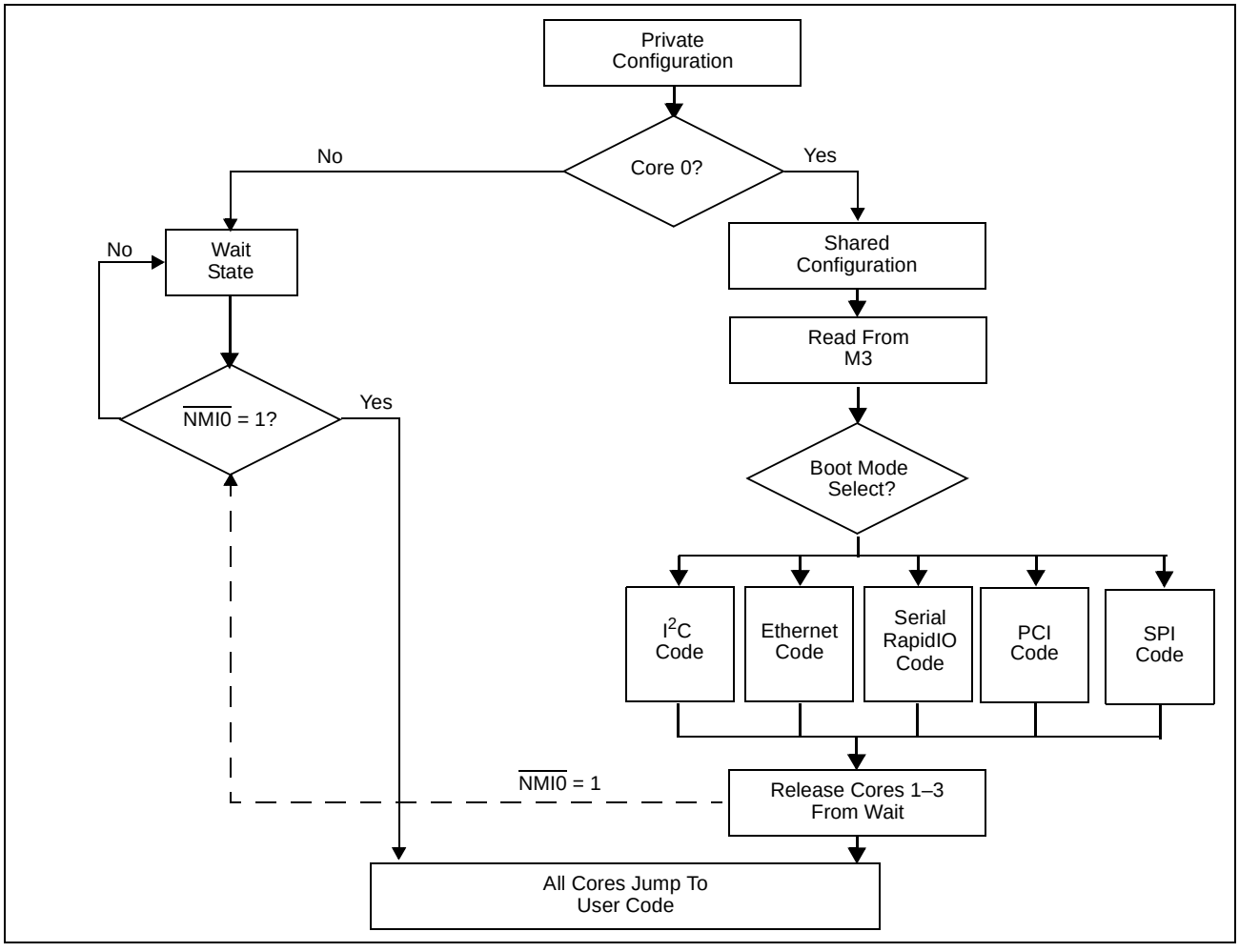


Figure 6-1. Boot Sequence Diagram

6.2 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 SC3400 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 M2 memory space dedicated to the boot operation.
- The Error Detection Code (EDC) exception is enabled by setting the EDCEE bit in the MMU Control Register (M_CR).

Note: The EPIC, L1 ICache, and MMU are part of the SC3400 DSP core subsystem. See the the *MSC8144 SC3400 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.

6.3 M3 Initialization Delay

The boot code reads the GSR[M3_EXIST] bit to determine whether M3 exists in this system (see **Chapter 8**, *General Configuration Registers* for details). If M3 exists in the system the core issues a read transaction from the M3 memory. The successful closing of this transaction indicates that the M3 controller has finished its initialization and the M3 memory is accessible. If M3 does not exist in the system, the core disables the T2 port of CLASS1 disabling accesses towards M3.

6.4 Shared Configuration

The shared configuration includes the following:

- If the I^2C controller is used to load the RCW for multi-device slaves (see **Section 6.4.1**, *Multi Device Support for the I^2C Bus*, on page 6-5) 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 RCWLR[ERIO] is set
  - 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_GCR[4–7] are set to disable the tri-stating of the serial RapidIO lanes.
  - If RCWLR[1X] is set, 1x is forced on the RapidIO interface (P0CCSR)
- QUICC Engine module priority is set to be 1 with emergency not masked (that is SDMR[EB1_PR] = 01).

6.4.1 Multi Device Support for the I²C Bus

The MSC8144E can share the I²C EEPROM device with other MSC8144E 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 I²C bus:

1. For each EEPROM in the system, there must be at least one EEPROM master. The EEPROM master is also the reset master (RCWHR[RM])
2. For each EEPROM, there can be 0 or more EEPROM slaves. An EEPROM slave is defined as a device that reads is RCW from the EEPROM and uses data on the bus during boot. The number of EEPROM slaves is stored as a single byte at address 0x8F of the EEPROM.
3. For each EEPROM, there can be 0 or more reset slaves. A reset slave is defined as a device that only reads its RCW from the EEPROM but does not read data from it during boot. The number of reset slaves is stored as a single byte at address 0x11 of the EEPROM.
4. Every EEPROM slave must also be a reset slave.
5. There may be up to 15 reset slaves per EEPROM
6. As a consequence of the conditions listed in 1–5, the limitations on the number of slaves is defined as $0 \leq$ #EEPROM slaves $\leq$ #reset slaves ≤ 15
7. The lowest numbered reset slave must be a higher numbered slave than the highest numbered EEPROM slave (for example, if EEPROM slaves are slaves 0–4, then reset slaves are slaves 5–12).
8. EEPROM slaves must be numbered sequentially from 0 upward.
9. All devices connected to the same EEPROM must have $\overline{\text{PORESET}}$ asserted simultaneously, that is, no single device go through the PORESET sequence without the others.
10. The EEPROM master must have $\overline{\text{HRESET}}$/$\overline{\text{SRESET}}$ asserted unless the slaves are reset as well

11. The EEPROM slaves can have $\overline{\text{HRESET}}$/$\overline{\text{SRESET}}$ asserted without the master being reset. However, there must be external logic that performs the actions that the master performs during its boot sequence. The logic may be implemented as periodic polling by the master, asserting $\overline{\text{NMI}}$ to the reset master, or using an FPGA or other implementation)
12. If there is a shared EEPROM in use in any stage of the reset/boot flow (RCW, serial RapidIO interface configuration, MAC address, I^2C boot), all devices MUST load their RCW from the shared EEPROM.

Note: If the reset master (RCWHR[RM]) fails (due to a stuck SDA) to read the data at 0x11 or 0x8F of the EEPROM or fails during the sequence of driving the RCW to the reset slaves, the core goes into a debug state and writes the appropriate error code to the M2 memory (see **Section 6.8**).

6.4.2 Example Configuration

Figure 6-2 describes a I^2C multi device system in which MSC8122 #0 is a reset master and MSC8122 #1 is a reset slave. The reset master uses {GPIO[0–3], GPIO[21]} to release the reset slaves. The MSC8144 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.

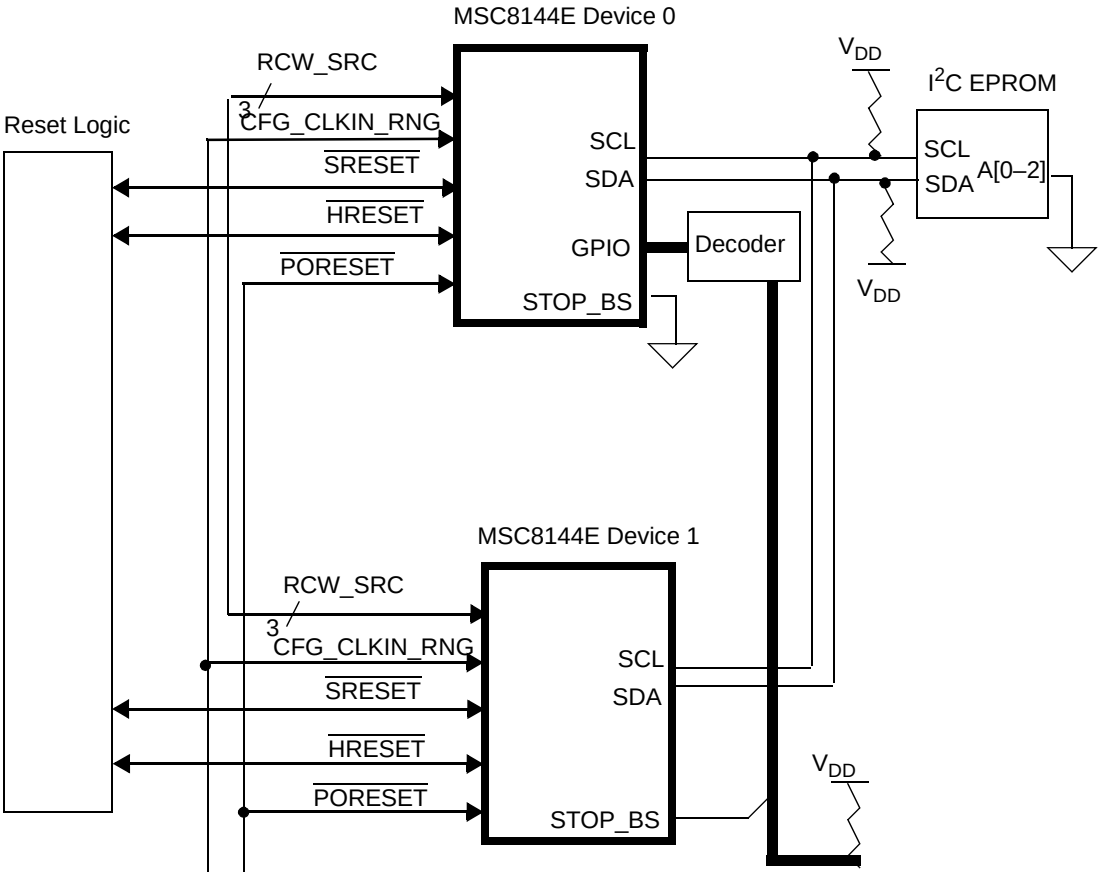


Figure 6-2. I^2C Multi Device System

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

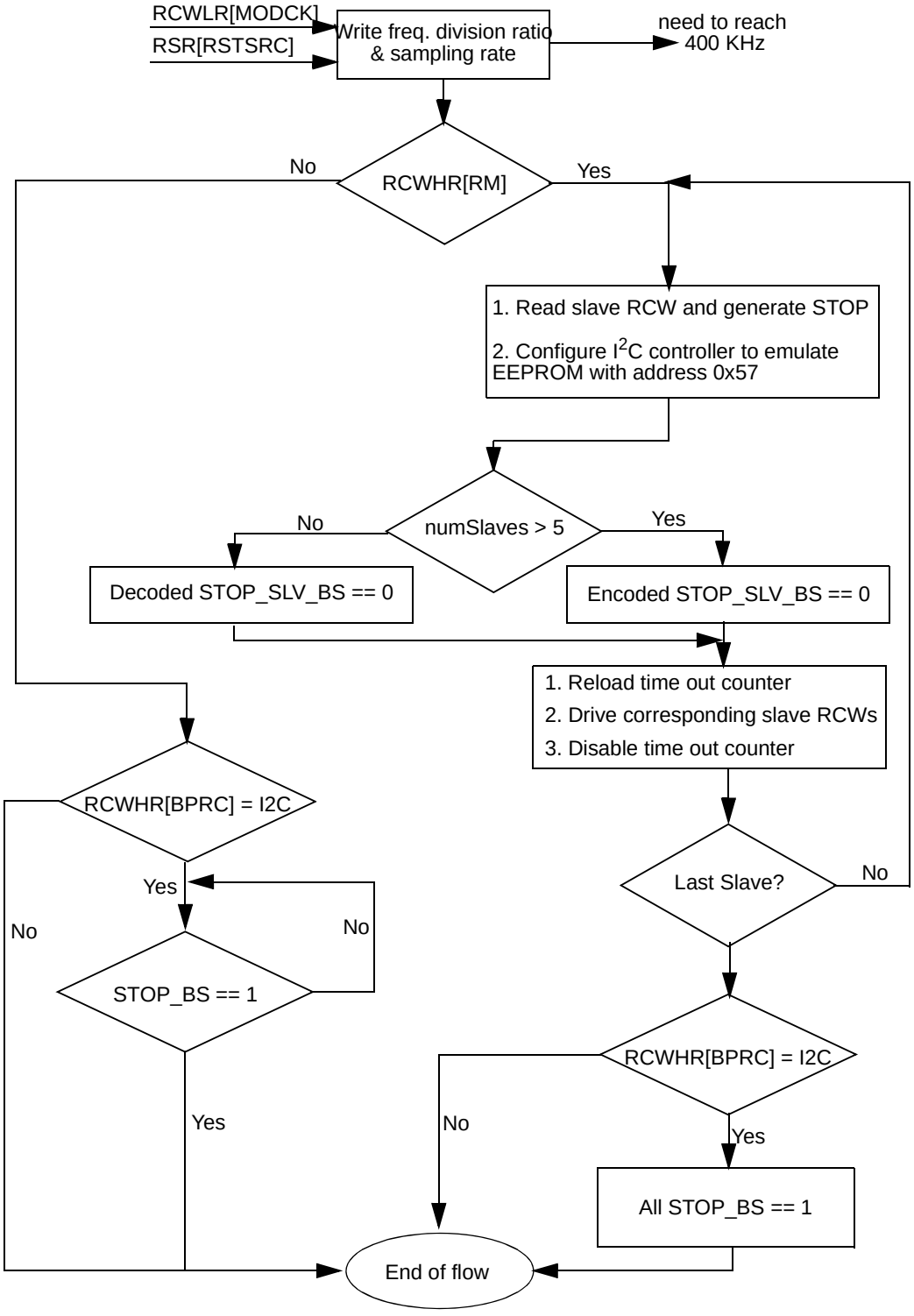


Figure 6-3. I^{2}C initialization and Multi Device Support

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

1. The MSC8144E reads RSR[SWSR] and RSR[SWHR] to determine if the reset is PORESET. If it is not PORESET, this section of the boot is bypassed entirely.
2. The MSC8144E reads RCWLR[RM] to determine if it is the master on the multi-device board.
3. If the MSC8144E 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 SCL is set as closely as possible to 400 kHz.
   b. The Reset master reads the slaves RCW into M2.
   c. The I^2C is configured to work as an EEPROM (slave mode) with address 0x57.
   d. The MSC8144E deassert $\overline{\text{STOP_BS}}$ for the current slave (directly or via the decoder).
   e. Drive preamble 0xAA55AA
      Drive header 0xFFFFFF
      Drive RCWLR
      Drive header 0xFFFFFF
      Drive RCWHR
      Disable the I^2C controller in order to free it up as the slaves I^2C controller does not generate 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.
4. If the MSC8144E is a an EEPROM slave, the boot waits until $\overline{\text{STOP_BS}}$ is to pulled high before continuing with the boot program, thus allowing all reset slaves to read their reset word before any device tries to access the EEPROM for boot code.

6.5 Boot Modes

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

6.5.1 I²C EEPROM

The MSC8144 boot expects the I²C EEPROM to be divided in to four sections:

1. Reset words. This section starts at address 0x0000 of the EEPROM and includes the reset words for the reset master, an indication as to the number of reset slaves and the reset words for all the slaves.
2. Reserved. This section starts at address 0x008A and should be set to 0 for future compatibility.
3. Boot configuration. This section starts at address 0x0090 of the EEPROM and can contain one of the following:
   - MAC addresses for up to 64 devices (6 bytes per address). The boot knows to associate each address with the appropriate device based on an offset of 6 × RCWHR[DEVID]. The expected format is consecutive 6 byte fields.
   - Serial RapidIO configuration. This option allows the user to configure up to 48[1] registers and should be used to set the appropriate values of RIO_CR and SGMII_CR in the general configuration block. The expected format is address, data pairs. After the last such pair (if there are less than 48) the address field should be set to 0xFFFFFFFF to signal that no more configurations are necessary.
4. Boot code. This section starts at address 0x0210 of the EEPROM and contains the user code. The boot code must be of the size (n × 4) + m[bytes], where n is any integer greater than or equal to 0 and m is either 0 or 1. If n is larger than 0, the value in the Destination Address field must be 32-bit aligned.

Note: Although not all sections are used by all boot port options, the section addresses are fixed. However, if an I²C EEPROM is required in the system for any reason (RCW, configuration, security, and so on), addresses 0x000–0x20F must be valid.

1. 48 pairs is the amount that fits in the same space as 64 MAC addresses ($\lfloor (64 \cdot 6)/(2 \cdot 4) \rfloor = 48$)

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

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

Figure 6-4. EEPROM Contents

Address	0	1	2	3	4	5	6	7	Description
0x0086	Reset Configuration Word High [31–24]								Reset Configuration Word High of Slave 15
0x0086	Reset Configuration Word High [23–16]								
0x0088	Reset Configuration Word High [15–8]								
0x0089	Reset Configuration Word High [7–0]								
0x008A–0x008E	.....								Reserved
0x008F	15 ≤ numResetSlaves ≤ numEEPROMslaves ≤ 0								Number of EEPROM Slaves
0x0090									Configuration/MAC Address First Byte
0x020F									Configuration Last Byte/ Device 0x2F Last MAC Address Byte
0x0210	1	0	1	1	1	1	1	1	Block Control (Block 0)
0x0211	size[31–24]								Block Size
0x0212	size[23–16]								
0x0213	size[15–8]								
0x0214	NBA[31–24]								Next Block Address
0x0215	NBA[23–16]								
0x0216	NBA[15–8]								
0x0217	NBA[7–0]								
0x0218	DA[31–24]								Destination Address
0x0219	DA[23–16]								
0x021A	DA[15–8]								
0x021B	DA[7–0]								
	CODE								Payload Data
	Checksum[15–8]								Checksum and $\overline{\text{Checksum}}$
	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 0.5 Mbytes, 19-bit addressing is used. The 3 most significant bits (msb) of the address are placed as the 3 least significant bits (lsb) of the I^2C slave address. The 4 msb of the I^2C slave address are always 1010. The I^2C controller expects a specific memory image when trying to read data during the EEPROM as shown in **Figure 6-5**.

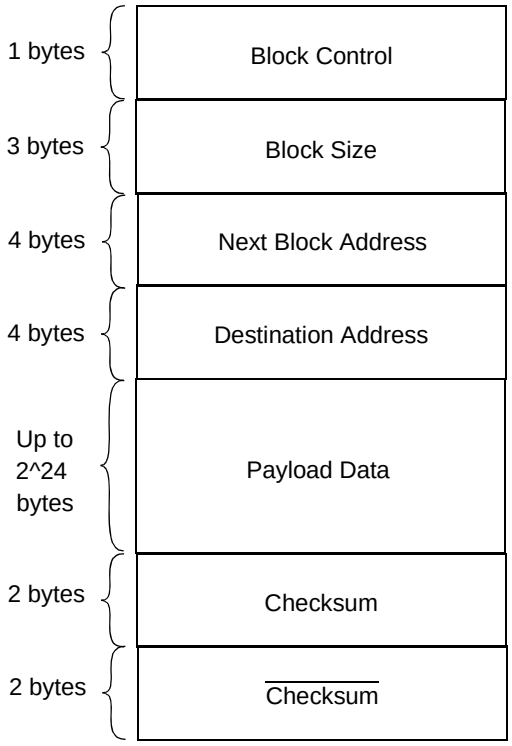


Figure 6-5. EEPROM Data Format

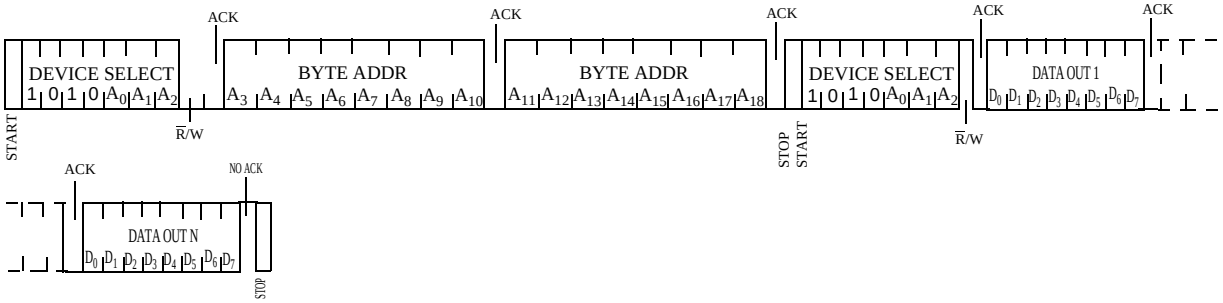
The I^2C memory image is consisted of:

1. *Block Control***.** A 1-byte control field that contains:
   - 1 bit of CSE. A 1 indicates that the checksum is enabled.
   - 1 reserved bit that should be cleared (0).
   - 6 bits of CHIP_ID indicate the destination chip. 0x3F means broadcast.
2. *Block Size***.** These 3 bytes represent the number of bytes in the payload data field (e.g if the payload size is 12 bytes, Block Size = {0x00, 0x00, 0x0C}).
3. *Next Block Address*. The address in which the next block is located. If the next block address equals 0x0 the bootloader assumes that the next block is sequential. If next block address equals 0xFFFFFFFF, this block is the end block.
4. *Destination Address*. The address to which the payload data should be written.
5. *Payload Data***.** Holds up to 2^{24} bytes (16 Mbytes) of data to be written to on-device memory according to the destination address.

6. *Checksum*. A 2-byte field that holds the XOR of all previous data (Block Control and on). The boot code XORs each received 2 bytes with the previous checksum value and verifies the validation by comparing it to this field.
7. $\overline{Checksum}$. A 2-byte field that holds bitwise-not of the Checksum.

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

The SCL frequency is set as closely to 400KHz as possible, as mentioned in **Section 6.4.1**. For each block, the Software I^{2}C read access begins with the boot code driving the device select and 2 bytes of address, followed by a RESTART condition. The I^{2}C 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 MSC8144. After the ninth unacknowledged bit, the boot code generates a STOP condition. **Figure 6-6** describes the Software I^{2}C read access.



Note: A_0 and D_0 are the most significant bits.

Figure 6-6. I^{2}C Read Access

6.5.2 Ethernet

The MSC8144E device can load files through the Ethernet port using DHCP (Dynamic Host Configuration Protocol) and TFTP (Trivial File Transfer Protocol).

- Supports RGMII and SGMII @1000 Mbps full duplex connected to a SWITCH (MAC-to-MAC).
- Supports RGMII, RMII, and SMII @100 Mbps full duplex connected to a SWITCH (MAC-to-MAC).
- 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.

Ethernet booting is enabled through Ethernet controller 1 (UCC1) depending on the values of RCWHR[PIN_MUX] and RCWLR[SGMII]. **Table 6-1** lists the available options.1

Table 6-1. Ethernet Boot Availability by I/O Multiplexing Mode

Multiplexing Mode	RGMII	RMII/SMII	SGMII
0	Not supported	Not supported	If RCWLR[SGMII1] = 1, Ethernet booting is supported. Otherwise, it is not supported
1	Supported	Supported	
2	Supported	Supported	
3	Not supported	Not supported	
4	Not supported	Not supported	
5	Not supported	Not supported	
6	Supported	Not supported	
7	Not supported	Not supported	
8	Supported	Supported	
9	Supported	Supported	
10	Not supported	Not supported	
11	Not supported	Not supported	
12	Supported	Supported	

Figure 6-7 describes the Ethernet bootloader flow.

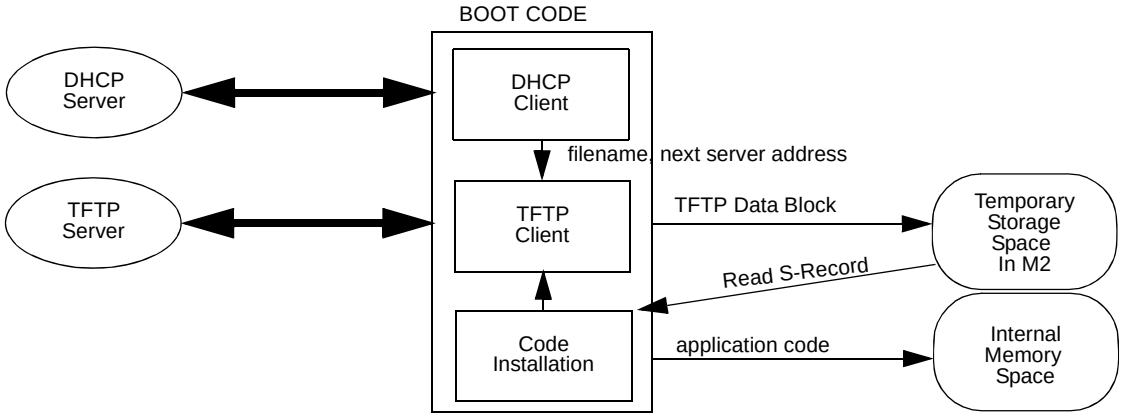


Figure 6-7. Ethernet Bootloader Flow

:The Ethernet bootloader flow includes:

1. Configuring the QUICC Engine drivers based on RCWHR[BPRT] and RCWHR[PIN_MUX].
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.

6.5.2.1 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**.

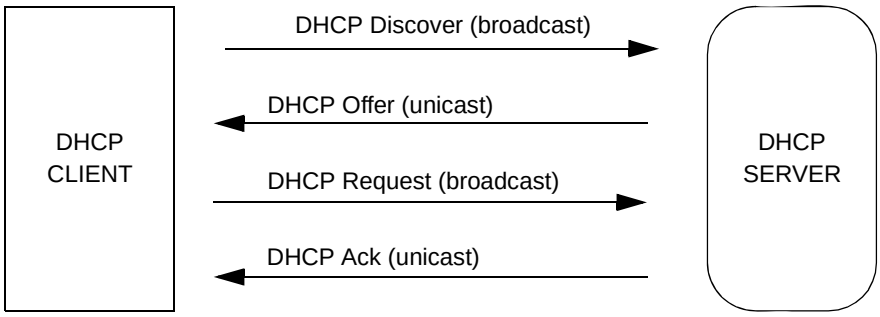


Figure 6-8. DHCP Transactions

- 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 MSC8144E MAC address during the boot:

- User defined and read from an I^2C EEPROM. See **Section 6.5.1** for details.
- Predefined default using the following fields:
  - A constant of 24 bits: {0x1E, 0xF7, 0xD5}
  - 8 bits consisting of (RCWHR[DEVID]) (aligned to the right and padded with 0)
  - A constant of 16 bits: {0x00, 0x00}

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

6.5.2.2 TFTP Client

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

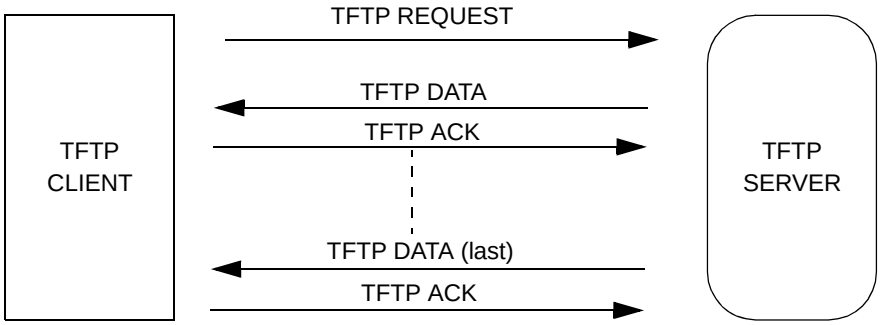


Figure 6-9. TFTP Transactions

- 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.

6.5.2.3 Boot File Format

The Ethernet bootloader expects an application file in the form of an S-Record file. The S-Record file is a text representation of the binary program code. The S-Record file structure is describes in **Figure 6-10**.



Figure 6-10. S-Record file structure

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

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. 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 6-2**.

Table 6-2. 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 no 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.

Shown below is a typical S-record format file:

S0030000FC
S30D00002FE731DC3180BEF09E7062
S705000000000C

The S0 record is comprised 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 comprised 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 comprised 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 0xC007B010).
- **0C**. Checksum field.

Each S-Record line includes an address field that maps the lines data content to a memory location in MSC8122. Core 0 moves the data to this address.

Note: Because the MSC8144E uses 32-bit addressing, use of S3 and S7 is recommended.

6.5.3 Serial RapidIO Interconnect

In this procedure a Serial RapidIO master waits for the MSC8122 boot program to finish its default initialization and then initializes the device by typically loading code and data to the on device memory.

6.5.3.1 Serial RapidIO Without I^2C Support

The boot code writes 0x17171717 to address 0xC007B000 and then disables the tri-states on the Serial RapidIO lanes based on RCWLR[RIOE] and RCWLR[1X] by setting the appropriate bits in L_GCR[4–7]. Afterwards, it polls this address until the serial RapidIO master writes 0xA5A5A5A5 to it, thereby indicating that it has finished its code loading. **Figure 6-11** describes the boot flow from Serial RapidIO interconnect**.**

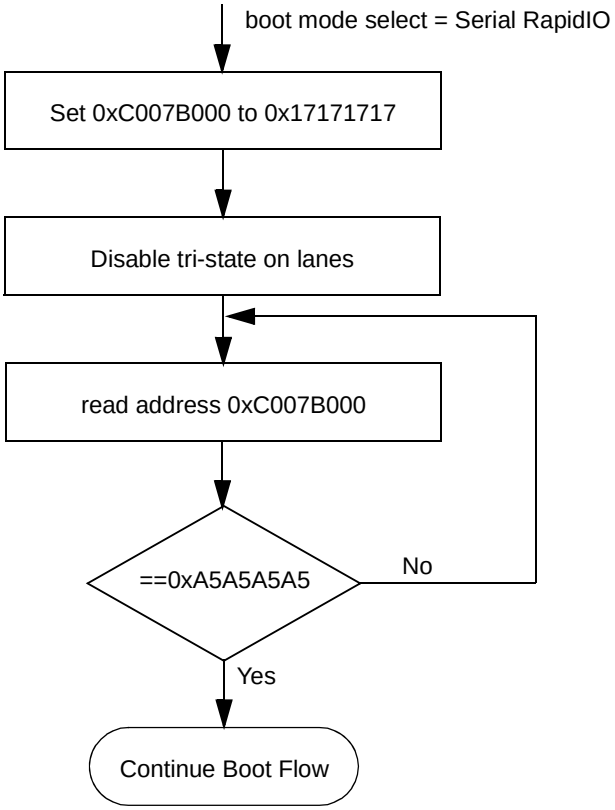


Figure 6-11. Serial RapidIO Boot Flow

6.5.3.2 Serial RapidIO with I^{2}C Support

The user can place {addr, data} pairs in the I^{2}C EEPROM to configure various registers. The address field should be as seen by the SC140. This feature is most commonly used to configure RIO_CR and SGMII_CR in the general configuration block. The boot supports up to 48 such pairs. If less than 48 are used, the 4 bytes following the last pair should be set to 0xFFFFFFFF.

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

6.5.4 PCI

In this procedure a PCI HOST waits for the MSC8122 boot program to finish its default initialization and then initializes the device by typically loading code and data to the internal memory. Before enabling the PCI Host the boot code configures 3 PCI address inbound windows as follows:

1. Inbound window 0 maps a window of 512 KB starting at address 0xC0000000 which consists of the entire M2 address space.
2. Inbound window 1 maps the a window of 16 MB starting at address 0xD0000000 which consists of the entire M3 address space.

3. Inbound window 2 maps the size of DDR address space indicated in RCWHR[BPRT] starting at address 0x40000000.

In addition to these three windows there is another hard-wired window that maps all the CCSR address space (956 KB). After the inbound window sizes are configured, the boot code writes 0x17171717 to address 0xC007B000 and enables the PCI by clearing the lock bit of the PCI Function Configuration Register. Once the PCIs on all the MSC8144E devices on board are enabled, the PCI host can either numerate the address space as is, or reconfigure some of the inbound window sizes by using the hard-wired window that grants access to the CCSR space prior to numeration. Core 0 waits for the PCI Host to write 0xA5A5A5A5 to address 0xC007B000, thus finishing the handshake process.

6.5.5 SPI

The MSC8144E 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 I^2C EEPROM (see **Section 6.5.1**, *I^2C EEPROM*, on page 6-10, 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.

6.6 Jump to User Code

Before finishing its tasks the boot code preforms these actions:

- If RCWHR[RIO] is cleared, the boot code disables host accesses by RapidIO interface to internal memory space by putting the lanes into tri-state high impedance state.
- Invalidate all range of ICaches and close MMU program windows.
- Core internal registers (other than R0 and VBA) are set to 0x00000000.
- All configurations which were done by the boot code are cleared.
  - Module registers
  - GPIO configurations
  - Write 0x00000000 to GIER (see **Chapter 8**, *General Configuration Registers*) to clear the register.
  - Disable all interrupts (NMI excluded)
  - Clear all QUICC Engine registers by writing 1 to QECMDR[RST]

- 0x900D900D is written to 0xC007B00C in M2 indicating that the boot has finished executing.
- All cores jump to the address written in 0xC007B010. The value in this address should be written during the boot loading process.

6.7 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 other than 91 and 92 in the EPIC puts the core in debug.
- EDC is enabled.
- Core register values are not guaranteed and should be initialized before use.

6.8 Boot Errors

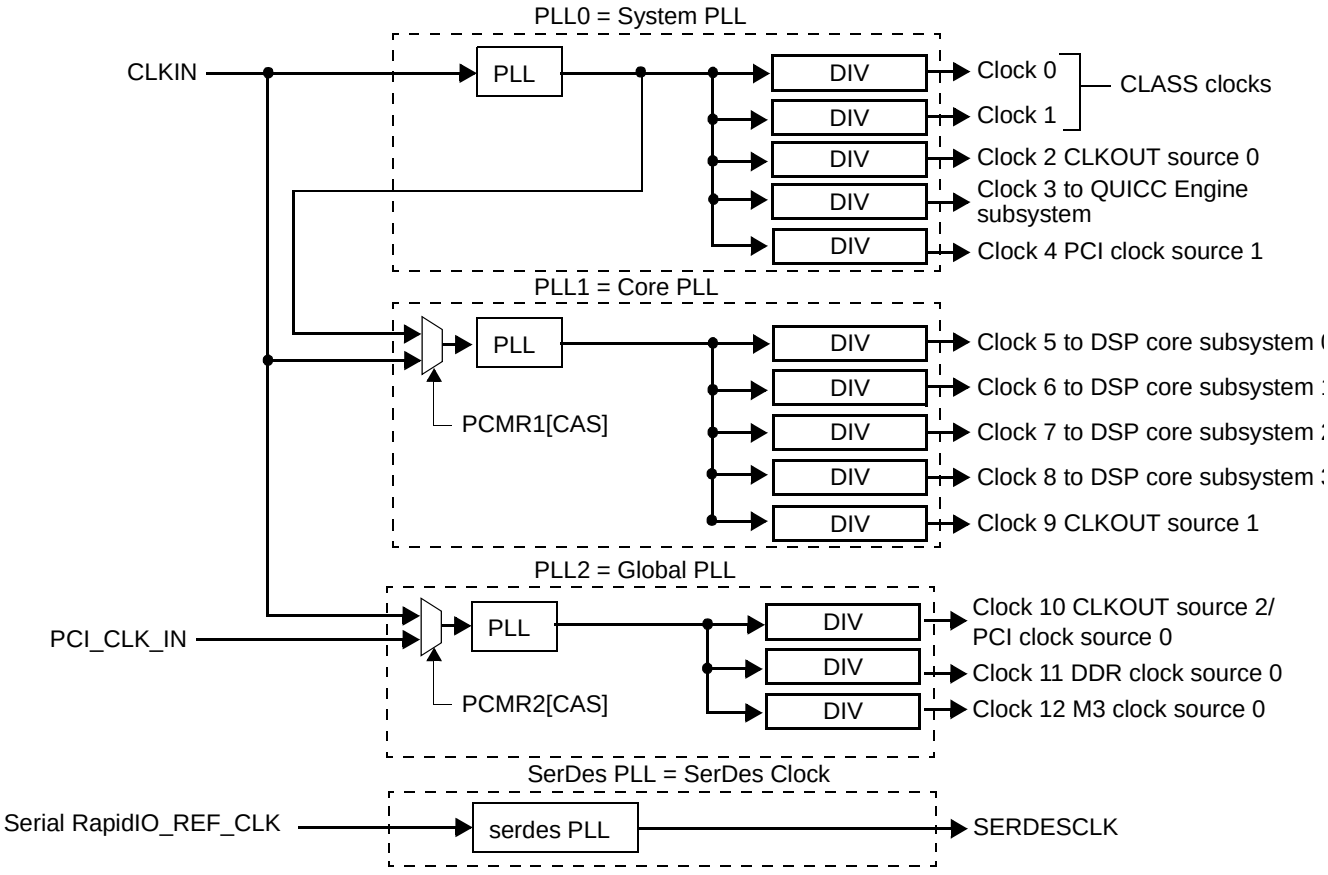
If the boot code fails, an indication as to the root cause is written to 0xC007B004. The possible reasons are listed in **Table 6-3**.

Table 6-3. Boot Error Codes

Error Code	Description
0x003FEFFF	Catastrophic Error. SmartDSP OS Function failed.
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
0x003FEFFB	Empty frame from TFTP server
0x003FEFFA	TFTP server sent ERROR code (05)
0x003FEFF9	Unsupported boot port
0x003FEFF8	Reset slave does not respond
0x003FEFF6	Using boot with RCWHR[STTM] equaling 1
0x003FEFF5	Boot port isn't supported with current pin multiplexing.
0x003FEFF4	Too many I^2C RCW slaves in address 0x11 of the EEPROM
0x003FEFF3	Error in protocol between I^2C RCW master and slave
0x003FEFF2	Boot port trying to write to area designated for boot (0xC007B014–0xC007FFFF)
0x0027EFFE	Lost arbitration on I^2C bus.
0x0027EFFD	Time-out on I^2C acknowledge (9th clock).
0x0027EFFC	Stuck SDA (I^2C bus).
0x00000000	Unexpected debug condition in the SC3400 Core (unexpected interrupt, EE0 asserted and so on)

Clocks 7

The clock circuits generate the clocks for the cores, the internal CLASS buses, the QUICC Engine subsystem, the PCI interface, the TDM, internal memory, the DDR-SDRAM memory controller, and SERDES interface for the RapidIO controller. The clock generation components and clock scheme are shown in **Figure 7-1**. The PLL0 input is the CLKIN signal generated from a crystal-based oscillator. The PLL1 input can be the CLKIN signal or the output from PLL0. The PLL2 input can be the CLKIN signal or the PCI_CLK_IN (PCI bus clock input) signal. Each PLL uses its input clock to generate a fast clock that is synchronized to the input clock. The fast clock is distributed to each of the clock dividers to generate the clocks that are distributed to the system blocks. The clock circuits are locked, according to the selected clock mode, when the first stage of the system reset configuration is done (reset configuration is controlled by the RESET block).



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

Figure 7-1. MSC8144E Clock Scheme

MSC8144E clock mapping is listed in **Table 7-1**.

Table 7-1. MSC8144E Clock Mapping

Clock Name	Divider Used		Select
	Select = 0	Select = 1	
CLASS128	0	NA	NA
CLASS64	1	NA	NA
QUICC Engine RISC	3	NA	NA
Core0	5	2	PCMR1[SYNCLK]
Core1	6	2	PCMR1[SYNCLK]
Core2	7	2	PCMR1[SYNCLK]
Core3	8	2	PCMR1[SYNCLK]
PCI	10	4	PCMR0[GP_CTL2]
DDR	11	0	PCMR0[GP_CTL1]
M3	12	0	PCMR0[GP_CTL0]

Note: The selection of a clock source, as indicated in **Table 7-1**, and a pre-defined clock mode (listed in **Table 7-2** through **Table 7-6**) is subject to the following limitations:

1. The PLL selected as the source for PCI_CLK must have its equivalent delay feedback loop active (that is, the PCMRn[EQDLY] bit must be set) and its input clock must be the external PCI bus clock.

Note: If the System PLL (PLL0) is selected, then you must connect the PCI bus clock to the CLKIN input.

2. The ratio between the PCI clock frequency and external PCI bus clock frequency must be one of the following:

   2:1, 3:1, 4:1, 5:1, or 6:1

3. The CLASS64 clock frequency must be greater than or equal to the PCI clock frequency.
4. The CLASS128 clock frequency must be greater than or equal to the M3 frequency.

Use the following formula to calculate a specific clock frequency (F_J):

$$\text{If BYPi} = 1 \qquad F_J = F_{IN}/(CK_J DF)$$

$$\text{Else} \qquad F_J = F_{IN}/PDi \times MFi/(CK_J DF)$$

where,

BYPi = the value of the BYP bit for PLLi

F_{IN} = the frequency of the input clock to PLLi (for PLL1 and PLL2, the clock selection depends on the value of the CASi bit)

CK_JDF = the divide factor for clockj

PDi = the predivide value for PLL i

MFi = the multiply factor for PLL i

7.1 Clock Control Logic

The clock control logic gets the encoded clock mode from the reset configuration word low (MODCK[5–0]) and decodes it to determine the division and multiplication factors, bypass state and additional settings. It then controls the locking of the clock generation circuits (PLL, dividers, and so forth) and synchronizes the output clocks to the input clock.

7.1.1 Clock Modes

The clock logic supports several pre-defined clock modes as described in **Table 7-2**, **Table 7-3**, **Table 7-4**, **Table 7-5**, and **Table 7-6**. The clock circuits are locked by the time that the first stage of the reset configuration is complete (see **Chapter 5**, *Reset* for details).

Table 7-2. Clock Modes (PLL0)

MODE	PCMR0				
	PD	MF	BYP	INPRNG	EQ DLY
0	1	6	0	1	1
1	1	6	0	1	1
2	1	6	0	1	1
3	1	6	0	1	1
4	1	6	0	1	0
5	1	6	0	1	0
6	1	6	0	1	0
7	1	6	0	1	0
8	1	6	0	1	1
9	1	6	0	1	1
10	1	6	0	1	1
11	1	6	0	1	1
12	1	5	0	1	1
13	1	5	0	1	1
14	1	6	0	1	0
15	1	6	0	1	0
16	1	6	0	1	0
17	1	3	0	1	0
18	1	6	0	1	0
19	1	6	0	1	0
20	1	6	0	1	0

Table 7-2. Clock Modes (PLL0) (Continued)

MODE	PCMR0				
	PD	MF	BYP	INPRNG	EQ DLY
21	1	6	0	1	0
22	1	6	0	1	0
23	1	6	0	1	0
24	1	6	0	1	0
25	1	3	0	1	0
26	1	6	0	1	0
27	1	6	0	1	0
28	1	6	0	1	0
29	1	6	0	1	0
30	1	6	0	1	0
31	1	3	0	1	0
32	1	6	0	1	0
33	1	6	0	1	0
34	1	6	0	1	0
35	1	3	0	1	0
36	1	6	0	1	0
37	1	3	0	1	0
38	1	6	0	1	0
39	1	6	0	1	0
40	1	6	0	1	0
41	1	3	0	1	0
42	1	6	0	1	0
43	1	6	0	1	0
44	1	4	0	1	0
45	1	4	0	1	0
46	1	2	0	1	0
47	1	2	0	1	0
48	1	4	0	1	0
49	1	2	0	1	0
50	1	2	0	1	0
51	1	4	0	1	0
52	1	2	0	1	0
53	1	12	0	0	0
54	1	12	0	0	0
55	1	12	0	0	0
56	1	12	0	0	0
57	1	10	0	0	0
58	1	8	0	0	0
59	1	6	0	1	0
60	1	5	0	1	0
61	1	1	1	0	1
62	1	1	1	0	1
63	1	1	1	0	1

Table 7-3. Clock Modes (PLL1)

MODE	PCMR1					
	PD	MF	SYNCLK	BYP	CAS	EQ DLY
0	1	12	0	0	0	0
1	1	2	0	0	1	0
2	1	15	0	0	0	0
3	2	5	0	0	1	0
4	1	12	0	0	0	0
5	1	2	0	0	1	0
6	1	15	0	0	0	0
7	2	5	0	0	1	0
8	1	12	0	0	0	0
9	1	2	0	0	1	0
10	1	15	0	0	0	0
11	2	5	0	0	1	0
12	1	15	0	0	0	0
13	2	6	0	0	1	0
14	1	12	0	0	0	0
15	1	2	0	0	1	0
16	1	12	0	0	0	0
17	1	6	0	0	0	0
18	1	12	0	0	0	0
19	1	2	0	0	1	0
20	1	2	0	0	1	0
21	1	12	0	0	0	0
22	1	2	0	0	1	0
23	1	12	0	0	0	0
24	1	12	0	0	0	0
25	1	6	0	0	0	0
26	1	2	0	0	1	0
27	1	2	0	0	1	0
28	1	15	0	0	0	0
29	2	5	0	0	1	0
30	1	15	0	0	0	0
31	1	8	0	0	0	0
32	1	15	0	0	0	0
33	2	5	0	0	1	0
34	2	5	0	0	1	0
35	1	8	0	0	0	0
36	1	15	0	0	0	0
37	1	8	0	0	0	0
38	2	5	0	0	1	0
39	1	15	0	0	0	0
40	1	15	0	0	0	0
41	1	8	0	0	0	0
42	2	5	0	0	1	0
43	2	5	0	0	1	0
44	1	8	0	0	0	0
45	1	2	0	0	1	0
46	1	8	0	0	0	0

Table 7-3. Clock Modes (PLL1) (Continued)

MODE	PCMR1					
	PD	MF	SYNCLK	BYP	CAS	EQ DLY
47	1	4	0	0	1	0
48	1	6	0	0	0	0
49	1	8	0	0	0	0
50	1	4	0	0	1	0
51	1	6	0	0	0	0
52	1	6	0	0	0	0
53	1	2	0	0	1	0
54	1	2	0	0	1	0
55	1	2	0	0	1	0
56	2	5	0	0	1	0
57	1	3	0	0	1	0
58	1	16	0	0	0	0
59	1	15	0	0	0	0
60	1	15	0	0	0	0
61	1	1	0	1	0	0
62	1	1	0	1	0	0
63	1	2	0	0	1	0

Table 7-4. Clock Modes (PLL2)

MODE	PCMR2				
	PD	MF	BYP	CAS	EQ DLY
0	1	5	0	0	0
1	1	5	0	0	0
2	1	8	0	0	0
3	1	8	0	0	0
4	1	12	0	1	1
5	1	12	0	1	1
6	1	12	0	1	1
7	1	12	0	1	1
8	1	8	0	0	0
9	1	8	0	0	0
10	1	5	0	0	0
11	1	5	0	0	0
12	1	6	0	0	0
13	1	6	0	0	0
14	1	4	0	1	1
15	1	4	0	1	1
16	1	10	0	1	1
17	1	10	0	1	1
18	1	5	0	1	1
19	1	10	0	1	1
20	1	5	0	1	1
21	1	6	0	1	1
22	1	6	0	1	1
23	1	16	0	1	1
24	1	8	0	1	1

Table 7-4. Clock Modes (PLL2) (Continued)

MODE	PCMR2				
	PD	MF	BYP	CAS	EQ DLY
25	1	8	0	1	1
26	1	16	0	1	1
27	1	8	0	1	1
28	1	4	0	1	1
29	1	4	0	1	1
30	1	10	0	1	1
31	1	10	0	1	1
32	1	5	0	1	1
33	1	10	0	1	1
34	1	5	0	1	1
35	1	12	0	1	1
36	1	6	0	1	1
37	1	6	0	1	1
38	1	6	0	1	1
39	1	16	0	1	1
40	1	8	0	1	1
41	1	8	0	1	1
42	1	16	0	1	1
43	1	8	0	1	1
44	1	6	0	1	1
45	1	6	0	1	1
46	1	6	0	1	1
47	1	6	0	1	1
48	1	6	0	1	1
49	1	6	0	1	1
50	1	6	0	1	1
51	1	6	0	1	1
52	1	8	0	1	1
53	1	10	0	1	1
54	1	12	0	1	1
55	1	16	0	1	1
56	1	12	0	1	1
57	1	12	0	1	1
58	1	12	0	1	1
59	1	12	0	1	1
60	1	12	0	1	1
61	1	1	1	1	1
62	1	1	1	1	1
63	1	1	1	1	1

Table 7-5. Clock Modes (Dividers for Clocks 0–7)

MODE	CK0DF	CK1DF	CK2DF	CK3DF	CK4DF	CK5DF	CK6DF	CK7DF
0	1	2	4	1	2	1	1	1
1	1	2	4	1	2	1	1	1
2	1	2	4	1	2	1	1	1
3	1	2	4	1	2	1	1	1
4	1	2	4	1	2	1	1	1
5	1	2	4	1	2	1	1	1
6	1	2	4	1	2	1	1	1
7	1	2	4	1	2	1	1	1
8	1	2	4	1	2	1	1	1
9	1	2	4	1	2	1	1	1
10	1	2	4	1	2	1	1	1
11	1	2	4	1	2	1	1	1
12	1	2	4	1	5	1	1	1
13	1	2	4	1	5	1	1	1
14	1	2	4	1	2	1	1	1
15	1	2	4	1	2	1	1	1
16	1	2	4	1	2	1	1	1
17	1	2	4	1	2	1	1	1
18	1	2	4	1	2	1	1	1
19	1	2	4	1	2	1	1	1
20	1	2	4	1	2	1	1	1
21	1	2	4	1	2	1	1	1
22	1	2	4	1	2	1	1	1
23	1	2	4	1	2	1	1	1
24	1	2	4	1	2	1	1	1
25	1	2	4	1	2	1	1	1
26	1	2	4	1	2	1	1	1
27	1	2	4	1	2	1	1	1
28	1	2	4	1	2	1	1	1
29	1	2	4	1	2	1	1	1
30	1	2	4	1	2	1	1	1
31	1	2	4	1	2	1	1	1
32	1	2	4	1	2	1	1	1
33	1	2	4	1	2	1	1	1
34	1	2	4	1	2	1	1	1
35	1	2	4	1	2	1	1	1
36	1	2	4	1	2	1	1	1
37	1	2	4	1	2	1	1	1
38	1	2	4	1	2	1	1	1
39	1	2	4	1	2	1	1	1
40	1	2	4	1	2	1	1	1
41	1	2	4	1	2	1	1	1
42	1	2	4	1	2	1	1	1
43	1	2	4	1	2	1	1	1
44	1	2	4	1	2	1	1	1
45	1	2	4	1	2	1	1	1
46	1	2	4	1	1	1	1	1
47	1	2	4	1	1	1	1	1

Table 7-5. Clock Modes (Dividers for Clocks 0–7) (Continued)

MODE	CK0DF	CK1DF	CK2DF	CK3DF	CK4DF	CK5DF	CK6DF	CK7DF
48	1	2	4	1	2	1	1	1
49	1	2	4	1	1	1	1	1
50	1	2	4	1	1	1	1	1
51	1	2	4	1	2	1	1	1
52	1	2	4	1	1	1	1	1
53	1	2	4	1	2	1	1	1
54	1	2	4	1	2	1	1	1
55	1	2	4	1	2	1	1	1
56	1	2	4	1	2	1	1	1
57	1	2	4	1	5	1	1	1
58	1	2	4	1	2	1	1	1
59	1	2	4	1	2	1	1	1
60	1	2	4	1	5	1	1	1
61	2	4	8	2	4	1	1	1
62	3	6	12	3	3	1	1	1
63	1	2	4	1	2	1	1	1

Table 7-6. Clock Modes (Dividers for Clocks 8–12)

MODE	CK8DF	CK9DF	CK10DF	CK11DF	CK12DF
0	1	8	5	1	1
1	1	8	5	1	1
2	1	8	4	1	1
3	1	8	4	1	1
4	1	8	2	1	1
5	1	8	2	1	1
6	1	8	2	1	1
7	1	8	2	1	1
8	1	8	4	1	1
9	1	8	4	1	1
10	1	8	5	1	1
11	1	8	5	1	1
12	1	8	2	1	1
13	1	8	2	1	1
14	1	8	2	1	1
15	1	8	2	1	1
16	1	8	5	1	1
17	1	8	5	1	1
18	1	8	5	1	1
19	1	8	5	1	1
20	1	8	5	1	1
21	1	8	2	1	1
22	1	8	2	1	1
23	1	8	4	1	1
24	1	8	4	1	1
25	1	8	4	1	1
26	1	8	4	1	1
27	1	8	4	1	1
28	1	8	2	1	1

Table 7-6. Clock Modes (Dividers for Clocks 8–12) (Continued)

MODE	CK8DF	CK9DF	CK10DF	CK11DF	CK12DF
29	1	8	2	1	1
30	1	8	5	1	1
31	1	8	5	1	1
32	1	8	5	1	1
33	1	8	5	1	1
34	1	8	5	1	1
35	1	8	2	1	1
36	1	8	2	1	1
37	1	8	2	1	1
38	1	8	2	1	1
39	1	8	4	1	1
40	1	8	4	1	1
41	1	8	4	1	1
42	1	8	4	1	1
43	1	8	4	1	1
44	1	8	2	1	1
45	1	8	2	1	1
46	1	8	1	1	1
47	1	8	1	1	1
48	1	8	1	1	1
49	1	8	1	1	1
50	1	8	1	1	1
51	1	8	1	1	1
52	1	8	4	1	1
53	1	8	5	1	1
54	1	8	2	1	1
55	1	8	4	1	1
56	1	8	2	1	1
57	1	8	2	1	1
58	1	8	2	1	1
59	1	8	2	1	1
60	1	8	2	1	1
61	1	8	1	1	1
62	1	8	1	1	1
63	1	8	1	1	1

7.1.2 Clock Locking

There are two conditions under which clock locking occurs: reset initialization and clock reprogramming.

7.1.2.1 Reset Initialization

The clock circuits are initialized after the first phase of the reset configuration when the low part of the reset configuration word is loaded according to the selected clock mode.

7.1.2.2 Clock Reprogramming

The clock circuits can be reprogrammed after the reset configuration stage is completed. Use the following steps to reprogram the clocks:

1. Write the new values to the front registers: DCMRnFs, PCMRnFs and PAMRnFs. Select a new clock mode and write its parameters to the front registers. You must write values to ALL the front registers. Register values for each clock mode are listed in **Table 7-13** *Clock Mode Register Reprogramming Values*, on page 7-25.

Note: The values listed for PCMR0 assume that the GP_CTL field = 000. This field can be changed, subject to the restrictions in this chapter.

2. Set SCCR[RLKPLL] and SCCR[RLKDIV] to activate the new settings.

Note: Setting the SCCR[RLKPLL] and SCCR[RLKDIV] bits activates the relock sequence.

3. After the clock relocks, you must clear the relock bits (SCCR[RLKPLL] and SCCR[RLKDIV]) by writing zeros to them.
4. Initiate an internal soft reset.

Note: During clock relocking, all system clocks are stopped. Some circuits in the system may require an oscillating clock to be reset.Therefore, either do not assert a hard or soft reset during the relock sequence, or extend the assertion of the reset signal ($\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$) for several CLKIN cycles after CLKOUT has started oscillating again. There is no restriction regarding $\overline{\text{PORESET}}$ during relock.

7.2 Clock Programming Model

This section describes the following clock configuration registers in detail:

- System Clock Control Register (SCCR), **page 7-12**.
- PLL Clock Mode Register 0 Back/Front (PCMR0B, PCMR0F), **page 7-14**.
- PLL Clock Mode Register 1 Back/Front (PCMR1B, PCMR1F), **page 7-16**.
- PLL Clock Mode Register 2 Back/Front (PCMR2B, PCMR2F), **page 7-18**.
- Dividers Clock Mode Register0 Back/Front (DCMR0B, DCMR0F), **page 7-20**.
- Dividers Clock Mode Register1 Back/Front (DCMR1B, DCMR1F), **page 7-22**
- PLL Auxiliary Mode Register 0 Back/Front (PAMR0B, PAMR0F),
- PLL Auxiliary Mode Register 1 Back/Front (PAMR1B, PAMR0F),
- PLL Auxiliary Mode Register 2 Back/Front (PAMR2B, PAMR0F),

Note: The base address for the clock registers is 0xFFF24000.

7.2.1 System Clock Control Register (SCCR)

SCCR System Clock Control Register Offset 0x000

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CLK0 DIS	CLK1 DIS	CLK2 DIS	CLK3 DIS	CLK4 DIS	CLK5 DIS	CLK6 DIS	CLK7 DIS	CLK8 DIS	CLK9 DIS	CLK10 DIS	CLK11 DIS	CLK12 DIS	—		
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SCCR stores the general configuration of the system clocks. It is initialized only by a power-on reset. **Table 7-7** defines the SCCR bit fields.

Table 7-7. SCCR Bit Descriptions

Name	Reset	Description	Settings
— 31–18	0	Reserved. Write to zero for future compatibility.	
RLKPLL 17	0	**Relock PLLs** Setting this bit initiates PLL clock relocking according to the clock mode programmed in PCMRs and PAMRs. You must clear the bit after clock relocking is complete	0 Do not relock clocks. 1 Relock clocks.

Table 7-7. SCCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
RLKDIV 16	0	**Relock Dividers** Setting this bit initiates clock divider relocking according to the dividers clock mode programmed in DCMR[0–1]. You must clear the bit after clock relocking is complete	0 Do not relock clock dividers. 1 Relock clock dividers.
CLK0DIS 15	0	**Clock 0 Disable** Used to disable clock 0 to conserve power. **Note:** Because Clocks 0 and 1 supply clocking for the internal MBus, disabling these clocks results in loss of control over the Clock block and is not recommended.	0 Clock 0 enabled. 1 Clock 0 disabled.
CLK1DIS 14	0	**Clock 1 Disable** Used to disable clock 1 to conserve power. **Note:** Because Clocks 0 and 1 supply clocking for the internal MBus, disabling these clocks results in loss of control over the Clock block and is not recommended.	0 Clock 1 enabled. 1 Clock 1 disabled.
CLK2DIS 13	0	**Clock 2 Disable** Used to disable clock 2 to conserve power.	0 Clock 2 enabled. 1 Clock 2 disabled.
CLK3DIS 12	0	**Clock 3 Disable** Used to disable clock 3 to conserve power.	0 Clock 3 enabled. 1 Clock 3 disabled.
CLK4DIS 11	0	**Clock 4 Disable** Used to disable clock 4 to conserve power.	0 Clock 4 enabled. 1 Clock 4 disabled.
CLK5DIS 10	0	**Clock 5 Disable** Used to disable clock 5 to conserve power.	0 Clock 5 enabled. 1 Clock 5 disabled.
CLK6DIS 9	0	**Clock 6 Disable** Used to disable clock 6 to conserve power.	0 Clock 6 enabled. 1 Clock 6 disabled.
CLK7DIS 8	0	**Clock 7 Disable** Used to disable clock 7 to conserve power.	0 Clock 7 enabled. 1 Clock 7 disabled.
CLK8DIS 7	0	**Clock 8 Disable** Used to disable clock 8 to conserve power.	0 Clock 8 enabled. 1 Clock 8 disabled.
CLK9DIS 6	0	**Clock 9 Disable** Used to disable clock 3 to conserve power.	0 Clock 9 enabled. 1 Clock 9 disabled.
CLK10DIS 5	0	**Clock 10 Disable** Used to disable clock 10 to conserve power.	0 Clock 10 enabled. 1 Clock 10 disabled.
CLK11DIS 4	0	**Clock 11 Disable** Used to disable clock 11 to conserve power.	0 Clock 11 enabled. 1 Clock 11 disabled.
CLK12DIS 3	0	**Clock 12 Disable** Used to disable clock 12 to conserve power.	0 Clock 12 enabled. 1 Clock 12 disabled.
— 2–0	0	Reserved. Write to zero for future compatibility.	

7.2.2 PLL Clock Mode Register 0 (PCMR0B/PCMR0F)

PCMR0B PLL Clock Mode Register 0 Offset 0x020
PCMR0F Offset 0x030

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PD				MF					—	DIS	BYP	INPRNG	EQDLY	—	
Type: B	R															
F	W															
Reset	X	X	X	X	X	X	X	X	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
	—													GP_CTL		
Type: B	R															
F	W															
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: The reset value is determined by the value of MODCK bits. See **Table 7-13** on page 7-25 for a summary by mode.

The PLL Clock Mode Register 0 (PCMR0) stores the configuration setting for PLL0. PCMR0 is reset only by a power-on reset. Read operations access the register as PCMR0B. Write operations access the register as PCMR0F. The reset value is determined by the MODCK bits in the reset configuration word low. **Table 7-8** defines the PCMR0 bit fields.

Table 7-8. PCMR0 Bit Descriptions

Name	Reset	Description	Settings	
PD 31–28	X	**PLL Predivider** Defines the PLL predivider division factor.	0000 PD = 1. 0001 PD = 2. 0010 PD = 3. 0011 PD = 4. 0100 PD = 5. 0101 PD = 6. 0110 PD = 7. 0111 PD = 8.	1000 PD = 9. 1001 PD = 10. 1010 PD = 11. 1011 PD = 12. 1100 PD = 13. 1101 PD = 14. 1110 PD = 15. 1111 PD = 16.
MF 27–23	X	**PLL Multiplication Factor** Defines the PLL multiplication factor. **Note:** A value of 00000 is not allowed.	00000 Reserved. 00001 MF = 2. 00010 MF = 3. 00011 MF = 4. 00100 MF = 5. 00101 MF = 6. 00110 MF = 7. 00111 MF = 8. 01000 MF = 9. 01001 MF = 10. 01010 MF = 11. 01011 MF = 12. 01100 MF = 13. 01101 MF = 14. 01110 MF = 15. 01111 MF = 16.	10000 MF = 17. 10001 MF = 18. 10010 MF = 19. 10011 MF = 20. 10100 MF = 21. 10101 MF = 22. 10110 MF = 23. 10111 MF = 24. 11000 MF = 25. 11001 MF = 26. 11010 MF = 27. 11011 MF = 28. 11100 MF = 29. 11101 MF = 30. 11110 MF = 31. 11111 MF = 32.
— 22	X	Reserved. Write to zero for future compatibility.		

Table 7-8. PCMR0 Bit Descriptions (Continued)

Name	Reset	Description	Settings
DIS 21	X	**PLL0 Disable** Used to disable PLL0. The reset value is determined directly by the value of the system PLL disable bit in the reset configuration word (RCWLR[SPD]).	0 PLL0 enabled. 1 PLL0 disabled.
BYP 20	X	**PLL0 Bypass** Used to bypass PLL0.	0 PLL0 is not bypassed. 1 PLL0 is bypassed.
INPRNG 19	X	**Input Clock Range** Determines the CLKIN range. This bit reflects the value of the RCFG_CLKIN_RNG input at the deassertion of PORESET.	0 CLKIN = 25 to 66 MHz 1 CLKIN = 66 to 133 MHz
EQDLY 18	X	**Use Equivalent Delay Feedback Loop** Determines the type of feedback loop used by the PLL.	0 Use short path. System clocks are positive-edge aligned to CLKIN with non-zero delay. 1 Use equivalent delay path. System clocks are positive-edge aligned to CLKIN with zero delay.
— 17–3	X	Reserved.	
GP_CTL 2–0	X	**General-Purpose Control** These bits reflect the value of RCWLR[12–10]. They control the clock multiplexing of the PCI, DDR, and M3 memory.	0 Clock source is PLL2 1 Clock source is PLL0

7.2.3 PLL Clock Mode Register 1 (PCMR1B/PC1MR1F)

PCMR1B PLL Clock Mode Register 1 Offset 0x024
PCMR1F Offset 0x034

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PD				MF					SYNCLK	DIS	BYP	CAS	EQDLY	—	
Type: B										R						
Type: F										W						
Reset	X	X	X	X	X	X	X	X	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
	—															
Type: B										R						
Type: F										W						
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: The reset value is determined by the value of MODCK bits. See **Table 7-13** on page 7-25 for a summary by mode.

The PLL Clock Mode Register 1 (PCMR1) stores the configuration setting for PLL1. PCMR1 is reset only by a power-on reset. Read operations access the register as PCMR1B. Write operations access the register as PCMR1F. The reset value is determined by the MODCK bits in the reset configuration word low. **Table 7-9** defines the PCMR1 bit fields.

Table 7-9. PCMR1 Bit Descriptions

Name	Reset	Description	Settings	
PD 31–28	X	**PLL Predivider** Defines the PLL predivider division factor.	0000 PD = 1. 0001 PD = 2. 0010 PD = 3. 0011 PD = 4. 0100 PD = 5. 0101 PD = 6. 0110 PD = 7. 0111 PD = 8.	1000 PD = 9. 1001 PD = 10. 1010 PD = 11. 1011 PD = 12. 1100 PD = 13. 1101 PD = 14. 1110 PD = 15. 1111 PD = 16.
MF 27–23	X	**PLL Multiplication Factor** Defines the PLL multiplication factor. **Note:** A value of 00000 is not allowed.	00000 Reserved. 00001 MF = 2. 00010 MF = 3. 00011 MF = 4. 00100 MF = 5. 00101 MF = 6. 00110 MF = 7. 00111 MF = 8. 01000 MF = 9. 01001 MF = 10. 01010 MF = 11. 01011 MF = 12. 01100 MF = 13. 01101 MF = 14. 01110 MF = 15. 01111 MF = 16.	10000 MF = 17. 10001 MF = 18. 10010 MF = 19. 10011 MF = 20. 10100 MF = 21. 10101 MF = 22. 10110 MF = 23. 10111 MF = 24. 11000 MF = 25. 11001 MF = 26. 11010 MF = 27. 11011 MF = 28. 11100 MF = 29. 11101 MF = 30. 11110 MF = 31. 11111 MF = 32.

Table 7-9. PCMR1 Bit Descriptions (Continued)

Name	Reset	Description	Settings
SYNCLK 22	X	**Synchronous Clock Mode** Selects clock multiplexing used by the supported blocks.	0 Clock source is PLL1 1 Clock source is PLL0
DIS 21	X	**PLL0 Disable** Used to disable PLL1. The value of this bit is determined by the value of the core PLL disable bit in the reset configuration word (RCWLR[CPD]).	0 PLL1 enabled. 1 PLL1 disabled.
BYP 20	X	**PLL0 Bypass** Used to bypass PLL1.	0 PLL1 is not bypassed. 1 PLL1 is bypassed.
CAS 19	X	**PLL Cascade** Cascades PLL1 with PLL0 so that the output of PLL0 is the input to PLL1.	0 Input is CLKIN. 1 Input is PLL0 output (cascaded).
EQDLY 18	X	**Use Equivalent Delay Feedback Loop** Determines the type of feedback loop used by the PLL.	0 Use short path. System clocks are positive-edge aligned to CLKIN with non-zero delay. 1 Use equivalent delay path. System clocks are positive-edge aligned to CLKIN with zero delay.
— 17–0	X	Reserved.	

7.2.4 PLL Clock Mode Register 2 (PCMR2B/PCMR2F)

PCMR2B PLL Clock Mode Register 2 Offset 0x028
PCMR2F Offset 0x038

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PD				MF					—	DIS	BYP	CAS	EQDLY	—	
Type: B F	R W															
Reset	X	X	X	X	X	X	X	X	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
	—															
Type: B F	R W															
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: The reset value is determined by the value of MODCK bits. See **Table 7-13** on page 7-25 for a summary by mode.

The PLL Clock Mode Register 2 (PCMR2) stores the configuration setting for PLL2. PCMR2 is reset only by a power-on reset. Read operations access the register as PCMR2B. Write operations access the register as PCMR2F. The reset value is determined by the MODCK bits in the reset configuration word low. **Table 7-10** defines the PCMR2 bit fields.

Table 7-10. PCMR2 Bit Descriptions

Name	Reset	Description	Settings	
PD 31–28	X	**PLL Predivider** Defines the PLL predivider division factor.	0000 PD = 1. 0001 PD = 2. 0010 PD = 3. 0011 PD = 4. 0100 PD = 5. 0101 PD = 6. 0110 PD = 7. 0111 PD = 8.	1000 PD = 9. 1001 PD = 10. 1010 PD = 11. 1011 PD = 12. 1100 PD = 13. 1101 PD = 14. 1110 PD = 15. 1111 PD = 16.
MF 27–23	X	**PLL Multiplication Factor** Defines the PLL multiplication factor. **Note:** A value of 00000 is not allowed.	00000 Reserved. 00001 MF = 2. 00010 MF = 3. 00011 MF = 4. 00100 MF = 5. 00101 MF = 6. 00110 MF = 7. 00111 MF = 8. 01000 MF = 9. 01001 MF = 10. 01010 MF = 11. 01011 MF = 12. 01100 MF = 13. 01101 MF = 14. 01110 MF = 15. 01111 MF = 16.	10000 MF = 17. 10001 MF = 18. 10010 MF = 19. 10011 MF = 20. 10100 MF = 21. 10101 MF = 22. 10110 MF = 23. 10111 MF = 24. 11000 MF = 25. 11001 MF = 26. 11010 MF = 27. 11011 MF = 28. 11100 MF = 29. 11101 MF = 30. 11110 MF = 31. 11111 MF = 32.
— 22	X	Reserved. Write to zero for future compatibility.		

Table 7-10. PCMR2 Bit Descriptions (Continued)

Name	Reset	Description	Settings
DIS 21	X	**PLL2 Disable** Used to disable PLL2. The value of this bit is determined by the value of the global PLL disable bit in the reset configuration word (RCWLR[GPD]).	0 PLL2 enabled. 1 PLL2 disabled.
BYP 20	X	**PLL2 Bypass** Used to bypass PLL2.	0 PLL2 is not bypassed. 1 PLL2 is bypassed.
CAS 19	X	**PLL Cascade** Cascades PLL2 with PCI input clock or system CLKIN.	0 PLL2 input is CLKIN. 1 PLL2 input is PCI_CLK_IN.
EQDLY 18	X	**Use Equivalent Delay Feedback Loop** Determines the type of feedback loop used by the PLL.	0 Use short path. System clocks are positive-edge aligned to CLKIN with non-zero delay. 1 Use equivalent delay path. System clocks are positive-edge aligned to CLKIN with zero delay.
— 17–0	X	Reserved.	

7.2.5 Dividers Clock Mode Register 0 (DCMR0B/DCMR0F)

DCMR0B Dividers Clock Mode Register 0 Offset 0x040
DCMR0F Offset 0x050

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CK0DF				CK1DF				CK2DF				CK3DF			
Type: B F	R W															
Reset	X	X	X	X	X	X	X	X	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
	CK4DF				CK5DF				CK6DF				CK7DF			
Type: B F	R W															
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: The reset value is determined by the value of MODCK bits. See **Table 7-13** on page 7-25 for a summary by mode.

DCMR0 stores the system clock divider configuration for clocks 0–7. DCMR0 is reset only by a power-on reset. Read operations access the register as DCMR0B. Write operations access the register as DCMR0F.The reset value is determined by the MODCK bits in the reset configuration word low. **Table 7-11** defines the DCMR0 bit fields.

Table 7-11. DCMR0 Bit Descriptions

Name	Reset	Description	Settings	
CK0DF 31–28	X	**CK0 Division Factor** Defines the division factor for clock 0.	0000 CK0DF = 1. 0001 CK0DF = 2. 0010 CK0DF = 3. 0011 CK0DF = 4. 0100 CK0DF = 5. 0101 CK0DF = 6. 0110 CK0DF = 7. 0111 CK0DF = 8.	1000 CK0DF = 9. 1001 CK0DF = 10. 1010 CK0DF = 11. 1011 CK0DF = 12. 1100 CK0DF = 13. 1101 CK0DF = 14. 1110 CK0DF = 15. 1111 CK0DF = 16.
CK1DF 27–24	X	**CK1 Division Factor** Defines the division factor for clock 1.	0000 CK1DF = 1. 0001 CK1DF = 2. 0010 CK1DF = 3. 0011 CK1DF = 4. 0100 CK1DF = 5. 0101 CK1DF = 6. 0110 CK1DF = 7. 0111 CK1DF = 8.	1000 CK1DF = 9. 1001 CK1DF = 10. 1010 CK1DF = 11. 1011 CK1DF = 12. 1100 CK1DF = 13. 1101 CK1DF = 14. 1110 CK1DF = 15. 1111 CK1DF = 16.
CK2DF 23–20	X	**CK2 Division Factor** Defines the division factor for clock 2.	0000 CK3DF = 1. 0001 CK3DF = 2. 0010 CK3DF = 3. 0011 CK3DF = 4. 0100 CK3DF = 5. 0101 CK3DF = 6. 0110 CK3DF = 7. 0111 CK3DF = 8.	1000 CK3DF = 9. 1001 CK3DF = 10. 1010 CK3DF = 11. 1011 CK3DF = 12. 1100 CK3DF = 13. 1101 CK3DF = 14. 1110 CK3DF = 15. 1111 CK3DF = 16.

Table 7-11. DCMR0 Bit Descriptions (Continued)

Name	Reset	Description	Settings	
CK3DF 19–16	X	**CK3 Division Factor** Defines the division factor for clock 3.	0000 CK3DF = 1. 0001 CK3DF = 2. 0010 CK3DF = 3. 0011 CK3DF = 4. 0100 CK3DF = 5. 0101 CK3DF = 6. 0110 CK3DF = 7. 0111 CK3DF = 8.	1000 CK3DF = 9. 1001 CK3DF = 10. 1010 CK3DF = 11. 1011 CK3DF = 12. 1100 CK3DF = 13. 1101 CK3DF = 14. 1110 CK3DF = 15. 1111 CK3DF = 16.
CK4DF 15–12	X	**CK4 Division Factor** Defines the division factor for clock 4.	0000 CK4DF = 1. 0001 CK4DF = 2. 0010 CK4DF = 3. 0011 CK4DF = 4. 0100 CK4DF = 5. 0101 CK4DF = 6. 0110 CK4DF = 7. 0111 CK4DF = 8.	1000 CK4DF = 9. 1001 CK4DF = 10. 1010 CK4DF = 11. 1011 CK4DF = 12. 1100 CK4DF = 13. 1101 CK4DF = 14. 1110 CK4DF = 15. 1111 CK4DF = 16.
CK5DF 11–8	X	**CK5 Division Factor** Defines the division factor for clock 5.	0000 CK5DF = 1. 0001 CK5DF = 2. 0010 CK5DF = 3. 0011 CK5DF = 4. 0100 CK5DF = 5. 0101 CK5DF = 6. 0110 CK5DF = 7. 0111 CK5DF = 8.	1000 CK5DF = 9. 1001 CK5DF = 10. 1010 CK5DF = 11. 1011 CK5DF = 12. 1100 CK5DF = 13. 1101 CK5DF = 14. 1110 CK5DF = 15. 1111 CK5DF = 16.
CK6DF 7–4	X	**CK6 Division Factor** Defines the division factor for clock 6.	0000 CK6DF = 1. 0001 CK6DF = 2. 0010 CK6DF = 3. 0011 CK6DF = 4. 0100 CK6DF = 5. 0101 CK6DF = 6. 0110 CK6DF = 7. 0111 CK6DF = 8.	1000 CK6F = 9. 1001 CK6DF = 10. 1010 CK6DF = 11. 1011 CK6DF = 12. 1100 CK6DF = 13. 1101 CK6DF = 14. 1110 CK6DF = 15. 1111 CK6DF = 16.
CK7DF 3–0	X	**CK7 Division Factor** Defines the division factor for clock 7.	0000 CK7DF = 1. 0001 CK7DF = 2. 0010 CK7DF = 3. 0011 CK7DF = 4. 0100 CK7DF = 5. 0101 CK7DF = 6. 0110 CK7DF = 7. 0111 CK7DF = 8.	1000 CK7DF = 9. 1001 CK7DF = 10. 1010 CK7DF = 11. 1011 CK7DF = 12. 1100 CK7DF = 13. 1101 CK7DF = 14. 1110 CK7DF = 15. 1111 CK7DF = 16.

7.2.6 Dividers Clock Mode Register 1 (DCMR1B, DCMR1F)

DCMR1B Dividers Clock Mode Register 1 Offset 0x044
DCMR1F Offset 0x054

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CK8DF				CK9DF				CK10DF				CK11DF			
Type: B F	R W															
Reset	X	X	X	X	X	X	X	X	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
	CK12DF				—											
Type: B F	R W															
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: The reset value is determined by the value of MODCK bits. See **Table 7-13** on page 7-25 for a summary by mode.

DCMR1 stores the configured system clock dividers for clocks 8–12. DCMR1 is reset only by a power-on reset. Read operations access the register as DCMR1B. Write operations access the register as DCMR1F. The reset value is determined by the MODCK bits in the reset configuration word low. **Table 7-12** defines the DCMR1 bit fields.

Table 7-12. DCMR1 Bit Descriptions

Name	Reset	Description	Settings	
CK8DF 31–28	X	**CK8 Division Factor** Defines the division factor for clock 8.	0000 CK8DF = 1. 0001 CK8DF = 2. 0010 CK8DF = 3. 0011 CK8DF = 4. 0100 CK8DF = 5. 0101 CK8DF = 6. 0110 CK8DF = 7. 0111 CK8DF = 8.	1000 CK8DF = 9. 1001 CK8DF = 10. 1010 CK8DF = 11. 1011 CK8DF = 12. 1100 CK8DF = 13. 1101 CK8DF = 14. 1110 CK8DF = 15. 1111 CK8DF = 16.
CK9DF 27–24	X	**CK9 Division Factor** Defines the division factor for clock 9.	0000 CK3DF = 1. 0001 CK3DF = 2. 0010 CK3DF = 3. 0011 CK3DF = 4. 0100 CK3DF = 5. 0101 CK3DF = 6. 0110 CK3DF = 7. 0111 CK3DF = 8.	1000 CK3DF = 9. 1001 CK3DF = 10. 1010 CK3DF = 11. 1011 CK3DF = 12. 1100 CK3DF = 13. 1101 CK3DF = 14. 1110 CK3DF = 15. 1111 CK3DF = 16.
CK10DF 23–20	X	**CK10 Division Factor** Defines the division factor for clock 10.	0000 CK10DF = 1. 0001 CK10DF = 2. 0010 CK10DF = 3. 0011 CK10DF = 4. 0100 CK10DF = 5. 0101 CK10DF = 6. 0110 CK10DF = 7. 0111 CK10DF = 8.	1000 CK10DF = 9. 1001 CK10DF = 10. 1010 CK10DF = 11. 1011 CK10DF = 12. 1100 CK10DF = 13. 1101 CK10DF = 14. 1110 CK10DF = 15. 1111 CK10DF = 16.

Table 7-12. DCMR1 Bit Descriptions (Continued)

Name	Reset	Description	Settings	
CK11DF 19–16	X	**CK11 Division Factor** Defines the division factor for clock 11.	0000 CK11DF = 1. 0001 CK11DF = 2. 0010 CK11DF = 3. 0011 CK11DF = 4. 0100 CK11DF = 5. 0101 CK11DF = 6. 0110 CK11DF = 7. 0111 CK11DF = 8.	1000 CK11DF = 9. 1001 CK11DF = 10. 1010 CK11DF = 11. 1011 CK11DF = 12. 1100 CK11DF = 13. 1101 CK11DF = 14. 1110 CK11DF = 15. 1111 CK11DF = 16.
CK12DF 15–12	X	**CK12 Division Factor** Defines the division factor for clock 12.	0000 CK12DF = 1. 0001 CK12DF = 2. 0010 CK12DF = 3. 0011 CK12DF = 4. 0100 CK12DF = 5. 0101 CK12DF = 6. 0110 CK12DF = 7. 0111 CK12DF = 8.	1000 CK12DF = 9. 1001 CK12DF = 10. 1010 CK12DF = 11. 1011 CK12DF = 12. 1100 CK12DF = 13. 1101 CK12DF = 14. 1110 CK12DF = 15. 1111 CK12DF = 16.
— 11–0	X	Reserved. Write to zero for future compatibility.		

7.2.7 PLL Auxiliary Mode Register 0 (PAMR0B/PAMR0F)

PAMR0B PLL Auxiliary Mode Register 0 Offset 0x060
PAMR0F Offset 0x070

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PAMR0															
Type: B F	R W															
Reset	X	X	X	X	X	X	X	X	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
	PAMR0															
Type: B F	R W															
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: The reset value is determined by the value of MODCK bits. See **Table 7-13** on page 7-25 for a summary by mode.

The PLL Auxiliary Mode Register 0 (PAMR0) stores additional configuration setting for PLL0. PAMR0 is reset only by a power-on reset. Read operations access the register as PAMR0B. Write operations access the register as PAMR0F.The PAMR0 is reset only by a power-on reset. The reset value is determined by the MODCK bits in the reset configuration word low. Settings from this field are used only during relock as defined in **Table 7-13** on page 7-25.

7.2.8 PLL Auxiliary Mode Register 1 (PCMR1B/PC1MR1F)

PAMR1B PLL Auxiliary Mode Register 1 Offset 0x064
PAMR1F Offset 0x074

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PAMR1															
Type: B																
F										R						
										W						
Reset	X	X	X	X	X	X	X	X	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
	PAMR1															
Type: B																
F										R						
										W						
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: The reset value is determined by the value of MODCK bits. See **Table 7-13** on page 7-25 for a summary by mode.

The PLL Auxiliary Mode Register 1 (PAMR1) stores additional configuration setting for PLL1. PAMR1 is reset only by a power-on reset. Read operations access the register as PAMR1B. Write operations access the register as PAMR1F. The reset value is determined by the MODCK bits in the reset configuration word low. Settings from this field are used only during relock as defined in **Table 7-13** on page 7-25.

7.2.9 PLL Auxiliary Mode Register 2 (PAMR2B/PAMR2F)

PAMR2B PLL Auxiliary Mode Register 2 Offset 0x068
PAMR2F Offset 0x078

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PAMR2															
Type: B																
F										R						
										W						
Reset	X	X	X	X	X	X	X	X	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
	PAMR2															
Type: B																
F										R						
										W						
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: The reset value is determined by the value of MODCK bits. See **Table 7-13** on page 7-25 for a summary by mode.

The PLL Auxiliary Mode Register 2 (PAMR2) stores additional configuration setting for PLL2. PAMR2 is reset only by a power-on reset. Read operations access the register as PAMR2B. Write operations access the register as PAMR2F. The reset value is determined by the MODCK bits in the reset configuration word low. Settings from this field are used only during relock as defined in **Table 7-13** on page 7-25.

7.2.10 Values for Reprogramming Clock Modes

Table 7-13 lists the register settings to reprogram the registers for specific clock modes.

Table 7-13. Clock Mode Register Reprogramming Values

Mode	DCMR0	DCMR1	PCMR0	PCMR1	PCMR2	PAMR0	PAMR1	PAMR2
0	0x01301000	0x07400000	0x028FC000	0x05834000	0x0203C000	0x000046FF	0x0000C6FF	0x000046FF
1	0x01301000	0x07400000	0x028FC000	0x008B4000	0x0203C000	0x000046FF	0x0000C6FF	0x000046FF
2	0x01301000	0x07300000	0x028FC000	0x07034000	0x03838000	0x000046FF	0x0000C6FF	0x000046FF
3	0x01301000	0x07300000	0x028FC000	0x120B4000	0x03838000	0x000046FF	0x0000C6FF	0x000046FF
4	0x01301000	0x07100000	0x028BC000	0x05834000	0x058F8000	0x000046FF	0x0000C6FF	0x000046FF
5	0x01301000	0x07100000	0x028BC000	0x008B4000	0x058F8000	0x000046FF	0x0000C6FF	0x000046FF
6	0x01301000	0x07100000	0x028BC000	0x07034000	0x058F8000	0x000046FF	0x0000C6FF	0x000046FF
7	0x01301000	0x07100000	0x028BC000	0x120B4000	0x058F8000	0x000046FF	0x0000C6FF	0x000046FF
8	0x01301000	0x07300000	0x028FC000	0x05834000	0x03838000	0x000046FF	0x0000C6FF	0x000046FF
9	0x01301000	0x07300000	0x028FC000	0x008B4000	0x03838000	0x000046FF	0x0000C6FF	0x000046FF
10	0x01301000	0x07400000	0x028FC000	0x07034000	0x0203C000	0x000046FF	0x0000C6FF	0x000046FF
11	0x01301000	0x07400000	0x028FC000	0x120B4000	0x0203C000	0x000046FF	0x0000C6FF	0x000046FF
12	0x01304000	0x07100000	0x020FC000	0x07034000	0x02838000	0x000046FF	0x0000C6FF	0x000046FF
13	0x01304000	0x07100000	0x020FC000	0x128B4000	0x02838000	0x000046FF	0x0000C6FF	0x000046FF
14	0x01301000	0x07100000	0x028BC000	0x05834000	0x018FC000	0x000046FF	0x0000C6FF	0x000046FF
15	0x01301000	0x07100000	0x028BC000	0x008B4000	0x018FC000	0x000046FF	0x0000C6FF	0x000046FF
16	0x01301000	0x07400000	0x028BC000	0x05834000	0x048FC000	0x000046FF	0x0000C6FF	0x000046FF
17	0x01301000	0x07400000	0x010BC000	0x02834000	0x048FC000	0x000046FF	0x0000C6FF	0x000046FF
18	0x01301000	0x07400000	0x028BC000	0x05834000	0x020FC000	0x000046FF	0x0000C6FF	0x000046FF
19	0x01301000	0x07400000	0x028BC000	0x008B4000	0x048FC000	0x000046FF	0x0000C6FF	0x000046FF
20	0x01301000	0x07400000	0x028BC000	0x008B4000	0x020FC000	0x000046FF	0x0000C6FF	0x000046FF
21	0x01301000	0x07100000	0x028BC000	0x05834000	0x028F8000	0x000046FF	0x0000C6FF	0x000046FF
22	0x01301000	0x07100000	0x028BC000	0x008B4000	0x028F8000	0x000046FF	0x0000C6FF	0x000046FF
23	0x01301000	0x07300000	0x028BC000	0x05834000	0x078F8000	0x000046FF	0x0000C6FF	0x000046FF
24	0x01301000	0x07300000	0x028BC000	0x05834000	0x038F8000	0x000046FF	0x0000C6FF	0x000046FF
25	0x01301000	0x07300000	0x010BC000	0x02834000	0x038F8000	0x000046FF	0x0000C6FF	0x000046FF
26	0x01301000	0x07300000	0x028BC000	0x008B4000	0x078F8000	0x000046FF	0x0000C6FF	0x000046FF
27	0x01301000	0x07300000	0x028BC000	0x008B4000	0x038F8000	0x000046FF	0x0000C6FF	0x000046FF
28	0x01301000	0x07100000	0x028BC000	0x07034000	0x018FC000	0x000046FF	0x0000C6FF	0x000046FF
29	0x01301000	0x07100000	0x028BC000	0x120B4000	0x018FC000	0x000046FF	0x0000C6FF	0x000046FF
30	0x01301000	0x07400000	0x028BC000	0x07034000	0x048FC000	0x000046FF	0x0000C6FF	0x000046FF
31	0x01301000	0x07400000	0x010BC000	0x03834000	0x048FC000	0x000046FF	0x0000C6FF	0x000046FF
32	0x01301000	0x07400000	0x028BC000	0x07034000	0x020FC000	0x000046FF	0x0000C6FF	0x000046FF
33	0x01301000	0x07400000	0x028BC000	0x120B4000	0x048FC000	0x000046FF	0x0000C6FF	0x000046FF
34	0x01301000	0x07400000	0x028BC000	0x120B4000	0x020FC000	0x000046FF	0x0000C6FF	0x000046FF
35	0x01301000	0x07100000	0x010BC000	0x03834000	0x058F8000	0x000046FF	0x0000C6FF	0x000046FF
36	0x01301000	0x07100000	0x028BC000	0x07034000	0x028F8000	0x000046FF	0x0000C6FF	0x000046FF
37	0x01301000	0x07100000	0x010BC000	0x03834000	0x028F8000	0x000046FF	0x0000C6FF	0x000046FF
38	0x01301000	0x07100000	0x028BC000	0x120B4000	0x028F8000	0x000046FF	0x0000C6FF	0x000046FF
39	0x01301000	0x07300000	0x028BC000	0x07034000	0x078F8000	0x000046FF	0x0000C6FF	0x000046FF
40	0x01301000	0x07300000	0x028BC000	0x07034000	0x038F8000	0x000046FF	0x0000C6FF	0x000046FF
41	0x01301000	0x07300000	0x010BC000	0x03834000	0x038F8000	0x000046FF	0x0000C6FF	0x000046FF
42	0x01301000	0x07300000	0x028BC000	0x120B4000	0x078F8000	0x000046FF	0x0000C6FF	0x000046FF
43	0x01301000	0x07300000	0x028BC000	0x120B4000	0x038F8000	0x000046FF	0x0000C6FF	0x000046FF
44	0x01301000	0x07100000	0x018BC000	0x03834000	0x028F8000	0x000046FF	0x0000C6FF	0x000046FF

Table 7-13. Clock Mode Register Reprogramming Values (Continued)

Mode	DCMR0	DCMR1	PCMR0	PCMR1	PCMR2	PAMR0	PAMR1	PAMR2
45	0x01301000	0x07100000	0x018BC000	0x008B4000	0x028F8000	0x000046FF	0x0000C6FF	0x000046FF
46	0x01300000	0x07000000	0x008BC000	0x03834000	0x028F8000	0x000046FF	0x0000C6FF	0x000046FF
47	0x01300000	0x07000000	0x008BC000	0x018B4000	0x028F8000	0x000046FF	0x0000C6FF	0x000046FF
48	0x01301000	0x07000000	0x018BC000	0x02834000	0x028FC000	0x000046FF	0x0000C6FF	0x000046FF
49	0x01300000	0x07000000	0x008BC000	0x03834000	0x028F8000	0x000046FF	0x0000C6FF	0x000046FF
50	0x01300000	0x07000000	0x008BC000	0x018B4000	0x028F8000	0x000046FF	0x0000C6FF	0x000046FF
51	0x01301000	0x07000000	0x018BC000	0x02834000	0x028FC000	0x000046FF	0x0000C6FF	0x000046FF
52	0x01300000	0x07300000	0x008BC000	0x02834000	0x038F8000	0x000046FF	0x0000C6FF	0x000046FF
53	0x01301000	0x07400000	0x0583C000	0x008B4000	0x048FC000	0x000046FF	0x0000C6FF	0x000046FF
54	0x01301000	0x07100000	0x0583C000	0x008B4000	0x058F8000	0x000046FF	0x0000C6FF	0x000046FF
55	0x01301000	0x07300000	0x0583C000	0x008B4000	0x078F8000	0x000046FF	0x0000C6FF	0x000046FF
56	0x01301000	0x07100000	0x0583C000	0x120B4000	0x058F8000	0x000046FF	0x0000C6FF	0x000046FF
57	0x01304000	0x07100000	0x0483C000	0x010B4000	0x058F8000	0x000046FF	0x0000C6FF	0x000046FF
58	0x01301000	0x07100000	0x0383C000	0x07834000	0x058F8000	0x000046FF	0x0000C6FF	0x000046FF
59	0x01301000	0x07100000	0x028BC000	0x07034000	0x058F8000	0x000046FF	0x0000C6FF	0x000046FF
60	0x01304000	0x07100000	0x020BC000	0x07034000	0x058F8000	0x000046FF	0x0000C6FF	0x000046FF
61	0x13713000	0x07000000	0x001FC000	0x00134000	0x001F8000	0x000046FF	0x0000C6FF	0x000046FF
62	0x25B22000	0x07000000	0x001FC000	0x00134000	0x001F8000	0x000046FF	0x0000C6FF	0x000046FF
63	0x01301000	0x07000000	0x001FC000	0x008B4000	0x001FC000	0x000046FF	0x0000C6FF	0x000046FF

General Configuration Registers 8

The MSC8144E device includes a general configuration block that includes thirty-two 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 Configuration Register 1 (GCR1), see **page 8-2**
- General Configuration Register 2 (GCR2), see **page 8-3**
- General Status Register 1 (GSR1), see **page 8-4**
- Lynx General Configuration Register (L_GCR), see **page 8-6**
- DDR General Control Register (DDR_GCR), see **page 8-7**
- RapidIO Control Register (RIO_CR), see **page 8-8**
- SGMII Control Register (SGMII_CR), see **page 8-9**
- QUICC Engine Control Register (QECTLR), see **page 8-10**
- GPIO Input Enable Register (GIER), see **page 8-11**
- System Part and Revision ID Register (SPRIDR), see **page 8-12**
- General Configuration Register 4 (GCR4), see **page 8-13**
- General Interrupt Register 1 (GIR1), see **page 8-14**
- General Interrupt Enable Register 1 for Cores 0–3 (GIER1_[0–3]), see **page 8-16**
- General Interrupt Register 2 (GIR2), see **page 8-17**
- General Interrupt Enable Register 2 for Cores 0–3 (GIER2_[0–3]), see **page 8-19**
- General Interrupt Register 3 (GIR3), see **page 8-21**
- General Interrupt Enable Register 3 for Cores 0–3 (GIER3_[0–3]), see **page 8-22**

Note: The base address for the general configuration registers is: 0xFFF78000.

8.2 Detailed Register Descriptions

8.2.1 General Configuration Register 1 (GCR1)

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

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

Bit	23	22	21	20	19	18	17	16
	—		M2M_DDRC_IM	—				UART_STOP
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							M2_ECC_INJECT
Type	R/W							
Reset	0	0	0	0	0	0	0	0

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

GCR1 configures various general functions for the MSC8144E device. **Table 8-1** lists the GCR1 bit field descriptions.

Table 8-1. GCR1 Bit Descriptions

Name	Reset	Description	Settings
— 31–22	0	Reserved. Write to 0 for future compatibility.	
M2M_DDRC_IM 21	0	**M2M DDRC Init Mode Enable** Enables/disables the DDR controller init mode.	0 Init mode disabled. 1 Init mode enabled.
— 20–17	0	Reserved. Write to 0 for future compatibility.	
UART_STOP 16	0	**UART Stop** Stops the UART clock.	0 Normal operation. 1 UART clock stopped.
— 15–9	0	Reserved. Write to 0 for future compatibility.	
M2_ECC_INJECT 8	0	**ECC Error Injection for M2** Disables the ECC in the M2 memory.	0 M2 ECC enabled. 1 M2 ECC disabled.
— 7–5	0	Reserved. Write to 0 for future compatibility.	
TDM_PIPE_LMT 4–0	10000	**TDM Pipeline Limit** Specifies the TDM complex pipeline depth.	

8.2.2 General Configuration Register 2 (GCR2)

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

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

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

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

Bit	7	6	5	4	3	2	1	0
	CORE3_STP_EN	CORE2_STP_EN	CORE1_STP_EN	CORE0_STP_EN	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 MSC8144E device. **Table 8-2** lists the GCR2 bit field descriptions.

Table 8-2. GCR2 Bit Descriptions

Name	Reset	Description	Settings
— 31–7	0	Reserved. Write to 0 for future compatibility.	
DMA_DBG 8	0	**DMA Debug Mode Request** When set, initiates a request for the DMA controller to enter Debug mode. See **Section 14.6.16**, *DMA Debug Event Status Register (DMADESR)*, on page 14-39	0 No request. 1 DMA debug request.
CORE3_STP_EN 7	0	**Core 3 Stop Enable** Enables core 3 subsystem to stop.	0 Stop disabled. 1 Stop enabled.
CORE2_STP_EN 6	0	**Core 2 Stop Enable** Enables core 2 subsystem to stop.	0 Stop disabled. 1 Stop enabled.
CORE1_STP_EN 5	0	**Core 1 Stop Enable** Enables core 1subsystem to stop.	0 Stop disabled. 1 Stop enabled.
CORE0_STP_EN 4	0	**Core 0 Stop Enable** Enables core 0 subsystem to stop.	0 Stop disabled. 1 Stop enabled.
CORE3_DBG_REQ 3	0	**Core 3 Debug Request** Asserts a debug request to core 3.	0 No debug request. 1 Debug request.
CORE2_DBG_REQ 2	0	**Core 2 Debug Request** Asserts a debug request to core 2.	0 No debug request. 1 Debug request.

Table 8-2. GCR2 Bit Descriptions (Continued)

Name	Reset	Description	Settings
CORE1_DBG_REQ 1	0	**Core 1 Debug Request** Asserts a debug request to core 1.	0 No debug request. 1 Debug request.
CORE0_DBG_REQ 0	0	**Core 0 Debug Request** Asserts a debug request to core 0.	0 No debug request. 1 Debug request.

8.2.3 General Status Register 1 (GSR1)

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

Bit	31	30	29	28	27	26	25	24
	—			M3_EXIST	—		STOP_BS	—
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—			L2I_IDLE	CORE3_WAIT_ACK	CORE2_WAIT_ACK	CORE1_WAIT_ACK	CORE0_WAIT_ACK
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—				CORE3_STP_ACK	CORE2_STP_ACK	CORE1_STP_ACK	CORE0_STP_ACK
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—				CORE3_DBG_STS	CORE2_DBG_STS	CORE1_DBG_STS	CORE0_DBG_STS
Type	R							
Reset	0	0	0	0	0	0	0	0

GSR1 reports the status various general functions for the MSC8144E device. **Table 8-3** lists the GSR1 bit field descriptions.

Table 8-3. GSR1 Bit Descriptions

Name	Reset	Description	Settings
— 31–29	0	Reserved. Write to 0 for future compatibility.	
M3_EXIST 28	0	**M3 Exists and Has Power** Indicates that M3 exists and has power.	0 M3 does not exist or power is down. 1 M3 exists and has power.
— 27–26	0	Reserved. Write to 0 for future compatibility.	
STOP_BS 25	0	**Stop Boot Sequence** Reflects the state of the STOP_BS input signal.	0 Signal is low. 1 Signal is high
— 24–21	0	Reserved. Write to 0 for future compatibility.	
L2I_IDLE 20	0	**L2 ICache Status** Reflects the L2 ICache status.	0 L2 ICache is not in Idle mode. 1 L2 ICache is in Idle mode.

Table 8-3. GSR1 Bit Descriptions (Continued)

Name	Reset	Description	Settings
CORE3_WAIT_ACK 19	0	**Core 3 Wait Acknowledge** Reflects whether core 3 subsystem is in Wait state.	0 Core subsystem not in Wait state. 1 Core subsystem in Wait state.
CORE2_WAIT_ACK 18	0	**Core 2 Wait Acknowledge** Reflects whether core 2 subsystem is in Wait state	0 Core subsystem not in Wait state. 1 Core subsystem in Wait state.
CORE1_WAIT_ACK 17	0	**Core 1 Wait Acknowledge** Reflects whether core 1 subsystem is in Wait state.	0 Core subsystem not in Wait state. 1 Core subsystem in Wait state.
CORE0_WAIT_ACK 16	0	**Core 0 Wait Acknowledge** Reflects whether core 0 subsystem is in Wait state.	0 Core subsystem not in Wait state. 1 Core subsystem in Wait state.
— 15–12	0	Reserved. Write to 0 for future compatibility.	
CORE3_STP_ACK 11	0	**Core 3 Stop Acknowledge** Reflects whether core 3 subsystem is stopped.	0 Core subsystem not stopped. 1 Core subsystem stopped.
CORE2_STP_ACK 10	0	**Core 2 Stop Acknowledge** Reflects whether core 2 subsystem is stopped.	0 Core subsystem not stopped. 1 Core subsystem stopped.
CORE1_STP_ACK 9	0	**Core 1 Stop Acknowledge** Reflects whether core 1 subsystem is stopped.	0 Core subsystem not stopped. 1 Core subsystem stopped.
CORE0_STP_ACK 8	0	**Core 0 Stop Acknowledge** Reflects whether core 0 subsystem is stopped.	0 Core subsystem not stopped. 1 Core subsystem stopped.
— 7–4	0	Reserved. Write to 0 for future compatibility.	
CORE3_DBG_STS 3	0	**Core 3 Debug Status** Reflects the mode of core 3.	0 Not in Debug mode. 1 In Debug mode.
CORE2_DBG_STS 2	0	**Core 2 Debug Status** Reflects the mode of core 2.	0 Not in Debug mode. 1 In Debug mode.
CORE1_DBG_STS 1	0	**Core 1 Debug Status** Reflects the mode of core 1.	0 Not in Debug mode. 1 In Debug mode.
CORE0_DBG_STS 0	0	**Core 0 Debug Status** Reflects the mode of core 0.	0 Not in Debug mode. 1 In Debug mode.

8.2.4 Lynx General Configuration Register (L_GCR)

L_GCR Lynx General Configuration Register **Offset 0x0C**

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
	LANE_D_XMIT_3S	LANE_C_XMIT_3S	LANE_B_XMIT_3S	LANE_A_XMIT_3S	—		RX_IMPCAL_STOP	TX_IMPCAL_STOP
Type	R/W							
Reset	1	1	1	1	0	0	0	0

L_GCR controls part of the SERDES operation for the MSC8144E device. The register is reset on a Hard reset. Write accesses can only be performed in Supervisor mode. **Table 8-4** lists the L_GCR bit field descriptions.

Table 8-4. L_GCR Bit Descriptions

Name	Reset	Description	Settings
— 31–8	0	Reserved. Write to 0 for future compatibility.	
LANE_D_XMIT_3S 7	1	**Lane D Transmit Tristate** Setting this bit forces the RapidIO lane D output to tristate.	0 Normal output. 1 Output tristated.
LANE_C_XMIT_3S 6	1	**Lane C Transmit Tristate** Setting this bit forces the RapidIO lane C output to tristate.	0 Normal output. 1 Output tristated.
LANE_B_XMIT_3S 5	1	**Lane B Transmit Tristate** Setting this bit forces the RapidIO lane B output to tristate.	0 Normal output. 1 Output tristated.
LANE_A_XMIT_3S 4	1	**Lane A Transmit Tristate** Setting this bit forces the RapidIO lane A output to tristate.	0 Normal output. 1 Output tristated.
— 3–2	0	Reserved. Write to 0 for future compatibility.	
RX_IMPCAL_STOP 1	0	**Receiver Impedance Calibration Stop** Stops receiver impedance calibration.	0 Rx impedance calibration occurs. 1 Rx impedance calibration stopped.
TX_IMPCAL_STOP 0	0	**Transmitter Impedance Calibration Stop** Stops transmitter impedance calibration.	0 Tx impedance calibration occurs. 1 Tx impedance calibration stopped.

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

DDR_GCR controls the DDR operation voltage for the MSC8144E 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
— 31–1	0	Reserved. Write to 0 for future compatibility.	
DDR_VSEL 0	0	**DDR Voltage Select** Selects the DDR voltage level.	0 2.5 V. 1 1.8 V.

8.2.6 RapidIO Control Register (RIO_CR)

RIO_CR RapidIO Control Register **Offset 0x14**

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
	—						RIO_EXTACCPL_EN	RIO_INTACCPL_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
	RIO_XMIT_EQ3	RIO_XMIT_EQ[2–0]			RIO_RECV_EQ3	—	RIO_RECV_EQ{1–0]	
Type	R/W							
Reset	0	0	1	1	0	0	0	1

RIO_CR controls various functions within the SERDES block for the MSC8144E device. The register is reset on a Hard reset. Write accesses can only be performed in Supervisor mode. **Table 8-6** lists the RIO_CR bit field descriptions. The default values are recommended.

Table 8-6. RIO_CR Bit Descriptions

Name	Reset	Description	Settings
— 31–10	0	Reserved. Write to 0 for future compatibility.	
RIO_EXTACCPL_EN 9	0	**Proper External DC Coupling Enable** Enables proper DC coupling when external coupling is used on RapidIO interface.	0 Not enabled. 1 Enabled.
RIO_INTACCPL_EN 8	1	**Internal AC Amplifier Coupling** Enables internal AC amplifier coupling for the RapidIO interface.	0 Not enabled. 1 Enabled.
RIO_XMIT_EQ3 7	0	**Transmit Equalization Selection Bus** Selects peak amplitude.	0 V_{DD}-diff-pk = pk 1 5/6 V_{DD}-diff-pk = pk
RIO_XMIT_EQ[2–0]3 6–4	011	**Equalization Selection**	000 No equalization. 001 1.09x relative amplitude. 010 1.2x relative amplitude. 011 1.33x relative amplitude. 100 1.5x relative amplitude 101 1.71x relative amplitude 110 2.0x relative amplitude 111 Reserved.

Table 8-6. RIO_CR Bit Descriptions (Continued)

Name	Reset	Description	Settings
RIO_RECV_EQ3 **3**	0	**Track Loop Centering Control** **Enables/disables the recentering algorithm.**	0 Enable recentering algorithm. 1 Disable recentering algorithm.
— **2**	0	Reserved. Write to 0 for future compatibility.	
RIO_RECV_EQ 1–0	01	**Receive Equalization Selection Bus** Selects receive equalization for the RapidIO interface.	00 No equalization. 01 2 dB of equalization. 10‘ 4 dB of equalization. 11 Reserved.

8.2.7 SGMII Control Register (SGMII_CR)

SGMII_CR SGMII Control Register **Offset 0x18**

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
	—						SGMII_ EXTACCPL_ EN	SGMII_ INTACCPL_ EN
Type	R/W							
Reset	0	0	0	0	0	0	0	1

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

SGMII_CR controls various functions within the SERDES block for the MSC8144E device. The register is reset on a Hard reset. Write accesses can only be performed in Supervisor mode. **Table 8-7** lists the SGMII_CR bit field descriptions.

Table 8-7. SGMII_CR Bit Descriptions

Name	Reset	Description	Settings
— 31–10	0	Reserved. Write to 0 for future compatibility.	
SGMII_EXTACCPL_EN 9	0	**Proper External DC Coupling Enable** Enables proper DC coupling when external coupling is used for SGMII.	0 Not enabled. 1 Enabled.
SGMII_INTACCPL_EN 8	1	**Internal AC Amplifier Coupling** Enables internal AC amplifier coupling for SGMII.	0 Not enabled. 1 Enabled.

Table 8-7. SGMII_CR Bit Descriptions (Continued)

Name	Reset	Description	Settings
SGMII_XMIT_EQ 7–4	0011	**Transmit Equalization Selected** Selects transmit equalization for SGMII.	
SGMII_RECV_EQ 3–0	0001	**Receive Equalization Selected** Selects receive equalization for SGMII.	

8.2.8 QUICC Engine Control Register (QECTLR)

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

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
	—				UTP_RX_M	UTP_TX_M	—	
Type	R/W							
Reset	0	0	0	0	0	0	0	0

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

QECTLR controls various functions within the QUICC Engine module block for the MSC8144E device. The register is reset on a Hard reset. Write accesses can only be performed in Supervisor mode. **Table 8-8** lists the QECTLR bit field descriptions.

Table 8-8. QECTLR Bit Descriptions

Name	Reset	Description	Settings
— 31–12	0	Reserved. Write to 0 for future compatibility.	
UTP_RX_M 11	0	**UTOPIA RX as Master** Configures the UTOPIA Rx pins as master.	0 UTOPIA Rx not master. 1 UTOPIA Rx is master.
UTP_TX_M 10	0	**UTOPIA TX as Master** Configures the UTOPIA Tx pins as master.	0 UTOPIA Tx not master. 1 UTOPIA Tx is master.
— 9–8	0	Reserved. Write to 0 for future compatibility.	

Table 8-8. QECTLR Bit Descriptions (Continued)

Name	Reset	Description	Settings
UTP_ENABLE 7	0	**UTOPIA Enable** Enables the UTOPIA interface.	0 UTOPIA not enabled. 1 UTOPIA enabled.
— 6–4	0	Reserved. Write to 0 for future compatibility.	
ENET_SGMII_MODE1 3	0	**SGMII Mode for Ethernet Controller 2** Selects SGMII mode for Ethernet controller 2.	0 SGMII not selected. 1 SGMII selected.
ENET_SGMII_MODE0 2	0	**SGMII Mode for Ethernet Controller 1** Selects SGMII mode for Ethernet controller 1.	0 SGMII1 not selected. 1 SGMII1 selected.
— 1–0	0	Reserved. Write to 0 for future compatibility.	

8.2.9 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 8-9** lists the GIER bit field descriptions.

Table 8-9. 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 I/O signals corresponding to the bit index number.	0 Input is disabled. 1 Input is enabled.

8.2.10 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	0	0	0	1	1	0	0	0

Bit	23	22	21	20	19	18	17	16
	PARTID							
Type	R							
Reset	0	0	0	0	0	1	1	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	1	0	0	0	0	0

SPRIDR configures various general functions for the MSC8144E device. **Table 8-10** lists the SPRIDR bit field descriptions.

Table 8-10. SPRIDR Bit Descriptions for Mask Set M31H

Name	Reset	Description	Settings
PARTID 31–16	0	**Part Identification** Mask programmed to indicate the device number.	0x1807
REVID 15–0	0	**Revision Identification** Mask programmed with the revision number for this part.	0x0020

8.2.11 General Configuration Register 4 (GCR4)

GCR4 General Configuration Register4 **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. **Table 8-8** lists the GCR4 bit field descriptions.

Table 8-11. 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.
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.

Table 8-11. GCR4 Bit Descriptions (Continued)

Name	Reset	Description	Settings
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.

8.2.12 General Interrupt Register 1 (GIR1)

GIR1 General Interrupt Register 1 Offset 0x40

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
	—				VNMI_3	VNMI_2	VNMI_1	VNMI_0
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—				M2_3_ECC	M2_2_ECC	M2_1_ECC	M2_0_ECC
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIR1 includes the interrupt status of ECC events of M2 and the virtual NMIs. Those bits are sticky and cleared by writing 1. The GIR1 is 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 8-12. GIR1 Bit Descriptions

Name	Description	Settings
— 31–12	Reserved. Write to zero for future compatibility.	
VNMI_3 11	**Virtual NMI 3** Asserted when VNMI_3 is activated	0 Interrupt not asserted 1 Interrupt asserted
VNMI_2 10	**Virtual NMI 2** Asserted when VNMI_2 is activated	0 Interrupt not asserted 1 Interrupt asserted
VNMI_1 9	**Virtual NMI 1** Asserted when VNMI_1 is activated	0 Interrupt not asserted 1 Interrupt asserted
VNMI_0 8	**Virtual NMI 0** Asserted when VNMI_0 is activated	0 Interrupt not asserted 1 Interrupt asserted
— 7–4	Reserved. Write to zero for future compatibility.	
M2_3_ECC 3	**M2 Block 3 ECC Error Interrupt** Asserted when ECC error is reported by M2_3	0 Interrupt not asserted 1 Interrupt asserted
M2_2_ECC 2	**M2 Block 2 ECC Error Interrupt** Asserted when ECC error is reported by M2_2	0 Interrupt not asserted 1 Interrupt asserted
M2_1_ECC 1	**M2 Block 1 ECC Error Interrupt** Asserted when ECC error is reported by M2_1	0 Interrupt not asserted 1 Interrupt asserted
M2_0_ECC 0	**M2 Block 0 ECC Error Interrupt** Asserted when ECC error is reported by M2_0	0 Interrupt not asserted 1 Interrupt asserted

8.2.13 General Interrupt Register 1 (GIER1_x)

GIER1_0 ‘General Interrupt Enable Register 1 for Cores 0–3 Offset 0x44
GIER1_1 Offset 0x48
GIER1_2 Offset 0x4C
GIER1_3 Offset 0x50

Bit	30	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
	—				M2_3_ECC_EN	M2_2_ECC_EN	M2_1_ECC_EN	M2_0_ECC_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIER1_[0–3] includes interrupt enable bits of ECC events of M2 for cores 0–3. 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 8-13. GIER1_n Bit Descriptions

Name	Description	Settings
— 31–4	Reserved. Write to zero for future compatibility.	
M2_3_ECC_EN 3	**M2 Block 3 ECC Error Enable**	0 Interrupt disabled 1 Interrupt enabled
M2_2_ECC_EN 2	**M2 Block 2 ECC Error Enable**	0 Interrupt disabled 1 Interrupt enabled
M2_1_ECC_EN 1	**M2 Block 1 ECC Error Enable**	0 Interrupt disabled 1 Interrupt enabled
M2_0_ECC_EN 0	**M2 Block 0 ECC Error Enable**	0 Interrupt disabled 1 Interrupt enabled

8.2.14 General Interrupt Register 2 (GIR2)

GIR2 General Interrupt Register 2 Offset 0x54

Bit	31	30	29	28	27	26	25	24
	—	—	SWT4	SWT3	SWT2	SWT1	SWT0	OCN_ERR
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PCI_ERR	DDR_ERR	DMA_ERR	—	CE_IECC	CE_DECC	TDM_P1ECC	TDM_P0ECC
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TDM7_TERR	TDM7_RERR	TDM6_TERR	TDM6_RERR	TDM5_TERR	TDM5_RERR	TDM4_TERR	TDM4_RERR
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TDM3_TERR	TDM3_RERR	TDM2_TERR	TDM2_RERR	TDM1_TERR	TDM1_RERR	TDM0_TERR	TDM0_RERR
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIR2 includes interrupt status of some events within MSC8144E that are rare. Those bits are not sticky but only sample the events. The GIR2 register is reset on a hard reset event. All bits will be deasserted on reset.

Table 8-14. GIR2 Bit Descriptions

Name	Description	Settings
— 31–30	Reserved. Write to zero for future compatibility.	
SWT4 29	**Software Watchdog Timer 4 Interrupt** Reflects SWT 4 interrupt	0 Interrupt not asserted 1 Interrupt asserted
SWT3 28	**Software Watchdog Timer 3 Interrupt** Reflects SWT 3 interrupt	0 Interrupt not asserted 1 Interrupt asserted
SWT2 27	**Software Watchdog Timer 2 Interrupt** Reflects SWT 2 interrupt	0 Interrupt not asserted 1 Interrupt asserted
SWT1 26	**Software Watchdog Timer 1 Interrupt** Reflects SWT 1 interrupt	0 Interrupt not asserted 1 Interrupt asserted
SWT0 25	**Software Watchdog Timer 0 Interrupt** Reflects SWT 0 interrupt	0 Interrupt not asserted 1 Interrupt asserted
OCN_ERR 24	**OCeaN-to-MBus Error Interrupt** Reflects OCeaN error interrupt	0 Interrupt not asserted 1 Interrupt asserted
PCI_ERR 23	**PCI Error Interrupt** Reflects PCI error interrupt	0 Interrupt not asserted 1 Interrupt asserted
DDR_ERR 22	**DDR Error Interrupt** Reflects DDR error interrupt	0 Interrupt not asserted 1 Interrupt asserted
DMA_ERR 21	**DMA Error Interrupt** Reflects DMA error interrupt	0 Interrupt not asserted 1 Interrupt asserted

Table 8-14. GIR2 Bit Descriptions

Name	Description	Settings
— 20	Reserved. Write to zero for future compatibility.	
QE_IECC 19	**QUICC Engine IMEM ECC Error Interrupt** Reflects ECC error interrupt of the QUICC Engine IMEM	0 Interrupt not asserted 1 Interrupt asserted
QE_DECC 18	**QUICC Engine DRAM ECC Error Interrupt** Reflects ECC error interrupt of the QUICC Engine DRAM	0 Interrupt not asserted 1 Interrupt asserted
TDM_P1ECC 17	**TDM[4–7] Parity Error Interrupt** Reflects parity error interrupt of TDM4, TDM5, TDM6 or TDM7	0 Interrupt not asserted 1 Interrupt asserted
TDM_P0ECC 16	**TDM[0–3] Parity Error Interrupt** Reflects parity error interrupt of TDM0, TDM1, TDM2 or TDM3	0 Interrupt not asserted 1 Interrupt asserted
TDM7_TERR 15	**TDM7 Transmit Error Interrupt** Reflects TDM7 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM7_RERR 14	**TDM7 Receive Error Interrupt** Reflects TDM7 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM6_TERR 13	**TDM6 Transmit Error Interrupt** Reflects TDM6 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM6_RERR 12	**TDM6 Receive Error Interrupt** Reflects TDM6 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM5_TERR 11	**TDM5 Transmit Error Interrupt** Reflects TDM5 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM5_RERR 10	**TDM5 Receive Error Interrupt** Reflects TDM5 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM4_TERR 9	**TDM4 Transmit Error Interrupt** Reflects TDM4 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM4_RERR 8	**TDM4 Receive Error Interrupt** Reflects TDM4 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM3_TERR 7	**TDM3 Transmit Error Interrupt** Reflects TDM3 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM3_RERR 6	**TDM3 Receive Error Interrupt** Reflects TDM3 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM2_TERR 5	**TDM2 Transmit Error Interrupt** Reflects TDM2 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM2_RERR 4	**TDM2 Receive Error Interrupt** Reflects TDM2 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM1_TERR 3	**TDM1 Transmit Error Interrupt** Reflects TDM1 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM1_RERR 2	**TDM1 Receive Error Interrupt** Reflects TDM1 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM0_TERR 1	**TDM0 Transmit Error Interrupt** Reflects TDM0 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM0_RERR 0	**TDM0 Receive Error Interrupt** Reflects TDM0 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted

8.2.15 General Interrupt Enable Register 2 (GIER2_x)

GIER2_0 General Interrupt Enable Register 2 for Cores 0–3 Offset 0x58
GIER2_1 Offset 0x5C
GIER2_2 Offset 0x60
GIER2_3 Offset 0x64

Bit	31	30	29	28	27	26	25	24
	—	—	SWT4_EN	SWT3_EN	SWT2_EN	SWT1_EN	SWT0_EN	OCN_ERR_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PCI_ERR_EN	DDR_ERR_EN	DMA_ERR_EN	—	CE_IECC_EN	CE_DECC_EN	TDM_P1ECC_EN	TDM_P0ECC_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TDM7_TERR_EN	TDM7_RERR_EN	TDM6_TERR_EN	TDM6_RERR_EN	TDM5_TERR_EN	TDM5_RERR_EN	TDM4_TERR_EN	TDM4_RERR_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TDM3_TERR_EN	TDM3_RERR_EN	TDM2_TERR_EN	TDM2_RERR_EN	TDM1_TERR_EN	TDM1_RERR_EN	TDM0_TERR_EN	TDM0_RERR_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIER2_[0–3] include interrupt enable bits for cores 0–3 for some events that rarely occur. The GIER2_[0–3] are 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 8-15. GIER2_x Bit Descriptions

Name	Description	Settings
— 31–30	Reserved. Write to zero for future compatibility.	
SWT4_EN 29	**SWT 4 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT3_EN 28	**SWT 3 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT2_EN 27	**SWT 2 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT1_EN 26	**SWT 1 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT0_EN 25	**SWT 0 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
OCN_ERR_EN 24	**OCeaN Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
PCI_ERR_EN 23	**PCI Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled

Table 8-15. GIER2_x Bit Descriptions

Name	Description	Settings
DDR_ERR_EN 22	**DDR Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
DMA_ERR_EN 21	**DMA Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
— 20	Reserved. Write to zero for future compatibility.	
CE_IECC_EN 19	**ECC Error Interrupt of the QUICC Engine IMEM Enable**	0 Interrupt disabled 1 Interrupt enabled
CE_DECC_EN 18	**ECC Error Interrupt of the QUICC Engine DRAM Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM_P1ECC_EN 17	**Parity Error Interrupt of TDM[4–7] Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM_P0ECC_EN 16	**Parity Error Interrupt of TDM[0–3] Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM7_TERR_EN 15	**TDM7 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM7_RERR_EN 14	**TDM7 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM6_TERR_EN 13	**TDM6 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM6_RERR_EN 12	**TDM6 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM5_TERR_EN 11	**TDM5 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM5_RERR_EN 10	**TDM5 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM4_TERR_EN 9	**TDM4 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM4_RERR_EN 8	**TDM4 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM3_TERR_EN 7	**TDM3 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM3_RERR_EN 6	**TDM3 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM2_TERR_EN 5	**TDM2 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM2_RERR_EN 4	**TDM2 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM1_TERR_EN 3	**TDM1 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM1_RERR_EN 2	**TDM1 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM0_TERR_EN 1	**TDM0 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled

Table 8-15. GIER2_x Bit Descriptions

Name	Description	Settings
TDM0_RERR_EN 0	**TDM0 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled

8.2.16 General Interrupt Register 3 (GIR3)

GIR3 General Interrupt Register 3 Offset 0x68

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**
	—	—	—	—	PM	L2ICS_WP	L2ICS_OV	L2ICM_WP
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	L2ICM_OV	CLS2_WP	CLS2_OV	CLS1_ERR	CLS1_WP	CLS1_OV	CLS0_WP	CLS0_OV
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIR3 includes interrupt status of some debug/profiling events within MSC8144E. 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 8-16. GIR2 Bit Descriptions

Name	Description	Settings
— 31–12	Reserved. Write to zero for future compatibility.	
PM 11	**Performance Monitor Interrupt** Reflects the performance monitor interrupt	0 Interrupt not asserted 1 Interrupt asserted
L2ICS_WP 10	**L2 ICache Target CLASS Watchpoint Interrupt** Reflects L2 ICache target CLASS watch-point interrupt	0 Interrupt not asserted 1 Interrupt asserted
L2ICS_OV 9	**L2 ICache Target CLASS Overrun Interrupt** Reflects L2 ICache target Class overrun interrupt	0 Interrupt not asserted 1 Interrupt asserted
L2ICM_WP 8	**L2 ICache Initiator CLASS Watchpoint Interrupt** Reflects L2 ICache initiator Class watchpoint interrupt	0 Interrupt not asserted 1 Interrupt asserted
L2ICM_OV 7	**L2 ICache Initiator CLASS Overrun Interrupt** Reflects L2 ICache initiator Class overrun interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS2_WP 6	**CLASS2 Watchpoint Interrupt** Reflects CLASS2 watchpoint interrupt	0 Interrupt not asserted 1 Interrupt asserted

Table 8-16. GIR2 Bit Descriptions

Name	Description	Settings
CLS2_OV 5	**CLASS2 Overrun Interrupt** Reflects CLASS2 overrun interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS1_ERR 4	**CLASS1 Error Interrupt** Reflects CLASS1 Error Interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS1_WP 3	**CLASS1 Watchpoint Interrupt** Reflects CLASS1 watchpoint interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS1_OV 2	**CLASS1 Overrun Interrupt** Reflects CLASS1 overrun interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS0_WP 1	**CLASS0 Watchpoint Interrupt** Reflects Class0 watchpoint interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS0_OV 0	**CLASS0 Overrun Interrupt** Reflects CLASS0 overrun interrupt	0 Interrupt not asserted 1 Interrupt asserted

8.2.17 General Interrupt Enable Register 3 (GIER3_x)

GIER3_0 General Interrupt Enable Register 3 for Cores 0–3 Offset 0x6C
GIER3_1 Offset 0x70
GIER3_2 Offset 0x74
GIER3_3 Offset 0x78

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
	—	—	—	—	PM_EN	L2ICS_WP_EN	L2ICS_OV_EN	L2ICM_WP_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	L2ICM_OV_EN	CLS2_WP_EN	CLS2_OV_EN	CLS1_ERR_EN	CLS1_WP_EN	CLS1_OV_EN	CLS0_WP_EN	CLS0_OV_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIER3_[0–3] include interrupt enable bits for cores 0–3 for debug/profiling events within MSC8144E. GIER3_[0–3] 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 8-17. GIER2_[0–3] Bit Descriptions

Name	Description	Settings
— 31–30	Reserved. Write to zero for future compatibility.	
PM_EN 11	**Performance Monitor Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
L2ICS_WP_EN 10	**L2 ICache Target CLASS Watchpoint Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
L2ICS_OV_EN 9	**L2 ICache Target CLASS Overrun Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
L2ICM_WP_EN 8	**L2 ICache Initiator CLASS Watchpoint Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
L2ICM_OV_EN 7	**L2 ICache Initiator CLASS Overrun Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS2_WP_EN 6	**CLASS2 Watchpoint Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS2_OV_EN 5	**CLASS2 Overrun Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS1_ERR_EN 4	**CLASS1 Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS1_WP_EN 3	**CLASS1 Watchpoint Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS1_OV_EN 2	**CLASS1 Overrun Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS0_WP_EN 1	**CLASS0 Watchpoint Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS0_OV_EN 0	**CLASS0 Overrun Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled

Memory Map 9

This section describes the memory map of MSC8144E.

The MSC8144E incorporates five address spaces:

- Shared memory (M2, M3, DDR, PCI, QUICC Engine subsystem, and boot ROM) address space.
- SC3400 DSP core subsystem internal address space is accessible only by the SC3400 core. Each SC3400 core can access its own internal address space.
- QUICC Engine module internal address space accessible only by internal QUICC Engine module elements.
- Control Configuration and Status Registers (CCSR) address space.
- L2 ICache address space accessible only by the instruction interfaces of the SC3400 DSP core subsystem.

Note: All addresses in this chapter are given as hexadecimal values.

9.1 Shared Memory Address Space

The shared memory address space resides within addresses 0x40000000–0xFEFFFFFF. It includes the M2 memory, M3 memory, DDR memory, PCI, QUICC Engine subsystem, and the boot ROM.

Table 9-1. Shared Memory Address Space

Address	Purpose	Size in Bytes
40000000–5FFFFFFF	DDR Memory	512 M
60000000–BFFFFFFF	Reserved	1.5 G
C0000000–C007FFFF	M2 Memory	512 K
C0080000–CFFFFFFF	Reserved	255.5 M
D0000000–D09FFFFF	M3 Memory	10 M
D0A00000–DFFFFFFF	Reserved	246 M
E0000000–E7FFFFFF	PCI	128 M
E8000000–FEDFFFFF	Reserved	366 M
FEE00000–FEE3FFFF	QUICC Engine Subsystem	256 K
FEE40000–FEEFFFFF	Reserved (QUICC Engine Subsystem)	768 K
FEF00000–FEF17FFF	Boot ROM	96 K
FEF18000–FEFFFFFF	Reserved	928 K

The PCI address space size includes 4 GB for PCI memory space, 4 GB for PCI I/O space, and also 256-byte sections of PCI configuration space. The MSC8144E initiators can access all of the PCI addresses using the defined 128 MB window with the help of the PCI outbound windows. The last 16 bytes of the PCI address space (0xE7FFFFF0–0xE7FFFFFF) are used as the PCI Configuration Access Registers memory map. Therefore, the PCI outbound window is 128 MB – 16 B.

9.2 SC3400 DSP core subsystem Internal Address Space

Each SC3400 core can access the internal address space of its DSP core subsystem. The SC3400 internal address space is located from address 0xFF000000–0xFFF0FFFF (15 MB + 64 KB). **Table 9-2** lists details for the DSP core subsystem internal address space.

Table 9-2. SC3400 DSP Core Subsystem Internal Address Space

Address	Purpose	Size (Bytes)	Remarks
FF000000–FFEFFDFF	Reserved	15 M – 512	
FFEFFE00–FFEFFFFF	OCE	512	User/Supervisor[1]
FFF00000[2]–FFF003FF	Reserved	1K	
FFF00400–FFF007FF	EPIC	1K	Supervisor
FFF00800–FFF00BFF	Data Cache registers	1K	Supervisor
FFF00C00–FFF00FFF	Instruction Cache registers	1K	Supervisor
FFF01000–FFF05FFF	Reserved	20K	
FFF06000–FFF07FFF	MMU	8K	Supervisor
FFF08000[3]–FFF09FFF	MMU (continued)	8K	Supervisor
FFF0A000–FFF0A2FF	DPU	768	User/Supervisor
FFF0A300–FFF0A3FF	TIMERS	256	User/Supervisor
FFF0A400–FFF0FFFF	Reserved	23K	

Notes: **1.** Access in both User and Supervisor modes is allowed only if enabled via the EMR[EAP] bit in the core.
2. Core access to address range FFF00000–FFF07FFF stalls the core until the access is completed.
3. Core access to addresses from FFF08000 and above does not stall the core.

9.3 QUICC Engine Module Internal Address Space

The MSC8144E QUICC Engine module is mapped within a contiguous block of memory. **Table 9-3** details the QUICC Engine module address space.

Table 9-3. QUICC Engine Module Address Space

Address	Purpose	Size in Bytes
0xFEE00000–0xFEE0003F	IRAM Registers	64
0xFEE00040–0xFEE0007F	reserved	64
0xFEE00080–0xFEE000FF	Interrupt Controller	128
0xFEE00100–0xFEE001FF	RISC Configuration Registers	256

Table 9-3. QUICC Engine Module Address Space (Continued)

Address	Purpose	Size in Bytes
0xFEE00200–0xFEE003FF	reserved	512
0xFEE00400–0xFEE0043F	QUICC Engine Clock Multiplex Registers	64
0xFEE00440–0xFEE0047F	QUICC Engine Timers	64
0xFEE00480–0xFEE004DF	reserved	96
0xFEE004E0–0xFEE004EF	SPI Registers	16
0xFEE004F0–0xFEE0063F	reserved	352
0xFEE00640–0xFEE0067F	Baud-Rate Generator Registers	64
0xFEE00680–0xFEE01FFF	reserved	6528
0xFEE02000–0xFEE021FF	UCC 1 Registers	512
0xFEE02200–0xFEE023FF	UCC 3 Registers	512
0xFEE02400–0xFEE025FF	UCC 5 Registers	512
0xFEE02600–0xFEE027FF	reserved	512
0xFEE02800–0xFEE029FF	MII1 Bridge Register	512
0xFEE02A00–0xFEE02BFF	MII2 Bridge Register	512
0xFEE02C00–0xFEE02DFF	reserved	512
0xFEE02E00–0xFEE02FFF	MultiPHY Controller Registers	512
0xFEE03000–0xFEE03FFF	reserved	4096
0xFEE04000–0xFEE0407F	Serial DMA Registers	128
0xFEE04080–0xFEE040FF	Debug Registers	128
0xFEE04100–0xFEE041FF	RISC1 Special Registers (trap and breakpoint)	256
0xFEE04200–0xFEE042FF	RISC1 Special Registers (trap and breakpoint)	256
0xFEE04300–0xFEE044FF	reserved	512
0xFEE04500–0xFEE045FF	Test Registers	256
0xFEE04600–0xFEE0467F	RISC1 Trace Buffer Registers	128
0xFEE04680–0xFEE046FF	RISC2 Trace Buffer Registers	128
0xFEE04700–0xFEE0FFFF	reserved	46.25 K
0xFEE10000–0xFEE1BFFF	Multi-User RAM	48 K
0xFEE1C000–0xFEE3FFFF	reserved	144 K

9.4 CCSR Address Space

The MSC8144E CCSR is mapped within a contiguous block of memory. The size of the CCSR in MSC8144E is 956 KB. **Table 9-4** details the CCSR address space.

Table 9-4. CCSR Address Space

Address	Purpose	Size (Bytes)
FFF10000–FFF103FF	DMA	1 K
FFF10400–FFF17FFF	Reserved	31 K
FFF18000–FFF18FFF	CLASS0 Registers	4 K
FFF19000–FFF19FFF	CLASS1 Registers	4 K
FFF1A000–FFF1AFFF	CLASS2 Registers	4 K
FFF1B000–FFF1FFFF	Reserved	20 K

Table 9-4. CCSR Address Space (Continued)

Address	Purpose	Size (Bytes)
FFF20000–FFF21FFF	DDR Controller	8 K
FFF22000–FFF23FFF	Reserved	8 K
FFF24000–FFF2407F	Clock	128
FFF24080–FFF247FF	reserved	2 K – 128
FFF24800–FFF248FF	Reset	256
FFF24900–FFF24BFF	reserved	768
FFF24C00–FFF24CFF	I^2C	256
FFF24D00–FFF24FFF	reserved	768
FFF25000–FFF250FF	Watch Dog Timers	256
FFF25100–FFF251FF		256
FFF25200–FFF252FF		256
FFF25300–FFF253FF		256
FFF25400–FFF254FF		256
FFF25500–FFF25FFF	reserved	3K – 256
FFF26000–FFF260FF	Timers	256
FFF26100–FFF261FF		256
FFF26200–FFF262FF		256
FFF26300–FFF263FF		256
FFF26400–FFF26FFF	Reserved	3K
FFF27000–FFF270FF	GIC Registers	256
FFF27100–FFF271FF	Hardware Semaphores	256
FFF27200–FFF272FF	GPIO Registers	256
FFF27300–FFF29FFF	Reserved	12 K – 768
FFF2A000–FFF2AFFF	L2 ICache Registers	4K
FFF2B000–FFF2BFFF		4K
FFF2C000–FFF2C01F		32
FFF2C020–FFF2FFFF	Reserved	16 K – 32
FFF30000–FFF33FFF	TDM0 Registers	16 K
FFF34000–FFF37FFF	TDM1 Registers	16 K
FFF38000–FFF3BFFF	TDM2 Registers	16 K
FFF3C000–FFF3FFFF	TDM3 Registers	16 K
FFF40000–FFF43FFF	TDM4 Registers	16 K
FFF44000–FFF47FFF	TDM5 Registers	16 K
FFF48000–FFF4BFFF	TDM6 Registers	16 K
FFF4C000–FFF4FFFF	TDM7 Registers	16 K
FFF50000–FFF77FFF	Reserved	160 K
FFF78000–FFF7807F	General Configuration Registers	128
FFF78080–FFF79FFF	Reserved	8 K – 128
FFF7A000–FFF7A1FF	PCI Registers	512
FFF7A200–FFF7EFFF	Reserved	20 K – 512
FFF7F000–FFF7F03F	UART Registers	64
FFF7F040–FFF7FFFF	Reserved	4 K – 64

Table 9-4. CCSR Address Space (Continued)

Address	Purpose	Size (Bytes)
FFF80000–FFF9FFFF	RapidIO Registers	128 K
FFFA0000–FFFA00FF	OCN Crossbar Switch Registers	256
FFFA0100–FFFA0FFF	Reserved	4 K – 256
FFFA1000–FFFA103F	OCN Crossbar Switch to MBus	64
FFFA1040–FFFA1FFF	Reserved	4 K
FFFA2000–FFFA3FFF	Dedicated DMA Controller Registers	8 K
FFFA4000–FFFC00FF	Reserved	28 K + 256
FFFC0100–FFFC01FF	Performance Monitor Registers	256
FFFC0200–FFFCFFFF	Reserved	64 K – 512
FFFD0000–FFFDEFFF	Security Engine Registers	60 K
FFFDF000–FFFFEFFF	Reserved	128 K

9.5 L2 ICache Address Space

The MSC8144E L2 ICache address space is divided into two sections: cacheable and non-cacheable. It is accessible only by the instruction bus interface of each of the SC3400 DSP core subsystems. The address space in the range of 0x00000000–0xFFFFFFFF can be divided into two zones (cacheable and non-cacheable) dynamically by programming the L2 ICache. The user can program one contiguous address space as cacheable (in 4 KB resolution); the rest of the address space is non-cacheable (that is, if the user programs the address space in the range of A0–A1 as cacheable, then the ranges 0x00000000–A0 and A1–0xFFFFFFFF are non-cacheable). The L2 ICache, as an initiator, sees the address space as listed in **Table 9-5**.

Table 9-5. L2 ICache Address Space (as an initiator)

Address	Purpose	Size (Bytes)
00000000–3FFFFFFF	Reserved	1 G
40000000–FEFFFFFF	Shared memory address space	3 G – 16 M
FF000000–FFF0FFFF	Reserved	15 M + 64 K
FFF10000–FFFFEFFF	CCSR address space	956 K
FFFFF000–FFFFFFFF	Reserved	4K

9.6 SC3400 (Data) View of the System Address Space

Table 9-7. describes the system address space as seen by the SC3400 data MBus interface.

Table 9-6. SC3400 (Data) View of the System Address Space

Address	Purpose	Size (Bytes)
00000000–3FFFFFFF	Reserved	1 G
40000000–FEFFFFFF	Shared Memory Address Space	3 G – 16 M
FF000000–FFF0FFFF	SC3400 DSP core subsystem Internal Address Space	15 M + 64 K

Table 9-6. SC3400 (Data) View of the System Address Space

Address	Purpose	Size (Bytes)
FFF10000–FFFFEFFF	CCSR Address Space	956 K
FFFFF000–FFFFFFFF	Reserved	4 K

9.7 Peripherals View of the System Address Space

Table 9-7 describes the system address space as seen by the MSC8144E peripherals (RapidIO, JTAG, QUICC Engine subsystem, TDM, DMA–both MBus interfaces).

Table 9-7. Peripherals View of the System Address Space

Address	Purpose	Size (Bytes)
00000000–3FFFFFFF	Reserved	1 G
40000000–FEFFFFFF	Shared Memory Address Space	3 G – 16 M
FF000000–FFF0FFFF	Reserved	15M + 64K
FFF10000–FFFFEFFF	CCSR Address Space	956 K
FFFFF000–FFFFFFFF	Reserved	4 K

An external initiator (to the MSC8144E device) can generate accesses to the system address space using the:

- JTAG with direct addressing.
- The RapidIO using the RapidIO inbound address translation.

9.8 PCI View of the System Address Space

Table 9-8. describes the system address space as seen by the PCI.

Table 9-8. PCI View of the System Address Space

Address	Purpose	Size (Bytes)
00000000–3FFFFFFF	Reserved	1 G
40000000–5FFFFFFF	DDR Memory	512 M
60000000–BFFFFFFF	Reserved	1.5 G
C0000000–C007FFFF	M2 Memory	512 K
C0080000–CFFFFFFF	Reserved	255.5 M
D0000000–D09FFFFF	M3 Memory	10 M
D0A00000–FEDFFFFF	Reserved	740 M
FEE00000–FEE3FFFF	QUICC Engine subsystem	256 K
FEE40000–FEEFFFFF	Reserved (QUICC Engine subsystem)	768 K
FEF00000–FEF17FFF	Boot ROM	96 K

Table 9-8. PCI View of the System Address Space

Address	Purpose	Size (Bytes)
FEF18000–FFF0FFFF	Reserved	16M–32K
FFF10000–FFFFEFFF	CCSR Address Space	956K
FFFFF000–FFFFFFFF	Reserved	4K

An external initiator (to the MSC8144E device) can generate accesses to the system address space using the PCI inbound address translation.

Note: To guarantee normal operation of the device, you cannot program the inbound and outbound windows of the PCI so that they overlap. Such programming can cause a PCI transaction that enters the device to be regenerated by the device into the PCI bus again. This condition may result in improper device operation and is not permitted.

9.9 Consolidated Memory Map

Table 9-9. Consolidated Memory Map

Address	Name/Status	Acronym	Reference
0x00000000–0x3FFFFFFF	Reserved		
0x40000000–0xFEFFFFFF	Shared memory		
• 0x40000000–0x5FFFFFFF	DDR memory		
• 0x60000000–0xBFFFFFFF	reserved		
• 0xC0000000–0xC007FFFF	M2 Memory		
• 0xC0080000–0xCFFFFFFF	reserved		
• 0xD0000000–0xD09FFFFF	M3 Memory		
• 0xD0A00000–0xDFFFFFFF	reserved		
• 0xE0000000–0xE7FFFFFF	PCI Address Space (last 16 B used for Configuration Access Registers)		
– 0xE0000000–0xE7FFFFEF	PCI address space		
– 0xE7FFFFF0	PCI Configuration Address Register	CONFIG_ADDRESS	**page 15-20**
– 0xE7FFFFF4	PCI Configuration Data Register	CONFIG_DATA	**page 15-21**
– 0xE7FFFFF8	PCI Interrupt Acknowledge Register	PCI_INT_ACK	**page 15-22**
– 0xE7FFFFFC–0xE7FFFFFF	reserved		
• 0xE8000000–0xFEDFFFFF	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
• 0xFEE00000–0xFEE3FFFF	QUICC Engine Subsystem (See the *QUICC Engine Block Reference Manual* with Protocol Interworking (QEIWRM) for register 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	QUICC Engine Interrupt Priority Register (YCC)	CIPYCC	
– 0xFEE00098	QUICC Engine Interrupt Priority Register (WCC)	CIPWCC	
– 0xFEE0009C	QUICC Engine Interrupt Priority Register (ZCC)	CIPZCC	
– 0xFEE000A0	QUICC Engine System Interrupt Mask Register	CIMR	
– 0xFEE000A4	QUICC Engine RISC Interrupt Mask Register	CRIMR	
– 0xFEE000A8	QUICC Engine System Interrupt Control Register	CICNR	
– 0xFEE000AC–0xFEE000AF	reserved		
– 0xFEE000B0	QUICC Engine System Interrupt Priority Register for RISC Tasks A	CIPRTA	
– 0xFEE000B4–0xFEE000BB	reserved		
– 0xFEE000BC	QUICC Engine System RISC Interrupt Control Register	CRICR	
– 0xFEE000C0–0xFEE000DF	reserved		
– 0xFEE000E0	QUICC Engine High System Interrupt Vector Register	CHIVEC	
– 0xFEE000E4–0xFEE000FF	reserved		
– 0xFEE00100	QUICC Engine Command Register	CECR	
– 0xFEE00104–0xFEE00107	reserved		
– 0xFEE00108	QUICC Engine Command Data Register	CECDR	
– 0xFEE0010C–0xFEE0011B	reserved		
– 0xFEE0011C	QUICC Engine Time-Stamp Control Register	CETSCR	
– 0xFEE00120–0xFEE0012F	reserved		
– 0xFEE00130	QUICC Engine Virtual Task Event Register	CEVTER	
– 0xFEE00134	QUICC Engine Virtual Task Mask Register	CEVTMR	
– 0xFEE00138	QUICC Engine RAM Control Register	CERCR	
– 0xFEE0013C–0xFEE001B7	reserved		
– 0xFEE001B8	QUICC Engine Microcode Revision Number Register	CEURNR	

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFEE001BC–0xFEE003FF	reserved		
– 0xFEE00400	CMX General Clock Route Register	CMXGCR	
– 0xFEE00404–0xFEE0040F	reserved		
– 0xFEE00410	CMX Clock Route Register 1	CMXUCR1	
– 0xFEE00414	CMX Clock Route Register 2	CMXUCR2	
– 0xFEE00414–0xFEE0041F	reserved		
– 0xFEE00420	CMX UPC Clock Route Register	CMXUPCR	
– 0xFEE00424–0xFEE0064F	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	
– 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		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 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		
– 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–0xFEE02179	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	

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 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		
– 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	

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 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	
– 0xFEE02310–0xFEE0231B	reserved		
– 0xFEE0231C	Ethernet 2 MAC Test Register	E2EMTR	
– 0xFEE02320	Ethernet 2 MII Management Configuration Register	MIIMCFG3	
– 0xFEE02324	Ethernet 2 MII Management Command Register	MIIMCOM3	
– 0xFEE02328	Ethernet 2 MII Management Address Register	MIIMADD3	
– 0xFEE0232C	Ethernet 2 MII Management Control Register	MIIMCON3	
– 0xFEE02330	Ethernet 2 MII Management Status Register	MIIMSTAT3	
– 0xFEE02334	Ethernet 2 MII Management Indication Register	MIIMIND3	
– 0xFEE02338	Ethernet 2 Interface Control Register	IFCTL3	
– 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–0xFEE02379	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	

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 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	
– 0xFEE023C4–0xFEE023FF	reserved		
– 0xFEE02400	UCC 5 Mode Register	GUMR5	
– 0xFEE02404	UCC 5 Protocol Specific Mode Register	UPSMR5	
– 0xFEE02408	UCC 5 Transmit-on-Demand Register	UTODR5	
– 0xFEE0240A–0xFEE0240F	reserved		
– 0xFEE02410	UCC 5 Event Register	UCCE5	
– 0xFEE02414	UCC 5 Mask Register	UCCM5	
– 0xFEE02418	UCC 5 Status Register	UCCS5	
– 0xFEE02419–0xFEE0241F	reserved		
– 0xFEE02420	UCC 5 Receive l FIFO Base	URFB5	
– 0xFEE02424	UCC 5 Receive FIFO Size	URFS5	
– 0xFEE02426–0xFEE02427	reserved		
– 0xFEE02428	UCC 5 Receive FIFO Emergency Threshold	URFET5	
– 0xFEE0242A	UCC 5 Receive FIFO Special Emergency Threshold	URFSET5	
– 0xFEE0242C	UCC 5 Transmit FIFO Base	UTFB5	
– 0xFEE02430	UCC 5 Transmit FIFO Size	UTFS5	
– 0xFEE02432–0xFEE02433	reserved		
– 0xFEE02434	UCC 5 Transmit FIFO Emergency Threshold	UTFET5	
– 0xFEE02436–0xFEE02437	reserved		
– 0xFEE02438	UCC 5 Transmit FIFO Transmit Threshold	UTFTT5	
– 0xFEE0243A–0xFEE0243B	reserved		
– 0xFEE0243C	UCC 5 Transmit Polling Timer	UFPT5	
– 0xFEE0243E–0xFEE0243F	reserved		
– 0xFEE02440	UCC 5 Retry Counter	URTRY5	
– 0xFEE02444–0xFEE0248F	reserved		
– 0xFEE02490	UCC 5 General Extended Mode Register	GUEMR5	
– 0xFEE02494–0xFEE027FF	reserved		
– 0xFEE02800	MIIGSK Configuration Register 1	MIIGSK1_CFGR	
– 0xFEE02804–0xFEE02807	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFEE02808	MIIGSK Enable Register 1	MIIGSK1_ENR	
– 0xFEE0280C	MIIGSK SMII SYNC Direction Register 1	MIIGSK1_SMII_SYNCDIR	
– 0xFEE02810	MIIGSK SMII Transmit Inter Frame Bits Register 1	MIIGSK1_TIFBR	
– 0xFEE02814	MIIGSK SMII Receive Inter Frame Bits Register 1	MIIGSK1_RIFBR	
– 0xFEE02818	MIIGSK SMII Expected Receive Inter Frame Bits Register 1	MIIGSK1_ERIFBR	
– 0xFEE0281C	MIIGSK Interrupt Event Register 1	MIIGSK1_IEVENT	
– 0xFEE02820	MIIGSK Interrupt Mask Register 1	MIIGSK1_IMASK	
– 0xFEE02824–0xFEE029FF	reserved		
– 0xFEE02A00	MIIGSK Configuration Register 2	MIIGSK2_CFGR	
– 0xFEE02A04–0xFEE02A07	reserved		
– 0xFEE02A08	MIIGSK Enable Register 2	MIIGSK2_ENR	
– 0xFEE02A0C	MIIGSK SMII SYNC Direction Register 2	MIIGSK2_SMII_SYNCDIR	
– 0xFEE02A10	MIIGSK SMII Transmit Inter Frame Bits Register 2	MIIGSK2_TIFBR	
– 0xFEE02A14	MIIGSK SMII Receive Inter Frame Bits Register 2	MIIGSK2_RIFBR	
– 0xFEE02A18	MIIGSK SMII Expected Receive Inter Frame Bits Register 2	MIIGSK2_ERIFBR	
– 0xFEE02A1C	MIIGSK Interrupt Event Register 2	MIIGSK2_IEVENT	
– 0xFEE02A20	MIIGSK Interrupt Mask Register 2	MIIGSK2_IMASK	
– 0xFEE02A24–0xFEE02DFF	reserved		
– 0xFEE02E00	UPC General Configuration Register	UPGCR	
– 0xFEE02E02–0xFEE02E03	reserved		
– 0xFEE02E04	UPC Last PHY Address	UPLPA	
– 0xFEE02E06–0xFEE02E07	reserved		
– 0xFEE02E08	UPC HEC Register	UPHEC	
– 0xFEE02E0C	UPC UCC Configuration Register	UPUC	
– 0xFEE02E10	UPC Device 1 Configuration Register	UPDC1	
– 0xFEE02E14–0xFEE02E2F	reserved		
– 0xFEE02E30	UPC Device 1 Transmit Rate Select High	UPDRS1H	
– 0xFEE02E34	UPC Device 1 Transmit Rate Select Low	UPDRS1L	
– 0xFEE02E38–0xFEE02E4F	reserved		
– 0xFEE02E50	UPC Device 1 Receive Port Priority Register	UPDRP1	
– 0xFEE02E54–0xFEE02E5F	reserved		
– 0xFEE02E60	UPC Device 1 Event	UPDE1	
– 0xFEE02E64–0xFEE02E6F	reserved		
– 0xFEE02E70	UPC Device 1 Internal Rate Configuration Register	UPRP1	

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFEE02E72–0xFEE02E7F	reserved		
– 0xFEE02E80	Device 1 Transmit Internal Rate 1	UPTIRR1_1	
– 0xFEE02E82	Device 1 Transmit Internal Rate 2	UPTIRR1_2	
– 0xFEE02E84	Device 1 Transmit Internal Rate 3	UPTIRR1_3	
– 0xFEE02E86	Device 1 Transmit Internal Rate 4	UPTIRR1_4	
– 0xFEE02E88–0xFEE02E9F	reserved		
– 0xFEE02EA0	Device 1 Port Enable Register	UPER1	
– 0xFEE02EA4–0xFEE03EFF	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	
– 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		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
0xFF000000–0xFFF0FFFF	DSP core subsystem internal memory space		
• 0xFF000000–0xFFEFFDFF	reserved		
• 0xFFEFFE00–0xFFEFFFFF	OCE		
– 0xFFEFFE00	OCE Status Register	ESR	OCE RM
– 0xFFEFFE04	OCE Monitor and Control Register	EMCR	OCE RM
– 0xFFEFFE08	OCE Receive Register (LSBs)	ERCVL	OCE RM
– 0xFFEFFE0C	OCE Receive Register (MSBs)	ERCVH	OCE RM
– 0xFFEFFE10	OCE Transmit Register (LSBs)	ETRSMTL	OCE RM
– 0xFFEFFE14	OCE Transmit Register (MSBs)	ETRSMTH	OCE RM
– 0xFFEFFE18	EE Signals Control Register	EE_CTRL	OCE RM
– 0xFFEFFE1C	Exception PC Register	PC_EXCP	OCE RM
– 0xFFEFFE20	PC of next execution set	PC_NEXT	OCE RM
– 0xFFEFFE24	PC of last execution set	PC_LAST	OCE RM
– 0xFFEFFE28	PC Breakpoint Detection Register	PC_DETECT	OCE RM
– 0xFFEFFE2C	PC MMU	PC_MMU	OCE RM
– 0xFFEFFE30	Core Revision Register	COREREV	OCE RM
– 0xFFEFFE34	Core Configuration Register	CORECFG	OCE RM
– 0xFFEFFE38	OCE Configuration Register (LSBs)	ECFGL	OCE RM
– 0xFFEFFE3C	OCE Configuration Register (MSBs)	ECFGH	OCE RM
– 0xFFEFFE40	EDCA0 Control Register	EDCA0_CTRL	OCE RM
– 0xFFEFFE44	EDCA1 Control Register	EDCA1_CTRL	OCE RM
– 0xFFEFFE48	EDCA2 Control Register	EDCA2_CTRL	OCE RM
– 0xFFEFFE4C	EDCA3 Control Register	EDCA3_CTRL	OCE RM
– 0xFFEFFE50	EDCA4 Control Register	EDCA4_CTRL	OCE RM
– 0xFFEFFE54	EDCA5 Control Register	EDCA5_CTRL	OCE RM
– 0xFFEFFE58–0xFFEFFE5F	reserved		
– 0xFFEFFE60	EDCA0 Reference Value A	EDCA0_REFA	OCE RM
– 0xFFEFFE64	EDCA1 Reference Value A	EDCA1_REFA	OCE RM
– 0xFFEFFE68	EDCA2 Reference Value A	EDCA2_REFA	OCE RM
– 0xFFEFFE6C	EDCA3 Reference Value A	EDCA3_REFA	OCE RM
– 0xFFEFFE70	EDCA4 Reference Value A	EDCA4_REFA	OCE RM
– 0xFFEFFE74	EDCA5 Reference Value A	EDCA5_REFA	OCE RM
– 0xFFEFFE78–0xFFEFFE7F	reserved		
– 0xFFEFFE80	EDCA0 Reference Value B	EDCA0_REFB	OCE RM
– 0xFFEFFE84	EDCA1 Reference Value B	EDCA1_REFB	OCE RM
– 0xFFEFFE88	EDCA2 Reference Value B	EDCA2_REFB	OCE RM
– 0xFFEFFE8C	EDCA3 Reference Value B	EDCA3_REFB	OCE RM
– 0xFFEFFE90	EDCA4 Reference Value B	EDCA4_REFB	OCE RM

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFEFFE94	EDCA5 Reference Value B	EDCA5_REFB	OCE RM
– 0xFFEFFE98–0xFFEFFE9F	reserved		
– 0xFFEFFEA0	EDCA0 Reference Register	EDCA0_REFID	OCE RM
– 0xFFEFFEA4	EDCA1 Reference Register	EDCA1_REFID	OCE RM
– 0xFFEFFEA8	EDCA2 Reference Register	EDCA2_REFID	OCE RM
– 0xFFEFFEAC	EDCA3 Reference Register	EDCA3_REFID	OCE RM
– 0xFFEFFEB0	EDCA4 Reference Register	EDCA4_REFID	OCE RM
– 0xFFEFFEB4	EDCA5 Reference Register	EDCA5_REFID	OCE RM
– 0xFFEFFEB8–0xFFEFFEBF	reserved		
– 0xFFEFFEC0	EDCA0 Mask Register	EDCA0_MASK	OCE RM
– 0xFFEFFEC4	EDCA1 Mask Register	EDCA1_MASK	OCE RM
– 0xFFEFFEC8	EDCA2 Mask Register	EDCA2_MASK	OCE RM
– 0xFFEFFEC	EDCA3 Mask Register	EDCA3_MASK	OCE RM
– 0xFFEFFED0	EDCA4 Mask Register	EDCA4_MASK	OCE RM
– 0xFFEFFED4	EDCA5 Mask Register	EDCA5_MASK	OCE RM
– 0xFFEFFED8–0xFFEFFEDF	reserved		
– 0xFFEFFEE0	EDCD Control Register	EDCD_CTRL	OCE RM
– 0xFFEFFEE4	EDCD Reference Register	EDCD_REF	OCE RM
– 0xFFEFFEE8	EDCD Mask Register	EDCD_MASK	OCE RM
– 0xFFEFFEEC–0xFFEFFEFF	reserved		
– 0xFFEFFF00	OCE Counter Control Register	ECNT_CTRL	OCE RM
– 0xFFEFFF04	OCE Counter Value	ECNT_VAL	OCE RM
– 0xFFEFFF08	OCE Extension Counter Value	ECNT_EXT	OCE RM
– 0xFFEFFF0C–0xFFEFFF1F	reserved		
– 0xFFEFFF20	OCE Selector Control Register	ESEL_CTRL	OCE RM
– 0xFFEFFF24	OCE Selector DM Mask	ESEL_DM	OCE RM
– 0xFFEFFF28	OCE Selector DI Mask	ESEL_DI	OCE RM
– 0xFFEFFF2C–0xFFEFFF2F	reserved		
– 0xFFEFFF30	OCE Selector Enable TB Mask	ESEL_ETB	OCE RM
– 0xFFEFFF34	OCE Selector Disable TB Mask	ESEL_DTB	OCE RM
– 0xFFEFFF3C–0xFFEFFF3F	reserved		
– 0xFFEFFF40	Trace Buffer Control Register	TB_CTRL	OCE RM
– 0xFFEFFF44	Trace Buffer Read Pointer	TB_RD	OCE RM
– 0xFFEFFF48	Trace Buffer Write Pointer	TB_WR	OCE RM
– 0xFFEFFF4C	Trace Buffer	TB_BUFF	OCE RM
– 0xFFEFFF50–0xFFEFFFF7	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFEFFFF8	Core Command Register	CORE_CMD	OCE RM
– 0xFFEFFFFC	No Register Selected	NOREG	OCE RM
• 0xFFF00000–0xFFF003FF	reserved		
• 0xFFF00400–0xFFF007FF	EPIC. See the *MSC8144 SC3400 DSP Core Subsystem Reference Manual* for details.		
– 0xFFF00400	EPIC Interrupt priority level Register 0	P_IPL0	
– 0xFFF00404	EPIC Interrupt priority level Register 1	P_IPL1	
– 0xFFF00408	EPIC Interrupt priority level Register 2	P_IPL2	
– 0xFFF0040C	EPIC Interrupt priority level Register 3	P_IPL3	
– 0xFFF00410	EPIC Interrupt priority level Register 4	P_IPL4	
– 0xFFF00414	EPIC Interrupt priority level Register 5	P_IPL5	
– 0xFFF00418	EPIC Interrupt priority level Register 6	P_IPL6	
– 0xFFF0041C	EPIC Interrupt priority level Register 7	P_IPL7	
– 0xFFF00420	EPIC Interrupt priority level Register 8	P_IPL8	
– 0xFFF00424	EPIC Interrupt priority level Register 9	P_IPL9	
– 0xFFF00428	EPIC Interrupt priority level Register 10	P_IPL10	
– 0xFFF0042C	EPIC Interrupt priority level Register 11	P_IPL11	
– 0xFFF00430	EPIC Interrupt priority level Register 12	P_IPL12	
– 0xFFF00434	EPIC Interrupt priority level Register 13	P_IPL13	
– 0xFFF00438	EPIC Interrupt priority level Register 14	P_IPL14	
– 0xFFF0043C	EPIC Interrupt priority level Register 15	P_IPL15	
– 0xFFF00440	EPIC Interrupt priority level Register 16	P_IPL16	
– 0xFFF00444	EPIC Interrupt priority level Register 17	P_IPL17	
– 0xFFF00448	EPIC Interrupt priority level Register 18	P_IPL18	
– 0xFFF0044C	EPIC Interrupt priority level Register 19	P_IPL19	
– 0xFFF00450	EPIC Interrupt priority level Register 20	P_IPL20	
– 0xFFF00454	EPIC Interrupt priority level Register 21	P_IPL21	
– 0xFFF00458	EPIC Interrupt priority level Register 22	P_IPL22	
– 0xFFF0045C	EPIC Interrupt priority level Register 23	P_IPL23	
– 0xFFF00460	EPIC Interrupt priority level Register 24	P_IPL24	
– 0xFFF00464	EPIC Interrupt priority level Register 25	P_IPL25	
– 0xFFF00468	EPIC Interrupt priority level Register 26	P_IPL26	
– 0xFFF0046C	EPIC Interrupt priority level Register 27	P_IPL27	

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF00470	EPIC Interrupt priority level Register 28	P_IPL28	
– 0xFFF00474	EPIC Interrupt priority level Register 29	P_IPL29	
– 0xFFF00478	EPIC Interrupt priority level Register 30	P_IPL30	
– 0xFFF0047C	EPIC Interrupt priority level Register 31	P_IPL31	
– 0xFFF00480	EPIC Interrupt priority level Register 32	P_IPL32	
– 0xFFF00484	EPIC Interrupt priority level Register 33	P_IPL33	
– 0xFFF00488	EPIC Interrupt priority level Register 34	P_IPL34	
– 0xFFF0048C	EPIC Interrupt priority level Register 35	P_IPL35	
– 0xFFF00490	EPIC Interrupt priority level Register 36	P_IPL36	
– 0xFFF00494	EPIC Interrupt priority level Register 37	P_IPL37	
– 0xFFF00498	EPIC Interrupt priority level Register 38	P_IPL38	
– 0xFFF0049C	EPIC Interrupt priority level Register 39	P_IPL39	
– 0xFFF004A0	EPIC Interrupt priority level Register 40	P_IPL40	
– 0xFFF004A4	EPIC Interrupt priority level Register 41	P_IPL41	
– 0xFFF004A8	EPIC Interrupt priority level Register 42	P_IPL42	
– 0xFFF004AC	EPIC Interrupt priority level Register 43	P_IPL43	
– 0xFFF004B0	EPIC Interrupt priority level Register 44	P_IPL44	
– 0xFFF004B4	EPIC Interrupt priority level Register 45	P_IPL45	
– 0xFFF004B8	EPIC Interrupt priority level Register 46	P_IPL46	
– 0xFFF004BC	EPIC Interrupt priority level Register 47	P_IPL47	
– 0xFFF004C0	EPIC Interrupt priority level Register 48	P_IPL48	
– 0xFFF004C4	EPIC Interrupt priority level Register 49	P_IPL49	
– 0xFFF004C8	EPIC Interrupt priority level Register 50	P_IPL50	
– 0xFFF004CC	EPIC Interrupt priority level Register 51	P_IPL51	
– 0xFFF004D0	EPIC Interrupt priority level Register 52	P_IPL52	
– 0xFFF004D4	EPIC Interrupt priority level Register 53	P_IPL53	
– 0xFFF004D8	EPIC Interrupt priority level Register 54	P_IPL54	
– 0xFFF004DC	EPIC Interrupt priority level Register 55	P_IPL55	
– 0xFFF004E0	EPIC Interrupt priority level Register 56	P_IPL56	
– 0xFFF004E4	EPIC Interrupt priority level Register 57	P_IPL57	
– 0xFFF004E8	EPIC Interrupt priority level Register 58	P_IPL58	
– 0xFFF004EC	EPIC Interrupt priority level Register 59	P_IPL59	
– 0xFFF004F0	EPIC Interrupt priority level Register 60	P_IPL60	
– 0xFFF004F4	EPIC Interrupt priority level Register 61	P_IPL61	

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF004F8	EPIC Interrupt priority level Register 62	P_IPL62	
– 0xFFF004FC	EPIC Interrupt priority level Register 63	P_IPL63	
– 0xFFF00500	EPIC Edge/Level trigger Register 0	P_ELR0	
– 0xFFF00504	EPIC Edge/Level trigger Register 1 (P_ELR1)	P_ELR1	
– 0xFFF00508	EPIC Edge/Level trigger Register 2 (P_ELR2)	P_ELR2	
– 0xFFF0050C	EPIC Edge/Level trigger Register 3 (P_ELR3)	P_ELR3	
– 0xFFF00510	EPIC Edge/Level trigger Register 4	P_ELR4	
– 0xFFF00514	EPIC Edge/Level trigger Register 5	P_ELR5	
– 0xFFF00518	EPIC Edge/Level trigger Register 6	P_ELR6	
– 0xFFF0051C	EPIC Edge/Level trigger Register 7	P_ELR7	
– 0xFFF00520	EPIC Interrupt Pending Register 0	P_IPR0	
– 0xFFF00524	EPIC Interrupt Pending Register 1	P_IPR1	
– 0xFFF00528	EPIC Interrupt Pending Register 2	P_IPR2	
– 0xFFF0052C	EPIC Interrupt Pending Register 3	P_IPR3	
– 0xFFF00530	EPIC Interrupt Pending Register 4	P_IPR4	
– 0xFFF00534	EPIC Interrupt Pending Register 5	P_IPR5	
– 0xFFF00538	EPIC Interrupt Pending Register 6	P_IPR6	
– 0xFFF0053C	EPIC Interrupt Pending Register 7	P_IPR7	
– 0xFFF00540	EPIC Enable/Disable interrupts register 0	P_ENDIS0	
– 0xFFF00544	EPIC Enable/Disable interrupts register 1	P_ENDIS1	
– 0xFFF00548	EPIC Enable/Disable interrupts register 2	P_ENDIS2	
– 0xFFF0054C	EPIC Enable/Disable interrupts register 3	P_ENDIS3	
– 0xFFF00550	EPIC Enable/Disable interrupts register 4	P_ENDIS4	
– 0xFFF00554	EPIC Enable/Disable interrupts register 5	P_ENDIS5	
– 0xFFF00558	EPIC Enable/Disable interrupts register 6	P_ENDIS6	
– 0xFFF0055C	EPIC Enable/Disable interrupts register 7	P_ENDIS7	
– 0xFFF00560–0xFFF00563	reserved		
– 0xFFF00564	EPIC Double Edge Interrupt Index	P_DE	
– 0xFFF00568–0xFFF007FF	reserved		
• 0xFFF00800–0xFFF00BFF	DCache Registers. See the *MSC8144 SC3400 DSP Core Subsystem Reference Manual* for details		
– 0xFFF00800	Data Cache Control Register 0	DC_CR0	
– 0xFFF00804	Data Cache Control Register 1	DC_CR1	
– 0xFFF00808	Data Cache Control Register 2	DC_CR2	
– 0xFFF0080C–0xFFF0081A	reserved		
– 0xFFF00820	Data Cache LRM State Register	DC_LRM	
– 0xFFF00824	Data Cache Extended Tag State Register	DC_ET	
– 0xFFF00828	Data Cache Valid Dirty State Register	DC_VD	
– 0xFFF0082C	Data Cache Debug Data Register	DC_DBG_DATA	
– 0xFFF00830	Data Cache Debug Access Register	DC_DBG_ACS	

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF00834–0xFFF00BFF	reserved		
• 0xFFF00C00–0xFFF00FFF	ICache Registers. See the *MSC8144 SC3400 DSP Core Subsystem Reference Manual* for details.		
– 0xFFF00C00	Instruction Cache Control Register 0	IC_CR0	
– 0xFFF00C04	Instruction Cache Control Register 1	IC_CR1	
– 0xFFF00C08	Instruction Cache Control Register 2	IC_CR2	
– 0xFFF00C0C–0xFFF00C1F	reserved		
– 0xFFF00C20	Instruction Cache LRM Status Register	IC_LRM	
– 0xFFF00C24	Instruction Cache Extended TAG State Register	IC_ET	
– 0xFFF00C28	Instruction Cache Valid State Register (IC_V)	IC_V	
– 0xFFF00C2C	Instruction Cache Debug Data Register	IC_DBG_DATA	
– 0xFFF00C30	Instruction Cache Debug Access Register	IC_DBG_ACS	
– 0xFFF00C34–0xFFF00FFF	reserved		
• 0xFFF01000–0xFFF05FFF	reserved		
• 0xFFF06000–0xFFF09FFF	MMU. See the *MSC8144 SC3400 DSP Core Subsystem Reference Manual* for details		
– 0xFFF06000	MMU Control Register	M_CR	
– 0xFFF06004	MMU Data Status Register	M_DSR	
– 0xFFF06008	MMU Data Violation Access Register	M_DVA	
– 0xFFF0600C	MMU Program Status Register	M_PSR	
– 0xFFF06010	MMU Program Violation Address Register	M_PVA	
– 0xFFF06014–0xFFF06027	reserved		
– 0xFFF06028	MMU Platform Information Register	M_PIR	
– 0xFFF0602C–0xFFF0602F	reserved		
– 0xFFF06030	MMU General Purpose Register 0	M_GPR0	
– 0xFFF06034–0xFFF06037	reserved		
– 0xFFF06038	MMU Initiator Error Status Register	M_PMESR	
– 0xFFF0603C	MMU Peripheral Violation Address Register	M_PVAR	
– 0xFFF06040	MMU Target Error Status Register	M_PSESR	
– 0xFFF06044–0xFFF060FF	reserved		
– 0xFFF06100	MMU Data Segment Descriptor Control Register	M_DSDCR	
– 0xFFF06104	MMU Current Data ID Register	M_CDID	
– 0xFFF06108	MMU Data Query Status Register	M_DQSR	
– 0xFFF0610C	MMU Data Query Physical Register	M_DQPR	
– 0xFFF06010–0xFFF06FFF	reserved		
– 0xFFF07000	MMU Program Segment Descriptor Control Register	M_PSDCR	

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF07004	MMU Current Program ID Register	M_CPID	
– 0xFFF07008	MMU Program Query Status Register	M_PQSR	
– 0xFFF0700C	MMU Program Query Physical Register	M_PQPR	
– 0xFFF07010–0xFFF07FFF	reserved		
– 0xFFF08000	MMU Program Segment Descriptor Registers A0	M_PSDA0	
– 0xFFF08004	MMU Program Segment Descriptor Registers B0	M_PSDB0	
– 0xFFF08010	MMU Program Segment Descriptor Registers A1	M_PSDA1	
– 0xFFF08014	MMU Program Segment Descriptor Registers B1	M_PSDB1	
– 0xFFF08018–0xFFF0801F	reserved		
– 0xFFF08020	MMU Program Segment Descriptor Registers A2	M_PSDA2	
– 0xFFF08024	MMU Program Segment Descriptor Registers B2	M_PSDB2	
– 0xFFF08028–0xFFF0802F	reserved		
– 0xFFF08030	MMU Program Segment Descriptor Registers A3	M_PSDA3	
– 0xFFF08034	MMU Program Segment Descriptor Registers B3	M_PSDB3	
– 0xFFF08038–0xFFF0803F	reserved		
– 0xFFF08040	MMU Program Segment Descriptor Registers A4	M_PSDA4	
– 0xFFF08044	MMU Program Segment Descriptor Registers B4	M_PSDB4	
– 0xFFF08048–0xFFF0804F	reserved		
– 0xFFF08050	MMU Program Segment Descriptor Registers A5	M_PSDA5	
– 0xFFF08054	MMU Program Segment Descriptor Registers B5	M_PSDB5	
– 0xFFF08058–0xFFF0805F	reserved		
– 0xFFF08060	MMU Program Segment Descriptor Registers A6	M_PSDA6	
– 0xFFF08064	MMU Program Segment Descriptor Registers B6	M_PSDB6	
– 0xFFF08068–0xFFF0806F	reserved		
– 0xFFF08070	MMU Program Segment Descriptor Registers A7	M_PSDA7	
– 0xFFF08074	MMU Program Segment Descriptor Registers B7	M_PSDB7	
– 0xFFF08078–0xFFF0807F	reserved		
– 0xFFF08080	MMU Program Segment Descriptor Registers A8	M_PSDA8	
– 0xFFF08084	MMU Program Segment Descriptor Registers B8	M_PSDB8	
– 0xFFF08088–0xFFF0808F	reserved		
– 0xFFF08090	MMU Program Segment Descriptor Registers A9	M_PSDA9	
– 0xFFF08094	MMU Program Segment Descriptor Registers B9	M_PSDB9	
– 0xFFF08098–0xFFF0809F	reserved		
– 0xFFF080A0	MMU Program Segment Descriptor Registers A10	M_PSDA10	

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF080A4	MMU Program Segment Descriptor Registers B10	M_PSDB10	
– 0xFFF080A8–0xFFF080AF	reserved		
– 0xFFF080B0	MMU Program Segment Descriptor Registers A11	M_PSDA11	
– 0xFFF080B4	MMU Program Segment Descriptor Registers B11	M_PSDB11	
– 0xFFF080B8–0xFFF08FFF	reserved		
– 0xFFF09000	MMU Data Segment Descriptor Registers A0	M_DSDA0	
– 0xFFF09004	MMU Data Segment Descriptor Registers B0	M_DSDB0	
– 0xFFF09008–0xFFF0900F	reserved		
– 0xFFF09010	MMU Data Segment Descriptor Registers A1	M_DSDA1	
– 0xFFF09014	MMU Data Segment Descriptor Registers B1	M_DSDB1	
– 0xFFF09018–0xFFF0901F	reserved		
– 0xFFF09020	MMU Data Segment Descriptor Registers A2	M_DSDA2	
– 0xFFF09024	MMU Data Segment Descriptor Registers B2	M_DSDB2	
– 0xFFF09028–0xFFF0902F	reserved		
– 0xFFF09030	MMU Data Segment Descriptor Registers A3	M_DSDA3	
– 0xFFF09034	MMU Data Segment Descriptor Registers B3	M_DSDB3	
– 0xFFF09038–0xFFF0903F	reserved		
– 0xFFF09040	MMU Data Segment Descriptor Registers A4	M_DSDA4	
– 0xFFF09044	MMU Data Segment Descriptor Registers B4	M_DSDB4	
– 0xFFF09048–0xFFF0904F	reserved		
– 0xFFF09050	MMU Data Segment Descriptor Registers A5	M_DSDA5	
– 0xFFF09054	MMU Data Segment Descriptor Registers B5	M_DSDB5	
– 0xFFF09058–0xFFF09FFF	reserved		
• 0xFFF0A000–0xFFF0A2FF	DPU		
– 0xFFF0A000	DPU Control Register	DP_CR	**page 25-31**
– 0xFFF0A004	DPU Status Register	DP_SR	**page 25-33**
– 0xFFF0A008	DPU Monitor Register	DP_MR	**page 25-34**
– 0xFFF0A00C	DPU PID Detection Reference Value Register	DP_RPID	**page 25-36**
– 0xFFF0A010	DPU DID Detection Reference Value Register	DP_RDID	**page 25-36**
– 0xFFF0A014–0xFFF0A01F	reserved		
– 0xFFF0A020	DPU Counter Triad A Control Register	DP_TAC	**page 25-37**
– 0xFFF0A024	DPU Counter Triad B Control Register	DP_TBC	**page 25-40**
– 0xFFF0A028–0xFFF0A02B	reserved		
– 0xFFF0A02C	DPU Counter A0 Control Register	DP_CA0C	**page 25-42**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF0A030	DPU Counter A0 Value Register	DP_CA0V	**page 25-45**
– 0xFFF0A034	DPU Counter A1 Control Register	DP_CA1C	**page 25-45**
– 0xFFF0A038	DPU Counter A1 Value Register	DP_CA1V	**page 25-48**
– 0xFFF0A03C	DPU Counter A2 Control Register	DP_CA2C	**page 25-48**
– 0xFFF0A040	DPU Counter A2 Value Register	DP_CA2V	**page 25-51**
– 0xFFF0A044–0xFFF0A053	reserved		
– 0xFFF0A054	DPU Counter B0 Control Register	DP_CB0C	**page 25-51**
– 0xFFF0A058	DPU Counter B0 Value Register	DP_CB0V	**page 25-54**
– 0xFFF0A05C	DPU Counter B1 Control Register	DP_CB1C	**page 25-54**
– 0xFFF0A060	DPU Counter B1 Value Register	DP_CB1V	**page 25-57**
– 0xFFF0A064	DPU Counter B2 Control Register	DP_CB2C	**page 25-57**
– 0xFFF0A068	DPU Counter B2 Value Register	DP_CB2V	**page 25-60**
– 0xFFF0A06C–0xFFF0A07B	reserved		
– 0xFFF0A07C	DPU Trace Control Register	DP_TC	**page 25-60**
– 0xFFF0A080	DPU VTB Start Address Register	DP_TSA	**page 25-64**
– 0xFFF0A084	DPU VTB End Address Register	DP_TEA	**page 25-65**
– 0xFFF0A088	DPU Trace Event Request Register	DP_TER	**page 25-66**
– 0xFFF0A08C	DPU Trace Write Pointer Register	DP_TW	**page 25-67**
– 0xFFF0A090	DPU Trace Data Register	DP_TD	**page 25-68**
– 0xFFF0A094–0xFFF0A2FF	reserved		
• 0xFFF0A300–0xFFF0A3FF	Core Timers. See the *MSC8144 SC3400 DSP Core Subsystem Reference Manual* for details		
– 0xFFF0A300	Timer 0 Control Register	TM_T0C	
– 0xFFF0A304	Timer 0 Value Register	TM_T0V	
– 0xFFF0A308	Timer 0 Pre-load Register	TM_T0P	
– 0xFFF0A30C	Timer 1 Control Register	TM_T1C	
– 0xFFF0A310	Timer 1 Value Register	TM_T1V	
– 0xFFF0A314	Timer 1 Pre-load Register	TM_T1P	
– 0xFFF0A318–0xFFF0A3FF	reserved		
• 0xFFF0A400–0xFFF0FFFF	reserved		
0xFFF10000–0xFFFFEFFF	CCSR		
• 0xFFF10000–0xFFF103FF	DMA		
– 0xFFF10000	DMA Buffer Descriptor Base Register 0	DMABDBR0	**page 14-24**
– 0xFFF10004	DMA Buffer Descriptor Base Register 1	DMABDBR1	**page 14-24**
– 0xFFF10008	DMA Buffer Descriptor Base Register 2	DMABDBR2	**page 14-24**
– 0xFFF1000C	DMA Buffer Descriptor Base Register 3	DMABDBR3	**page 14-24**
– 0xFFF10010	DMA Buffer Descriptor Base Register 4	DMABDBR4	**page 14-24**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF10014	DMA Buffer Descriptor Base Register 5	DMABDBR5	**page 14-24**
– 0xFFF10018	DMA Buffer Descriptor Base Register 6	DMABDBR6	**page 14-24**
– 0xFFF1001C	DMA Buffer Descriptor Base Register 7	DMABDBR7	**page 14-24**
– 0xFFF10020	DMA Buffer Descriptor Base Register 8	DMABDBR8	**page 14-24**
– 0xFFF10024	DMA Buffer Descriptor Base Register 9	DMABDBR9	**page 14-24**
– 0xFFF10028	DMA Buffer Descriptor Base Register 10	DMABDBR10	**page 14-24**
– 0xFFF1002C	DMA Buffer Descriptor Base Register 11	DMABDBR11	**page 14-24**
– 0xFFF10030	DMA Buffer Descriptor Base Register 12	DMABDBR12	**page 14-24**
– 0xFFF10034	DMA Buffer Descriptor Base Register 13	DMABDBR13	**page 14-24**
– 0xFFF10038	DMA Buffer Descriptor Base Register 14	DMABDBR14	**page 14-24**
– 0xFFF1003C	DMA Buffer Descriptor Base Register 15	DMABDBR15	**page 14-24**
– 0xFFF10040–0xFFF100FF	Reserved		
– 0xFFF10100	DMA Channel Configuration Register 0	DMACHCR0	**page 14-25**
– 0xFFF10104	DMA Channel Configuration Register 1	DMACHCR1	**page 14-25**
– 0xFFF10108	DMA Channel Configuration Register 2	DMACHCR2	**page 14-25**
– 0xFFF1010C	DMA Channel Configuration Register 3	DMACHCR3	**page 14-25**
– 0xFFF10110	DMA Channel Configuration Register 4	DMACHCR4	**page 14-25**
– 0xFFF10114	DMA Channel Configuration Register 5	DMACHCR5	**page 14-25**
– 0xFFF10118	DMA Channel Configuration Register 6	DMACHCR6	**page 14-25**
– 0xFFF1011C	DMA Channel Configuration Register 7	DMACHCR7	**page 14-25**
– 0xFFF10120	DMA Channel Configuration Register 8	DMACHCR8	**page 14-25**
– 0xFFF10124	DMA Channel Configuration Register 9	DMACHCR9	**page 14-25**
– 0xFFF10128	DMA Channel Configuration Register 10	DMACHCR10	**page 14-25**
– 0xFFF1012C	DMA Channel Configuration Register 11	DMACHCR11	**page 14-25**
– 0xFFF10130	DMA Channel Configuration Register 12	DMACHCR12	**page 14-25**
– 0xFFF10134	DMA Channel Configuration Register 13	DMACHCR13	**page 14-25**
– 0xFFF10138	DMA Channel Configuration Register 14	DMACHCR14	**page 14-25**
– 0xFFF1013C	DMA Channel Configuration Register 15	DMACHCR15	**page 14-25**
– 0xFFF10140–0xFFF101FF	Reserved		
– 0xFFF10200	DMA Global Configuration Register	DMAGCR	**page 14-27**
– 0xFFF10204	DMA Channel Enable Register	DMACHER	**page 14-28**
– 0xFFF10208–0xFFF1020B	Reserved		
– 0xFFF1020C	DMA Channel Disable Register	DMACHDR	**page 14-28**
– 0xFFF10210–0xFFF10213	Reserved		
– 0xFFF10214	DMA Channel Freeze Register	DMACHFR	**page 14-29**
– 0xFFF10218–0xFFF10223	Reserved		
– 0xFFF10224	DMA Channel Defrost Register	DMACHDFR	**page 14-29**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF10228–0xFFF10233	Reserved		
– 0xFFF10234	DMA Time-To-Deadline Register 0	DMAEDFTDL0	**page 14-30**
– 0xFFF10238	DMA Time-To-Deadline Register 1	DMAEDFTDL1	**page 14-30**
– 0xFFF1023C	DMA Time-To-Deadline Register 2	DMAEDFTDL2	**page 14-30**
– 0xFFF10240	DMA Time-To-Deadline Register 3	DMAEDFTDL3	**page 14-30**
– 0xFFF10244	DMA Time-To-Deadline Register 4	DMAEDFTDL4	**page 14-30**
– 0xFFF10248	DMA Time-To-Deadline Register 5	DMAEDFTDL5	**page 14-30**
– 0xFFF1024C	DMA Time-To-Deadline Register 6	DMAEDFTDL6	**page 14-30**
– 0xFFF10250	DMA Time-To-Deadline Register 7	DMAEDFTDL7	**page 14-30**
– 0xFFF10254	DMA Time-To-Deadline Register 8	DMAEDFTDL8	**page 14-30**
– 0xFFF10258	DMA Time-To-Deadline Register 9	DMAEDFTDL9	**page 14-30**
– 0xFFF1025C	DMA Time-To-Deadline Register 10	DMAEDFTDL10	**page 14-30**
– 0xFFF10260	DMA Time-To-Deadline Register 11	DMAEDFTDL11	**page 14-30**
– 0xFFF10264	DMA Time-To-Deadline Register 12	DMAEDFTDL12	**page 14-30**
– 0xFFF10268	DMA Time-To-Deadline Register 13	DMAEDFTDL13	**page 14-30**
– 0xFFF1026C	DMA Time-To-Deadline Register 14	DMAEDFTDL14	**page 14-30**
– 0xFFF10270	DMA Time-To-Deadline Register 15	DMAEDFTDL15	**page 14-30**
– 0xFFF10274–0xFFF10333	Reserved		
– 0xFFF10334	DMA EDF Control Register	DMAEDFCTRL	**page 14-31**
– 0xFFF10338	DMA EDF Mask Register	DMAEDFMR	**page 14-32**
– 0xFFF1033C–0xFFF1033f	Reserved		
– 0xFFF10340	DMA EDF Mask Update Register	DMAEDFMUR	**page 14-32**
– 0xFFF10344	DMA EDF Status Register	DMAEDFSTR	**page 14-34**
– 0xFFF10348–0xFFF1034B	Reserved		
– 0xFFF1034C	DMA Mask Register	DMAMR	**page 14-34**
– 0xFFF10350–0xFFF1035B	Reserved		
– 0xFFF1035C	DMA Mask Update Register	DMAMUR	**page 14-35**
– 0xFFF10360	DMA Destination Status Register	DMASTR	**page 14-36**
– 0xFFF10364–0xFFF1036F	Reserved		
– 0xFFF10370	DMA Error Register	DMAERR	**page 14-37**
– 0xFFF10374–0xFFF10377	Reserved		
– 0xFFF10378	DMA Local Profiling Configuration Register	DMALPCR	**page 14-39**
– 0xFFF1037C	DMA Round Robin Priority Group Update Register	DMARRPGUR	**page 14-40**
– 0xFFF10380	DMA Channel Active Status Register	DMACHASTR	**page 14-41**
– 0xFFF10384–0xFFF10387	Reserved		
– 0xFFF10388	DMA Channel Freeze Status Register	DMACHFSTR	**page 14-41**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF1038C–0xFFF103FF	reserved		
• 0xFFF10400–0xFFF17FFF	reserved		
• 0xFFF18000–0xFFF18FFF	CLASS 0		
– 0xFFF18000	CLASS 0 MBus Target Configuration Register 0	C0MTCR0	**page 4-15**
– 0xFFF18004–0xFFF1801F	Reserved		
– 0xFFF18020	CLASS 0 MBus Target Configuration Register 1	C0MTCR1	**page 4-15**
– 0xFFF18024–0xFFF1803F	Reserved		
– 0xFFF18040	CLASS 0 MBus Target Configuration Register 2	C0MTCR2	**page 4-15**
– 0xFFF18044–0xFFF1805F	Reserved		
– 0xFFF18060	CLASS 0 MBus Target Configuration Register 3	C0MTCR3	**page 4-15**
– 0xFFF18064–0xFFF1807F	Reserved		
– 0xFFF18080	CLASS 0 MBus Target Configuration Register 4	C0MTCR4	**page 4-15**
– 0xFFF18084–0xFFF187FF	Reserved		
– 0xFFF18800	CLASS 0 Priority Mapping Register 0	C0PMR0	**page 4-17**
– 0xFFF18804	CLASS 0 Priority Mapping Register 1	C0PMR1	**page 4-17**
– 0xFFF18808	CLASS 0 Priority Mapping Register 2	C0PMR2	**page 4-17**
– 0xFFF1880C	CLASS 0 Priority Mapping Register 3	C0PMR3	**page 4-17**
– 0xFFF18810	CLASS 0 Priority Mapping Register 4	C0PMR4	**page 4-17**
– 0xFFF18814	CLASS 0 Priority Mapping Register 5	C0PMR5	**page 4-17**
– 0xFFF18818–0xFFF1883F	reserved		
– 0xFFF18840	CLASS 0 Priority Auto Upgrade Value Register 0	C0PAVR0	**page 4-19**
– 0xFFF18844	CLASS 0 Priority Auto Upgrade Value Register 1	C0PAVR1	**page 4-19**
– 0xFFF18848	CLASS 0 Priority Auto Upgrade Value Register 2	C0PAVR2	**page 4-19**
– 0xFFF1884C	CLASS 0 Priority Auto Upgrade Value Register 3	C0PAVR3	**page 4-19**
– 0xFFF18850	CLASS 0 Priority Auto Upgrade Value Register 4	C0PAVR4	**page 4-19**
– 0xFFF18854	CLASS 0 Priority Auto Upgrade Value Register 5	C0PAVR5	**page 4-19**
– 0xFFF18858–0xFFF1887F	reserved		
– 0xFFF18880	CLASS 0 Priority Auto Upgrade Control Register 0	C0PACR0	**page 4-20**
– 0xFFF18884	CLASS 0 Priority Auto Upgrade Control Register 1	C0PACR1	**page 4-20**
– 0xFFF18888	CLASS 0 Priority Auto Upgrade Control Register 2	C0PACR2	**page 4-20**
– 0xFFF1888C	CLASS 0 Priority Auto Upgrade Control Register 3	C0PACR3	**page 4-20**
– 0xFFF18890	CLASS 0 Priority Auto Upgrade Control Register 4	C0PACR4	**page 4-20**
– 0xFFF18894	CLASS 0 Priority Auto Upgrade Control Register 5	C0PACR5	**page 4-20**
– 0xFFF18898–0xFFF189FF	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF18A00	CLASS 0 Initiator Profiling Configuration Register 0	C0IPCR0	**page 4-24**
– 0xFFF18A04	CLASS 0 Initiator Profiling Configuration Register 1	C0IPCR1	**page 4-24**
– 0xFFF18A08	CLASS 0 Initiator Profiling Configuration Register 2	C0IPCR2	**page 4-24**
– 0xFFF18A0C	CLASS 0 Initiator Profiling Configuration Register 3	C0IPCR3	**page 4-24**
– 0xFFF18A10	CLASS 0 Initiator Profiling Configuration Register 4	C0IPCR4	**page 4-24**
– 0xFFF18A14	CLASS 0 Initiator Profiling Configuration Register 5	C0IPCR5	**page 4-24**
– 0xFFF18A18–0xFFF18A3F	reserved		
– 0xFFF18A40	CLASS 0 Initiator Watch Point Control Register 0	C0IWPCR0	**page 4-25**
– 0xFFF18A44	CLASS 0 Initiator Watch Point Control Register 1	C0IWPCR1	**page 4-25**
– 0xFFF18A48	CLASS 0 Initiator Watch Point Control Register 2	C0IWPCR2	**page 4-25**
– 0xFFF18A4C	CLASS 0 Initiator Watch Point Control Register 3	C0IWPCR3	**page 4-25**
– 0xFFF18A50	CLASS 0 Initiator Watch Point Control Register 4	C0IWPCR4	**page 4-25**
– 0xFFF18A54	CLASS 0 Initiator Watch Point Control Register 5	C0IWPCR5	**page 4-25**
– 0xFFF18A58–0xFFF18A7F	reserved		
– 0xFFF18A80	CLASS 0 Arbitration Weight Register 0	C0AWR0	**page 4-26**
– 0xFFF18A84	CLASS 0 Arbitration Weight Register 1	C0AWR1	**page 4-26**
– 0xFFF18A88	CLASS 0 Arbitration Weight Register 2	C0AWR2	**page 4-26**
– 0xFFF18A8C	CLASS 0 Arbitration Weight Register 3	C0AWR3	**page 4-26**
– 0xFFF18A90	CLASS 0 Arbitration Weight Register 4	C0AWR4	**page 4-26**
– 0xFFF18A94	CLASS 0 Arbitration Weight Register 5	C0AWR5	**page 4-26**
– 0xFFF18A98–0xFFF18D7F	reserved		
– 0xFFF18D80	CLASS 0 IRQ Status Register	C0ISR	**page 4-27**
– 0xFFF18D84–0xFFF18DBF	Reserved		
– 0xFFF18DC0	CLASS 0 IRQ Enable Register	C0IER	**page 4-28**
– 0xFFF18DC4–0xFFF18DFF	Reserved		
– 0xFFF18E00	CLASS 0 Target Profiling Configuration Register	C0TPCR	**page 4-29**
– 0xFFF18E04	CLASS 0 Profiling Control Register	C0PCR	**page 4-31**
– 0xFFF18E08	CLASS 0 Watch Point Control Register	C0WPCR	**page 4-32**
– 0xFFF18E0C	CLASS 0 Watch Point Access Configuration Register	C0WPACR	**page 4-34**
– 0xFFF18E10	CLASS 0 Watch Point Extended Access Configuration Register	C0WPEACR	**page 4-36**
– 0xFFF18E14	CLASS 0 Watch Point Address Mask Register	C0WPAMR	**page 4-38**
– 0xFFF18E18	CLASS 0 Profiling Time Out Register	C0PTOR	**page 4-39**
– 0xFFF18E1C	CLASS 0 Target Watch Point Control Register	C0TWPCR	**page 4-40**
– 0xFFF18E20	CLASS 0 Profiling IRQ Status Register	C0PISR	**page 4-41**
– 0xFFF18E24	CLASS 0 Profiling IRQ Enable Register	C0PIER	**page 4-42**
– 0xFFF18E28-0xFFF18E3F	Reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF18E40	CLASS 0 Profiling Reference Counter Register	C0PRCR	**page 4-42**
– 0xFFF18E44	CLASS 0 Profiling General Counter Register 0	C0PGCR0	**page 4-43**
– 0xFFF18E48	CLASS 0 Profiling General Counter Register 1	C0PGCR1	**page 4-43**
– 0xFFF18E4C	CLASS 0 Profiling General Counter Register 2	C0PGCR2	**page 4-43**
– 0xFFF18E50	CLASS 0 Profiling General Counter Register 3	C0PGCR3	**page 4-43**
– 0xFFF18E54–0xFFF18FBF	Reserved		
– 0xFFF18FC0	CLASS 0 Arbitration Control Register	C0ACR	**page 4-45**
– 0xFFF18FC4–0xFFF18FFF	Reserved		
• 0xFFF19000–0xFFF19FFF	CLASS 1		
– 0xFFF19000	CLASS 1 MBus Target Configuration Register 0	C1MTCR0	**page 4-15**
– 0xFFF19004–0xFFF1901F	Reserved		
– 0xFFF19020	CLASS 1 MBus Target Configuration Register 1	C1MTCR1	**page 4-15**
– 0xFFF19024–0xFFF1903F	Reserved		
– 0xFFF19040	CLASS 1 MBus Target Configuration Register 2	C1MTCR2	**page 4-15**
– 0xFFF19044–0xFFF1905F	Reserved		
– 0xFFF19060	CLASS 1 MBus Target Configuration Register 3	C1MTCR3	**page 4-15**
– 0xFFF19064–0xFFF197FF	Reserved		
– 0xFFF19800	CLASS 1 Priority Mapping Register 0	C1PMR0	**page 4-17**
– 0xFFF19804	CLASS 1 Priority Mapping Register 1	C1PMR1	**page 4-17**
– 0xFFF19808	CLASS 1 Priority Mapping Register 2	C1PMR2	**page 4-17**
– 0xFFF1980C	CLASS 1 Priority Mapping Register 3	C1PMR3	**page 4-17**
– 0xFFF19810	CLASS 1 Priority Mapping Register 4	C1PMR4	**page 4-17**
– 0xFFF19814	CLASS 1 Priority Mapping Register 5	C1PMR5	**page 4-17**
– 0xFFF19818–0xFFF1983F	reserved		
– 0xFFF19840	CLASS 1 Priority Auto Upgrade Value Register 0	C1PAVR0	**page 4-19**
– 0xFFF19844	CLASS 1 Priority Auto Upgrade Value Register 1	C1PAVR1	**page 4-19**
– 0xFFF19848	CLASS 1 Priority Auto Upgrade Value Register 2	C1PAVR2	**page 4-19**
– 0xFFF1984C	CLASS 1 Priority Auto Upgrade Value Register 3	C1PAVR3	**page 4-19**
– 0xFFF19850	CLASS 1 Priority Auto Upgrade Value Register 4	C1PAVR4	**page 4-19**
– 0xFFF19854	CLASS 1 Priority Auto Upgrade Value Register 5	C1PAVR5	**page 4-19**
– 0xFFF19858–0xFFF1987F	reserved		
– 0xFFF19880	CLASS 1 Priority Auto Upgrade Control Register 0	C1PACR0	**page 4-20**
– 0xFFF19884	CLASS 1 Priority Auto Upgrade Control Register 1	C1PACR1	**page 4-20**
– 0xFFF19888	CLASS 1 Priority Auto Upgrade Control Register 2	C1PACR2	**page 4-20**
– 0xFFF1988C	CLASS 1 Priority Auto Upgrade Control Register 3	C1PACR3	**page 4-20**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF19890	CLASS 1 Priority Auto Upgrade Control Register 4	C1PACR4	**page 4-20**
– 0xFFF19894	CLASS 1 Priority Auto Upgrade Control Register 5	C1PACR5	**page 4-20**
– 0xFFF19898–0xFFF1997F	reserved		
– 0xFFF19980	CLASS 1 Error Address Register 0	C1EAR0	**page 4-21**
– 0xFFF19984	CLASS 1 Error Address Register 1	C1EAR1	**page 4-21**
– 0xFFF19988	CLASS 1 Error Address Register 2	C1EAR2	**page 4-21**
– 0xFFF1998C	CLASS 1 Error Address Register 3	C1EAR3	**page 4-21**
– 0xFFF19990	CLASS 1 Error Address Register 4	C1EAR4	**page 4-21**
– 0xFFF19994–0xFFF199BF	reserved		
– 0xFFF199C0	CLASS 1 Error Extended Address Register 0	C1EEAR0	**page 4-22**
– 0xFFF199C4	CLASS 1 Error Extended Address Register 1	C1EEAR1	**page 4-22**
– 0xFFF199C8	CLASS 1 Error Extended Address Register 2	C1EEAR2	**page 4-22**
– 0xFFF199CC	CLASS 1 Error Extended Address Register 3	C1EEAR3	**page 4-22**
– 0xFFF199D0	CLASS 1 Error Extended Address Register 4	C1EEAR4	**page 4-22**
– 0xFFF199D4–0xFFF199FF	reserved		
– 0xFFF19A00	CLASS 1 Initiator Profiling Configuration Register 0	C1IPCR0	**page 4-24**
– 0xFFF19A04	CLASS 1 Initiator Profiling Configuration Register 1	C1IPCR1	**page 4-24**
– 0xFFF19A08	CLASS 1 Initiator Profiling Configuration Register 2	C1IPCR2	**page 4-24**
– 0xFFF19A0C	CLASS 1 Initiator Profiling Configuration Register 3	C1IPCR3	**page 4-24**
– 0xFFF19A10	CLASS 1 Initiator Profiling Configuration Register 4	C1IPCR4	**page 4-24**
– 0xFFF19A14	CLASS 1 Initiator Profiling Configuration Register 5	C10IPCR5	**page 4-24**
– 0xFFF19A18–0xFFF19A3F	reserved		
– 0xFFF19A40	CLASS 1 Initiator Watch Point Control Register 0	C1IWPCR0	**page 4-25**
– 0xFFF19A44	CLASS 1 Initiator Watch Point Control Register 1	C1IWPCR1	**page 4-25**
– 0xFFF19A48	CLASS 1 Initiator Watch Point Control Register 2	C1IWPCR2	**page 4-25**
– 0xFFF19A4C	CLASS 1 Initiator Watch Point Control Register 3	C1IWPCR3	**page 4-25**
– 0xFFF19A50	CLASS 1 Initiator Watch Point Control Register 4	C1IWPCR4	**page 4-25**
– 0xFFF19A54	CLASS 1 Initiator Watch Point Control Register 5	C1IWPCR5	**page 4-25**
– 0xFFF19A58–0xFFF19A7F	reserved		
– 0xFFF19A80	CLASS 1 Arbitration Weight Register 0	C1AWR0	**page 4-26**
– 0xFFF19A84	CLASS 1 Arbitration Weight Register 1	C10AWR1	**page 4-26**
– 0xFFF19A88	CLASS 1 Arbitration Weight Register 2	C1AWR2	**page 4-26**
– 0xFFF19A8C	CLASS 1 Arbitration Weight Register 3	C1AWR3	**page 4-26**
– 0xFFF19A90	CLASS 1 Arbitration Weight Register 4	C1AWR4	**page 4-26**
– 0xFFF19A94	CLASS 1 Arbitration Weight Register 5	C1AWR5	**page 4-26**
– 0xFFF19A98–0xFFF19C03	reserved		
– 0xFFF19C04	CLASS 1 Start Address Decoder 1	C1SAD1	**page 4-46**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF19C08	CLASS 1 Start Address Decoder 2	C1SAD2	**page 4-46**
– 0xFFF19C0C–0xFFF19C43	reserved		
– 0xFFF19C44	CLASS 1 End Address Decoder 1	C1EAD1	**page 4-47**
– 0xFFF19C48	CLASS 1 End Address Decoder 2	C1EAD2	**page 4-47**
– 0xFFF19C4C–0xFFF19C83	reserved		
– 0xFFF19C84	CLASS 1 Attributes Decoder 1	C1ATD1	**page 4-48**
– 0xFFF19C88	CLASS 1 Attributes Decoder 2	C1ATD2	**page 4-48**
– 0xFFF19C8C–0xFFF19D7F	reserved		
– 0xFFF19D80	CLASS 1 IRQ Status Register	C1ISR	**page 4-27**
– 0xFFF19D84–0xFFF19DBF	Reserved		
– 0xFFF19DC0	CLASS 1 IRQ Enable Register	C1IER	**page 4-28**
– 0xFFF19DC4–0xFFF19DFF	Reserved		
– 0xFFF19E00	CLASS 1 Target Profiling Configuration Register	C1TPCR	**page 4-29**
– 0xFFF19E04	CLASS 1 Profiling Control Register	C1PCR	**page 4-31**
– 0xFFF19E08	CLASS 1 Watch Point Control Register	C1WPCR	**page 4-32**
– 0xFFF19E0C	CLASS 1 Watch Point Access Configuration Register	C1WPACR	**page 4-34**
– 0xFFF19E10	CLASS 1 Watch Point Extended Access Configuration Register	C1WPEACR	**page 4-36**
– 0xFFF19E14	CLASS 1 Watch Point Address Mask Register	C1WPAMR	**page 4-38**
– 0xFFF19E18	CLASS 1 Profiling Time Out Register	C1PTOR	**page 4-39**
– 0xFFF19E1C	CLASS 1 Target Watch Point Control Register	C1TWPCR	**page 4-40**
– 0xFFF19E20	CLASS 1 Profiling IRQ Status Register	C1PISR	**page 4-41**
– 0xFFF19E24	CLASS 1 Profiling IRQ Enable Register	C1PIER	**page 4-42**
– 0xFFF19E28-0xFFF19E3F	Reserved		
– 0xFFF19E40	CLASS 1 Profiling Reference Counter Register	C1PRCR	**page 4-42**
– 0xFFF19E44	CLASS 1 Profiling General Counter Register 0	C1PGCR0	**page 4-43**
– 0xFFF19E48	CLASS 1 Profiling General Counter Register 1	C1PGCR1	**page 4-43**
– 0xFFF19E4C	CLASS 1 Profiling General Counter Register 2	C1PGCR2	**page 4-43**
– 0xFFF19E50	CLASS 1 Profiling General Counter Register 3	C1PGCR3	**page 4-43**
– 0xFFF19E54-0xFFF19FBF	Reserved		
– 0xFFF19FC0	CLASS 1 Arbitration Control Register	C1ACR	**page 4-45**
– 0xFFF19FC4–0xFFF19FFF	Reserved		
• 0xFFF1A000–0xFFF1AFFF	CLASS 2		
– 0xFFF1A000	CLASS 2 MBus Target Configuration Register 0	C2MTCR0	**page 4-15**
– 0xFFF1A004–0xFFF1A7FF	Reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF1A800	CLASS 2 Priority Mapping Register 0	C2PMR0	**page 4-17**
– 0xFFF1A804	CLASS 2 Priority Mapping Register 1	C2PMR1	**page 4-17**
– 0xFFF1A808	CLASS 2 Priority Mapping Register 2	C2PMR2	**page 4-17**
– 0xFFF1A80C	CLASS 2 Priority Mapping Register 3	C2PMR3	**page 4-17**
– 0xFFF1A810–0xFFF1A83F	reserved		
– 0xFFF1A840	CLASS 2 Priority Auto Upgrade Value Register 0	C2PAVR0	**page 4-19**
– 0xFFF1A844	CLASS 2 Priority Auto Upgrade Value Register 1	C2PAVR1	**page 4-19**
– 0xFFF1A848	CLASS 2 Priority Auto Upgrade Value Register 2	C2PAVR2	**page 4-19**
– 0xFFF1A84C	CLASS 2 Priority Auto Upgrade Value Register 3	C2PAVR3	**page 4-19**
– 0xFFF1A850–0xFFF1A87F	reserved		
– 0xFFF1A880	CLASS 2 Priority Auto Upgrade Control Register 0	C2PACR0	**page 4-20**
– 0xFFF1A884	CLASS 2 Priority Auto Upgrade Control Register 1	C2PACR1	**page 4-20**
– 0xFFF1A888	CLASS 2 Priority Auto Upgrade Control Register 2	C2PACR2	**page 4-20**
– 0xFFF1A88C	CLASS 2 Priority Auto Upgrade Control Register 3	C2PACR3	**page 4-20**
– 0xFFF1A890–0xFFF1A9FF	reserved		
– 0xFFF1AA00	CLASS 2 Initiator Profiling Configuration Register 0	C2IPCR0	**page 4-24**
– 0xFFF1AA04–0xFFF1AA07	reserved		
– 0xFFF1AA08	CLASS 2 Initiator Profiling Configuration Register 2	C2IPCR2	**page 4-25**
– 0xFFF1AA0C	CLASS 2 Initiator Profiling Configuration Register 3	C2IPCR3	**page 4-25**
– 0xFFF1AA10–0xFFF1AA3F	reserved		
– 0xFFF1AA40	CLASS 2 Initiator Watch Point Control Register 0	C2IWPCR0	**page 4-25**
– 0xFFF1AA44	CLASS 2 Initiator Watch Point Control Register 1	C2IWPCR1	**page 4-25**
– 0xFFF1AA48	CLASS 2 Initiator Watch Point Control Register 2	C2IWPCR2	**page 4-25**
– 0xFFF1AA4C	CLASS 2 Initiator Watch Point Control Register 3	C2IWPCR3	**page 4-25**
– 0xFFF1AA50–0xFFF1AA7F	reserved		
– 0xFFF1AA80	CLASS 2 Arbitration Weight Register 0	C2AWR0	**page 4-26**
– 0xFFF1AA84	CLASS 2 Arbitration Weight Register 1	C2AWR1	**page 4-26**
– 0xFFF1AA88	CLASS 2 Arbitration Weight Register 2	C2AWR2	**page 4-26**
– 0xFFF1AA8C	CLASS 2 Arbitration Weight Register 3	C2AWR3	**page 4-26**
– 0xFFF1AA90–0xFFF1AD7F	reserved		
– 0xFFF1AD80	CLASS 2 IRQ Status Register	C2ISR	**page 4-27**
– 0xFFF1AD84–0xFFF1ADBF	Reserved		
– 0xFFF1ADC0	CLASS 2 IRQ Enable Register	C2IER	**page 4-28**
– 0xFFF1ADC4–0xFFF1ADFF	Reserved		
– 0xFFF1AE00	CLASS 2 Target Profiling Configuration Register	C2TPCR	**page 4-29**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF1AE04	CLASS 2 Profiling Control Register	C2PCR	**page 4-31**
– 0xFFF1AE08	CLASS 2 Watch Point Control Register	C2WPCR	**page 4-32**
– 0xFFF1AE0C	CLASS 2 Watch Point Access Configuration Register	C2WPACR	**page 4-34**
– 0xFFF1AE10	CLASS 2 Watch Point Extended Access Configuration Register	C2WPEACR	**page 4-36**
– 0xFFF1AE14	CLASS 2 Watch Point Address Mask Register	C2WPAMR	**page 4-38**
– 0xFFF1AE18	CLASS 2 Profiling Time Out Register	C2PTOR	**page 4-39**
– 0xFFF1AE1C	CLASS 2 Target Watch Point Control Register	C2TWPCR	**page 4-40**
– 0xFFF1AE20	CLASS 2 Profiling IRQ Status Register	C2PISR	**page 4-41**
– 0xFFF1AE24	CLASS 2 Profiling IRQ Enable Register	C2PIER	**page 4-42**
– 0xFFF1AE28-0xFFF1AE3F	Reserved		
– 0xFFF1AE40	CLASS 2 Profiling Reference Counter Register	C2PRCR	**page 4-42**
– 0xFFF1AE44	CLASS 2 Profiling General Counter Register 0	C2PGCR0	**page 4-43**
– 0xFFF1AE48	CLASS 2 Profiling General Counter Register 1	C2PGCR1	**page 4-43**
– 0xFFF1AE4C	CLASS 2 Profiling General Counter Register 2	C2PGCR2	**page 4-43**
– 0xFFF1AE50	CLASS 2 Profiling General Counter Register 3	C2PGCR3	**page 4-43**
– 0xFFF1AE54-0xFFF1AF7F	Reserved		
– 0xFFF1AF80	CLASS 2 General Purpose Register	C2GPR	**page 4-44**
– 0xFFF1AF84–0xFFF1AFBF	Reserved		
– 0xFFF1AFC0	CLASS 2 Arbitration Control Register	C2ACR	**page 4-45**
– 0xFFF1AFC4–0xFFF1AFFF	Reserved		
• 0xFFF1B000–0xFFF1FFFF	reserved		
• 0xFFF20000–0xFFF21FFF	DDR Controller		
– 0xFFF20000	Chip Select 0 Memory Bounds	CS0_BNDS	**page 12-32**
– 0xFFF20004–0xFFF20007	reserved		
– 0xFFF20008	Chip Select 1 Memory Bounds	CS1_BNDS	**page 12-32**
– 0xFFF2000C–0xFFF2007F	reserved		
– 0xFFF20080	Chip Select 0 Configuration	CS0_CONFIG	**page 12-33**
– 0xFFF20084	Chip Select 1 Configuration	CS1_CONFIG	**page 12-33**
– 0xFFF20088–0xFFF200FF	reserved		
– 0xFFF20100	DDR SDRAM Extended Refresh Recovery	EXT_REFREC	**page 12-34**
– 0xFFF20104	DDR SDRAM Timing Configuration 0	TIMING_CFG_0	**page 12-35**
– 0xFFF20108	DDR SDRAM Timing Configuration 1	TIMING_CFG_1	**page 12-38**
– 0xFFF2010C	DDR SCRAM Timing Configuration 2	TIMING_CFG_2	**page 12-40**
– 0xFFF20110	DDR SDRAM Control Configuration	DDR_SDRAM_CFG	**page 12-42**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF20114	DDR SDRAM Control Configuration 2	DDR_SDRAM_CFG_2	**page 12-44**
– 0xFFF20118	DDR SDRAM Mode Configuration	DDR_SDRAM_MODE	**page 12-46**
– 0xFFF2011C	DDR SDRAM Mode Configuration 2	DDR_SDRAM_MODE_2	**page 12-47**
– 0xFFF20120	DDR SDRAM Model Control	DDR_SCRAM_MD_CNTL	**page 12-47**
– 0xFFF20124	DDR SDRAM Interval Configuration	DDR_SDRAM_INTERVAL	**page 12-49**
– 0xFFF20128	DDR SDRAM Data Initialization	DDR_DATA_INIT	**page 12-50**
– 0xFFF2012C–0xFFF2012F	reserved		
– 0xFFF20130	DDR SDRAM Clock Control	DDR_SDRAM_CLK_CNT	**page 12-50**
– 0xFFF20134–0xFFF20147	reserved		
– 0xFFF20148	DDR Training Initialization Address	DDR_INIT_ADDRESS	**page 12-51**
– 0xFFF2014C	DDR Training Initialization Enable	DDR_INIT_EN	**page 12-52**
– 0xFFF20150–0xFFF20BF7	reserved		
– 0xFFF20BF8	DDR IP Block Revision 1	DDR_IP_REV1	**page 12-53**
– 0xFFF20BFC	DDR IP Block Revision 2	DDR_IP_REV2	**page 12-53**
– 0xFFF20C00–0xFFF20DFF	reserved		
– 0xFFF20E00	Memory Data Path Error Injection Mask High	DDR_ERR_INJECT_HI	**page 12-54**
– 0xFFF20E04	Memory Data Path Error Injection Mask Low	DDR_ERR_INJECT_LO	**page 12-55**
– 0xFFF20E08	Memory Data Path Error Injection Mask ECC	DDR_ERR_INJECT	**page 12-55**
– 0xFFF20E0C–0xFFF20E1F	reserved		
– 0xFFF20E20	Memory Data Path Read Capture High	CAPTURE_DATA_HI	**page 12-56**
– 0xFFF20E24	Memory Data Path Read Capture Low	CAPTURE-DATA_LO	**page 12-57**
– 0xFFF20E28	Memory Data Path Read Capture ECC	CAPTURE_ECC	**page 12-57**
– 0xFFF20E2C–0xFFF20E3F	reserved		
– 0xFFF20E40	Memory Error Detect	ERR_DETECT	**page 12-58**
– 0xFFF20E44	Memory Error Disable	ERR_DISABLE	**page 12-59**
– 0xFFF20E48	Memory Error Interrupt Enable	ERR_INT_EN	**page 12-60**
– 0xFFF20E4C	Memory Error Attributes Capture	CAPTURE_ATTRIBUTES	**page 12-61**
– 0xFFF20E50	Memory Error Address Capture	CAPTURE_ADDRESS	**page 12-62**
– 0xFFF20E54–0xFFF20E57	reserved		
– 0xFFF20E58	Single-Bit ECC Memory Error Management	ERR_SBE	**page 12-62**
– 0xFFF20E5C–0xFFF20FFF	reserved		
– 0xFFF21000	DDR SDRAM DDR Status	DDR_STOP_STATUS	**page 12-63**
– 0xFFF21004	DDR SDRAM DDR Power Control Register	DDR_PWR	**page 12-64**
– 0xFFF21008	DDR SDRAM MDIC Output Enable Control Register	MDIC_OE_CONT	**page 12-65**
– 0xFFF2100C	DDR SDRAM Termination, OCD, and ODT Control Register	TERM_OCD_ODT_CONT	**page 12-66**
– 0xFFF21010	DDR SDRAM Clock Ratio Control Register	CLK_RATIO_CONT	**page 12-67**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF21014–0xFFF21FFF	reserved		
• 0xFFF22000–0xFFF23FFF	reserved		
• 0xFFF24000–0xFFF2407F	Clocks		
– 0xFFF24000	System Clock Control Register	SCCR	**page 7-12**
– 0xFFF24004–0xFFF2401F	reserved		
– 0xFFF24020	PLL Clock Mode Register0	PCMR0B	**page 7-14**
– 0xFFF24024	PLL Clock Mode Register1	PCMR1B	**page 7-16**
– 0xFFF24028	PLL Clock Mode Register2	PCMR2B	**page 7-18**
– 0xFFF2402C–0xFFF2402F	reserved		
– 0xFFF24030	PLL Clock Mode Front Register0	PCMR0F	**page 7-14**
– 0xFFF24034	PLL Clock Mode Front Register1	PCMR1F	**page 7-16**
– 0xFFF24038	PLL Clock Mode Front Register2	PCMR2F	**page 7-18**
– 0xFFF2403C–0xFFF2403F	reserved		
– 0xFFF24040	Dividers Clock Mode Register0	DCMR0B	**page 7-20**
– 0xFFF24044	Dividers Clock Mode Register1	DCMR1B	**page 7-22**
– 0xFFF24048–0xFFF2404F	reserved		
– 0xFFF24050	Dividers Clock Mode Front Register0	DCMR0F	**page 7-20**
– 0xFFF24054	Dividers Clock Mode Front Register1	DCMR1F	**page 7-22**
– 0xFFF24058–0xFFF2405F	reserved		
– 0xFFF24060	PLL Auxiliary Mode Register0	PAMR0B	**page 7-23**
– 0xFFF24064	PLL Auxiliary Mode Register1	PAMR1B	**page 7-24**
– 0xFFF24068	PLL Auxiliary Mode Register2	PAMR2B	**page 7-24**
– 0xFFF2406C–0xFFF2406F	reserved		
– 0xFFF24070	PLL Auxiliary Mode Front Register0	PAMR0F	**page 7-23**
– 0xFFF24074	PLL Auxiliary Mode Front Register1	PAMR1F	**page 7-24**
– 0xFFF24078	PLL Auxiliary Mode Front Register2	PAMR2F	**page 7-24**
• 0xFFF2407C–0xFFF247FF	reserved		
• 0xFFF24800–0xFFF248FF	Reset		
– 0xFFF24800	Reset Configuration Word Low Register	RCWLR	**page 5-17**
– 0xFFF24804	Reset Configuration Word High Register	RCWHR	**page 5-19**
– 0xFFF24808–0xFFF2480F	reserved		
– 0xFFF24810	Reset Status Register	RSR	**page 5-21**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF24814–0xFFF24817	reserved		
– 0xFFF24818	Reset Protection Register	RPR	**page 5-23**
– 0xFFF2481C	Reset Control Register	RCR	**page 5-24**
– 0xFFF24820	Reset Control Enable Register	RCER	**page 5-25**
– 0xFFF24824–0xFFF248FF	reserved		
• 0xFFF24900–0xFFF24BFF	reserved		
• 0xFFF24C00–0xFFF24CFF	I^2C		
– 0xFFF24C00	I^2C Address Register	I2CADR	**page 24-15**
– 0xFFF24C04	I^2C Frequency Divider Register	I2CFDR	**page 24-16**
– 0xFFF24C08	I^2C Control Register	I2CCR	**page 24-17**
– 0xFFF24C0C	I^2C Status Register	I2CSR	**page 24-18**
– 0xFFF24C10	I^2C Data Register	I2CDR	**page 24-20**
– 0xFFF24C14	I^2C Digital Filter Sampling Rate Register	I2CDFSRR	**page 24-20**
– 0xFFF24C18–0xFFF24CFF	reserved		
• 0xFFF24D00–0xFFF24FFF	reserved		
• 0xFFF25000–0xFFF250FF	Watchdog Timer 0		
– 0xFFF25000–0xFFF25003	reserved		
– 0xFFF25004	System Watchdog 0 Control Register	SW0CRR	**page 21-25**
– 0xFFF25008	System Watchdog 0 Count Register	SW0CNR	**page 21-26**
– 0xFFF2500C–0xFFF2500D	reserved		
– 0xFFF2500E	System Watchdog 0 Service Register	SW0SRR	**page 21-27**
– 0xFFF25010–0xFFF250FF	reserved		
• 0xFFF25100–0xFFF251FF	Watchdog Timer 1		
– 0xFFF25100–0xFFF25103	reserved		
– 0xFFF25104	System Watchdog 1 Control Register	SW1CRR	**page 21-25**
– 0xFFF25108	System Watchdog 1 Count Register	SW1CNR	**page 21-26**
– 0xFFF2510C–0xFFF2510D	reserved		
– 0xFFF2510E	System Watchdog 1 Service Register	SW1SRR	**page 21-27**
– 0xFFF25110–0xFFF251FF	reserved		
• 0xFFF25200–0xFFF252FF	Watchdog Timer 2		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF25200–0xFFF25203	reserved		
– 0xFFF25204	System Watchdog 2 Control Register	SW2CRR	**page 21-25**
– 0xFFF25208	System Watchdog 2 Count Register	SW2CNR	**page 21-26**
– 0xFFF2520C–0xFFF2520D	reserved		
– 0xFFF2520E	System Watchdog 2 Service Register	SW2SRR	**page 21-27**
– 0xFFF25210–0xFFF252FF	reserved		
• 0xFFF25300–0xFFF253FF	Watchdog Timer 3		
– 0xFFF25300–0xFFF25303	reserved		
– 0xFFF25304	System Watchdog 3 Control Register	SW3CRR	**page 21-25**
– 0xFFF25308	System Watchdog 3 Count Register	SW3CNR	**page 21-26**
– 0xFFF2530C–0xFFF2530D	reserved		
– 0xFFF2530E	System Watchdog 3 Service Register	SW3SRR	**page 21-27**
– 0xFFF25310–0xFFF253FF	reserved		
• 0xFFF25400–0xFFF254FF	Watchdog Timer 4		
– 0xFFF25400–0xFFF25403	reserved		
– 0xFFF25404	System Watchdog 4 Control Register	SW4CRR	**page 21-25**
– 0xFFF25408	System Watchdog 4 Count Register	SW4CNR	**page 21-26**
– 0xFFF2540C–0xFFF2540D	reserved		
– 0xFFF2540E	System Watchdog 4 Service Register	SW4SRR	**page 21-27**
– 0xFFF25410–0xFFF254FF	reserved		
• 0xFFF25500–0xFFF25FFF	reserved		
• 0xFFF26000–0xFFF260FF	Timer 0		
– 0xFFF26000	Timer 0 Channel 0 Compare 1 Register	TMR0CMP10	**page 21-21**
– 0xFFF26004	Timer 0 Channel 0 Compare 2 Register	TMR0CMP20	**page 21-21**
– 0xFFF26008	Timer 0 Channel 0 Capture Register	TMR0CAP0	**page 21-23**
– 0xFFF2600C	Timer 0 Channel 0 Load Register	TMR0LOAD0	**page 21-24**
– 0xFFF26010	Timer 0 Channel 0 Hold Register	TMR0HOLD0	**page 21-24**
– 0xFFF26014	Timer 0 Channel 0 Counter Register	TMR0CNTR0	**page 21-24**
– 0xFFF26018	Timer 0 Channel 0 Control Register	TMR0CTL0	**page 21-17**
– 0xFFF2601C	Timer 0 Channel 0 Status and Control Register	TMR0SCTL0	**page 21-19**
– 0xFFF26020	Timer 0 Channel 0 Compare Load 1 Register	TMR0CMPLD10	**page 21-22**
– 0xFFF26024	Timer 0 Channel 0 Compare Load 2 Register	TMR0CMPLD20	**page 21-22**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF26028	Timer 0 Channel 0 Comparator Status and Control Register	TMR0COMSC0	**page 21-22**
– 0xFFF2602C–0xFFF2603F	reserved		
– 0xFFF26040	Timer 0 Channel 1 Compare 1 Register	TMR0CMP11	**page 21-21**
– 0xFFF26044	Timer 0 Channel 1 Compare 2 Register	TMR0CMP21	**page 21-21**
– 0xFFF26048	Timer 0 Channel 1 Capture Register	TMR0CAP1	**page 21-23**
– 0xFFF2604C	Timer 0 Channel 1 Load Register	TMR0LOAD1	**page 21-24**
– 0xFFF26050	Timer 0 Channel 1 Hold Register	TMR0HOLD1	**page 21-24**
– 0xFFF26054	Timer 0 Channel 1 Counter Register	TMR0CNTR1	**page 21-24**
– 0xFFF26058	Timer 0 Channel 1 Control Register	TMR0CTL1	**page 21-17**
– 0xFFF2605C	Timer 0 Channel 1 Status and Control Register	TMR0SCTL1	**page 21-19**
– 0xFFF26060	Timer 0 Channel 1 Compare Load 1 Register	TMR0CMPLD11	**page 21-22**
– 0xFFF26064	Timer 0 Channel 1 Compare Load 2 Register	TMR0CMPLD21	**page 21-22**
– 0xFFF26068	Timer 0 Channel 1 Comparator Status and Control Register	TMR0COMSC1	**page 21-22**
– 0xFFF2606C–0xFFF2607F	reserved		
– 0xFFF26080	Timer 0 Channel 2 Compare 1 Register	TMR0CMP12	**page 21-21**
– 0xFFF26084	Timer 0 Channel 2 Compare 2 Register	TMR0CMP22	**page 21-21**
– 0xFFF26088	Timer 0 Channel 2 Capture Register	TMR0CAP2	**page 21-23**
– 0xFFF2608C	Timer 0 Channel 2 Load Register	TMR0LOAD2	**page 21-24**
– 0xFFF26090	Timer 0 Channel 2 Hold Register	TMR0HOLD2	**page 21-24**
– 0xFFF26094	Timer 0 Channel 2 Counter Register	TMR0CNTR2	**page 21-24**
– 0xFFF26098	Timer 0 Channel 2 Control Register	TMR0CTL2	**page 21-17**
– 0xFFF2609C	Timer 0 Channel 2 Status and Control Register	TMR0SCTL2	**page 21-19**
– 0xFFF260A0	Timer 0 Channel 2 Compare Load 1 Register	TMR0CMPLD12	**page 21-22**
– 0xFFF260A4	Timer 0 Channel 2 Compare Load 2 Register	TMR0CMPLD22	**page 21-22**
– 0xFFF260A8	Timer 0 Channel 2 Comparator Status and Control Register	TMR0COMSC2	**page 21-22**
– 0xFFF260AC–0xFFF260BF	reserved		
– 0xFFF260C0	Timer 0 Channel 3 Compare 1 Register	TMR0CMP13	**page 21-21**
– 0xFFF260C4	Timer 0 Channel 3 Compare 2 Register	TMR0CMP23	**page 21-21**
– 0xFFF260C8	Timer 0 Channel 3 Capture Register	TMR0CAP3	**page 21-23**
– 0xFFF260CC	Timer 0 Channel 3 Load Register	TMR0LOAD3	**page 21-24**
– 0xFFF260D0	Timer 0 Channel 3 Hold Register	TMR0HOLD3	**page 21-24**
– 0xFFF260D4	Timer 0 Channel 3 Counter Register	TMR0CNTR3	**page 21-24**
– 0xFFF260D8	Timer 0 Channel 3 Control Register	TMR0CTL3	**page 21-17**
– 0xFFF260DC	Timer 0 Channel 3 Status and Control Register	TMR0SCTL3	**page 21-19**
– 0xFFF260E0	Timer 0 Channel 3 Compare Load 1 Register	TMR0CMPLD13	**page 21-22**
– 0xFFF260E4	Timer 0 Channel 3 Compare Load 2 Register	TMR0CMPLD23	**page 21-22**
– 0xFFF260E8	Timer 0 Channel 3 Comparator Status and Control Register	TMR0COMSC3	**page 21-22**
– 0xFFF260EC–0xFFF260FF	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
• 0xFFF26100–0xFFF261FF	Timer 1		
– 0xFFF26100	Timer 1 Channel 0 Compare 1 Register	TMR1CMP10	**page 21-21**
– 0xFFF26104	Timer 1 Channel 0 Compare 2 Register	TMR1CMP20	**page 21-21**
– 0xFFF26108	Timer 1 Channel 0 Capture Register	TMR1CAP0	**page 21-23**
– 0xFFF2610C	Timer 1 Channel 0 Load Register	TMR1LOAD0	**page 21-24**
– 0xFFF26110	Timer 1 Channel 0 Hold Register	TMR1HOLD0	**page 21-24**
– 0xFFF26114	Timer 1 Channel 0 Counter Register	TMR1CNTR0	**page 21-24**
– 0xFFF26118	Timer 1 Channel 0 Control Register	TMR1CTL0	**page 21-17**
– 0xFFF2611C	Timer 1 Channel 0 Status and Control Register	TMR1SCTL0	**page 21-19**
– 0xFFF26120	Timer 1 Channel 0 Compare Load 1 Register	TMR1CMPLD10	**page 21-22**
– 0xFFF26124	Timer 1 Channel 0 Compare Load 2 Register	TMR1CMPLD20	**page 21-22**
– 0xFFF26128	Timer 1 Channel 0 Comparator Status and Control Register	TMR1COMSC0	**page 21-22**
– 0xFFF2612C–0xFFF2613F	reserved		
– 0xFFF26140	Timer 1 Channel 1 Compare 1 Register	TMR1CMP11	**page 21-21**
– 0xFFF26144	Timer 1 Channel 1 Compare 2 Register	TMR1CMP21	**page 21-21**
– 0xFFF26148	Timer 1 Channel 1 Capture Register	TMR1CAP1	**page 21-23**
– 0xFFF2614C	Timer 1 Channel 1 Load Register	TMR1LOAD1	**page 21-24**
– 0xFFF26150	Timer 1 Channel 1 Hold Register	TMR1HOLD1	**page 21-24**
– 0xFFF26154	Timer 1 Channel 1 Counter Register	TMR1CNTR1	**page 21-24**
– 0xFFF26158	Timer 1 Channel 1 Control Register	TMR1CTL1	**page 21-17**
– 0xFFF2615C	Timer 1 Channel 1 Status and Control Register	TMR1SCTL1	**page 21-19**
– 0xFFF26160	Timer 1 Channel 1 Compare Load 1 Register	TMR1CMPLD11	**page 21-22**
– 0xFFF26164	Timer 1 Channel 1 Compare Load 2 Register	TMR1CMPLD21	**page 21-22**
– 0xFFF26168	Timer 1 Channel 1 Comparator Status and Control Register	TMR1COMSC1	**page 21-22**
– 0xFFF2616C–0xFFF2617F	reserved		
– 0xFFF26180	Timer 1 Channel 2 Compare 1 Register	TMR1CMP12	**page 21-21**
– 0xFFF26184	Timer 1 Channel 2 Compare 2 Register	TMR1CMP22	**page 21-21**
– 0xFFF26188	Timer 1 Channel 2 Capture Register	TMR1CAP2	**page 21-23**
– 0xFFF2618C	Timer 1 Channel 2 Load Register	TMR1LOAD2	**page 21-24**
– 0xFFF26190	Timer 1 Channel 2 Hold Register	TMR1HOLD2	**page 21-24**
– 0xFFF26194	Timer 1 Channel 2 Counter Register	TMR1CNTR2	**page 21-24**
– 0xFFF26198	Timer 1 Channel 2 Control Register	TMR1CTL2	**page 21-17**
– 0xFFF2619C	Timer 1 Channel 2 Status and Control Register	TMR1SCTL2	**page 21-19**
– 0xFFF261A0	Timer 1 Channel 2 Compare Load 1 Register	TMR1CMPLD12	**page 21-22**
– 0xFFF261A4	Timer 1 Channel 2 Compare Load 2 Register	TMR1CMPLD22	**page 21-22**
– 0xFFF261A8	Timer 1 Channel 2 Comparator Status and Control Register	TMR1COMSC2	**page 21-22**
– 0xFFF261AC–0xFFF261BF	reserved		
– 0xFFF261C0	Timer 1 Channel 3 Compare 1 Register	TMR1CMP13	**page 21-21**
– 0xFFF261C4	Timer 1 Channel 3 Compare 2 Register	TMR1CMP23	**page 21-21**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF261C8	Timer 1 Channel 3 Capture Register	TMR1CAP3	**page 21-23**
– 0xFFF261CC	Timer 1 Channel 3 Load Register	TMR1LOAD3	**page 21-24**
– 0xFFF261D0	Timer 1 Channel 3 Hold Register	TMR1HOLD3	**page 21-24**
– 0xFFF261D4	Timer 1 Channel 3 Counter Register	TMR1CNTR3	**page 21-24**
– 0xFFF261D8	Timer 1 Channel 3 Control Register	TMR1CTL3	**page 21-17**
– 0xFFF261DC	Timer 1 Channel 3 Status and Control Register	TMR1SCTL3	**page 21-19**
– 0xFFF261E0	Timer 1 Channel 3 Compare Load 1 Register	TMR1CMPLD13	**page 21-22**
– 0xFFF261E4	Timer 1 Channel 3 Compare Load 2 Register	TMR1CMPLD23	**page 21-22**
– 0xFFF261E8	Timer 1 Channel 3 Comparator Status and Control Register	TMR1COMSC3	**page 21-22**
– 0xFFF261EC–0xFFF261FF	reserved		
• 0xFFF26200–0xFFF262FF	Timer 2		
– 0xFFF26200	Timer 2 Channel 0 Compare 1 Register	TMR2CMP10	**page 21-21**
– 0xFFF26204	Timer 2 Channel 0 Compare 2 Register	TMR2CMP20	**page 21-21**
– 0xFFF26208	Timer 2 Channel 0 Capture Register	TMR2CAP0	**page 21-23**
– 0xFFF2620C	Timer 2 Channel 0 Load Register	TMR2LOAD0	**page 21-24**
– 0xFFF26210	Timer 2 Channel 0 Hold Register	TMR2HOLD0	**page 21-24**
– 0xFFF26214	Timer 2 Channel 0 Counter Register	TMR2CNTR0	**page 21-24**
– 0xFFF26218	Timer 2 Channel 0 Control Register	TMR2CTL0	**page 21-17**
– 0xFFF2621C	Timer 2 Channel 0 Status and Control Register	TMR2SCTL0	**page 21-19**
– 0xFFF26220	Timer 2 Channel 0 Compare Load 1 Register	TMR2CMPLD10	**page 21-22**
– 0xFFF26224	Timer 2 Channel 0 Compare Load 2 Register	TMR2CMPLD20	**page 21-22**
– 0xFFF26228	Timer 2 Channel 0 Comparator Status and Control Register	TMR2COMSC0	**page 21-22**
– 0xFFF2622C–0xFFF2623F	reserved		
– 0xFFF26240	Timer 2 Channel 1 Compare 1 Register	TMR2CMP11	**page 21-21**
– 0xFFF26244	Timer 2 Channel 1 Compare 2 Register	TMR2CMP21	**page 21-21**
– 0xFFF26248	Timer 2 Channel 1 Capture Register	TMR2CAP1	**page 21-23**
– 0xFFF2624C	Timer 2 Channel 1 Load Register	TMR2LOAD1	**page 21-24**
– 0xFFF26250	Timer 2 Channel 1 Hold Register	TMR2HOLD1	**page 21-24**
– 0xFFF26254	Timer 2 Channel 1 Counter Register	TMR2CNTR1	**page 21-24**
– 0xFFF26258	Timer 2 Channel 1 Control Register	TMR2CTL1	**page 21-17**
– 0xFFF2625C	Timer 2 Channel 1 Status and Control Register	TMR2SCTL1	**page 21-19**
– 0xFFF26260	Timer 2 Channel 1 Compare Load 1 Register	TMR2CMPLD11	**page 21-22**
– 0xFFF26264	Timer 2 Channel 1 Compare Load 2 Register	TMR2CMPLD21	**page 21-22**
– 0xFFF26268	Timer 2 Channel 1 Comparator Status and Control Register	TMR2COMSC1	**page 21-22**
– 0xFFF2626C–0xFFF2627F	reserved		
– 0xFFF26280	Timer 2 Channel 2 Compare 1 Register	TMR2CMP12	**page 21-21**
– 0xFFF26284	Timer 2 Channel 2 Compare 2 Register	TMR2CMP22	**page 21-21**
– 0xFFF26288	Timer 2 Channel 2 Capture Register	TMR2CAP2	**page 21-23**
– 0xFFF2628C	Timer 2 Channel 2 Load Register	TMR2LOAD2	**page 21-24**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF26290	Timer 2 Channel 2 Hold Register	TMR2HOLD2	**page 21-24**
– 0xFFF26294	Timer 2 Channel 2 Counter Register	TMR2CNTR2	**page 21-24**
– 0xFFF26298	Timer 2 Channel 2 Control Register	TMR2CTL2	**page 21-17**
– 0xFFF2629C	Timer 2 Channel 2 Status and Control Register	TMR2SCTL2	**page 21-19**
– 0xFFF262A0	Timer 2 Channel 2 Compare Load 1 Register	TMR2CMPLD12	**page 21-22**
– 0xFFF262A4	Timer 2 Channel 2 Compare Load 2 Register	TMR2CMPLD22	**page 21-22**
– 0xFFF262A8	Timer 2 Channel 2 Comparator Status and Control Register	TMR2COMSC2	**page 21-22**
– 0xFFF262AC–0xFFF262BF	reserved		
– 0xFFF262C0	Timer 2 Channel 3 Compare 1 Register	TMR2CMP13	**page 21-21**
– 0xFFF262C4	Timer 2 Channel 3 Compare 2 Register	TMR2CMP23	**page 21-21**
– 0xFFF262C8	Timer 2 Channel 3 Capture Register	TMR2CAP3	**page 21-23**
– 0xFFF262CC	Timer 2 Channel 3 Load Register	TMR2LOAD3	**page 21-24**
– 0xFFF262D0	Timer 2 Channel 3 Hold Register	TMR2HOLD3	**page 21-24**
– 0xFFF262D4	Timer 2 Channel 3 Counter Register	TMR2CNTR3	**page 21-24**
– 0xFFF262D8	Timer 2 Channel 3 Control Register	TMR2CTL3	**page 21-17**
– 0xFFF262DC	Timer 2 Channel 3 Status and Control Register	TMR2SCTL3	**page 21-19**
– 0xFFF262E0	Timer 2 Channel 3 Compare Load 1 Register	TMR2CMPLD13	**page 21-22**
– 0xFFF262E4	Timer 2 Channel 3 Compare Load 2 Register	TMR2CMPLD23	**page 21-22**
– 0xFFF262E8	Timer 2 Channel 3 Comparator Status and Control Register	TMR2COMSC3	**page 21-22**
– 0xFFF262EC–0xFFF262FF	reserved		
• 0xFFF26300–0xFFF263FF	Timer 3		
– 0xFFF26300	Timer 3 Channel 0 Compare 1 Register	TMR3CMP10	**page 21-21**
– 0xFFF26304	Timer 3 Channel 0 Compare 2 Register	TMR3CMP20	**page 21-21**
– 0xFFF26308	Timer 3 Channel 0 Capture Register	TMR3CAP0	**page 21-23**
– 0xFFF2630C	Timer 3 Channel 0 Load Register	TMR3LOAD0	**page 21-24**
– 0xFFF26310	Timer 3 Channel 0 Hold Register	TMR3HOLD0	**page 21-24**
– 0xFFF26314	Timer 3 Channel 0 Counter Register	TMR3CNTR0	**page 21-24**
– 0xFFF26318	Timer 3 Channel 0 Control Register	TMR3CTL0	**page 21-17**
– 0xFFF2631C	Timer 3 Channel 0 Status and Control Register	TMR3SCTL0	**page 21-19**
– 0xFFF26320	Timer 3 Channel 0 Load 1 Register	TMR3CMPLD10	**page 21-22**
– 0xFFF26324	Timer 3 Channel 0 Load 2 Register	TMR3CMPLD20	**page 21-22**
– 0xFFF26328	Timer 3 Channel 0 Comparator Status and Control Register	TMR3COMSC0	**page 21-22**
– 0xFFF2632C–0xFFF2633F	reserved		
– 0xFFF26340	Timer 3 Channel 1 Compare 1 Register	TMR3CMP11	**page 21-21**
– 0xFFF26344	Timer 3 Channel 1 Compare 2 Register	TMR3CMP21	**page 21-21**
– 0xFFF26348	Timer 3 Channel 1 Capture Register	TMR3CAP1	**page 21-23**
– 0xFFF2634C	Timer 3 Channel 1 Load Register	TMR3LOAD1	**page 21-24**
– 0xFFF26350	Timer 3 Channel 1 Hold Register	TMR3HOLD1	**page 21-24**
– 0xFFF26354	Timer 3 Channel 1 Counter Register	TMR3CNTR1	**page 21-24**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF26358	Timer 3 Channel 1 Control Register	TMR3CTL1	**page 21-17**
– 0xFFF2635C	Timer 3 Channel 1 Status and Control Register	TMR3SCTL1	**page 21-19**
– 0xFFF26360	Timer 3 Channel 1 Load 1 Register	TMR3CMPLD11	**page 21-22**
– 0xFFF26364	Timer 3 Channel 1 Load 2 Register	TMR3CMPLD21	**page 21-22**
– 0xFFF26368	Timer 3 Channel 1 Comparator Status and Control Register	TMR3COMSC1	**page 21-22**
– 0xFFF2636C–0xFFF2637F	reserved		
– 0xFFF26380	Timer 3 Channel 2 Compare 1 Register	TMR3CMP12	**page 21-21**
– 0xFFF26384	Timer 3 Channel 2 Compare 2 Register	TMR3CMP22	**page 21-21**
– 0xFFF26388	Timer 3 Channel 2 Capture Register	TMR3CAP2	**page 21-23**
– 0xFFF2638C	Timer 3 Channel 2 Load Register	TMR3LOAD2	**page 21-24**
– 0xFFF26390	Timer 3 Channel 2 Hold Register	TMR3HOLD2	**page 21-24**
– 0xFFF26394	Timer 3 Channel 2 Counter Register	TMR3CNTR2	**page 21-24**
– 0xFFF26398	Timer 3 Channel 2 Control Register	TMR3CTL2	**page 21-17**
– 0xFFF2639C	Timer 3 Channel 2 Status and Control Register	TMR3SCTL2	**page 21-19**
– 0xFFF263A0	Timer 3 Channel 2 Load 1 Register	TMR3CMPLD12	**page 21-22**
– 0xFFF263A4	Timer 3 Channel 2 Load 2 Register	TMR3CMPLD22	**page 21-22**
– 0xFFF263A8	Timer 3 Channel 2 Comparator Status and Control Register	TMR3COMSC2	**page 21-22**
– 0xFFF263AC–0xFFF263BF	reserved		
– 0xFFF263C0	Timer 3 Channel 3 Compare 1 Register	TMR3CMP13	**page 21-21**
– 0xFFF263C4	Timer 3 Channel 3 Compare 2 Register	TMR3CMP23	**page 21-21**
– 0xFFF263C8	Timer 3 Channel 3 Capture Register	TMR3CAP3	**page 21-23**
– 0xFFF263CC	Timer 3 Channel 3 Load Register	TMR3LOAD3	**page 21-24**
– 0xFFF263D0	Timer 3 Channel 3 Hold Register	TMR3HOLD3	**page 21-24**
– 0xFFF263D4	Timer 3 Channel 3 Counter Register	TMR3CNTR3	**page 21-24**
– 0xFFF263D8	Timer 3 Channel 3 Control Register	TMR3CTL3	**page 21-17**
– 0xFFF263DC	Timer 3 Channel 3 Status and Control Register	TMR3SCTL3	**page 21-19**
– 0xFFF263E0	Timer 3 Channel 3 Load 1 Register	TMR3CMPLD13	**page 21-22**
– 0xFFF263E4	Timer 3 Channel 3 Load 2 Register	TMR3CMPLD23	**page 21-22**
– 0xFFF263E8	Timer 3 Channel 3 Comparator Status and Control Register	TMR3COMSC3	**page 21-22**
– 0xFFF263EC–0xFFF263FF	reserved		
• 0xFFF26400–0xFFF26FFF	reserved		
• 0xFFF27000–0xFFF270FF	GIC		
– 0xFFF27000	Virtual Interrupt Generation Register	VIGR	**page 13-13**
– 0xFFF27004–0xFFF27007	reserved		
– 0xFFF27008	Virtual Interrupt Status Register	VISR	**page 13-14**
– 0xFFF2700C–0xFFF2700F	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF27010	Virtual NMI Generation Register	VNMIGR	**page 13-14**
– 0xFFF27014–0xFFF270FF	reserved		
• 0xFFF27100–0xFFF271FF	Hardware Semaphores		
– 0xFFF27100	Hardware Semaphore Register 0	HSMPR0	**page 23-2**
– 0xFFF27104–0xFFF27107	reserved		
– 0xFFF27108	Hardware Semaphore Register 1	HSMPR1	**page 23-2**
– 0xFFF2710C–0xFFF2711F	reserved		
– 0xFFF27110	Hardware Semaphore Register 2	HSMPR2	**page 23-2**
– 0xFFF27114–0xFFF27117	reserved		
– 0xFFF27118	Hardware Semaphore Register 3	HSMPR3	**page 23-2**
– 0xFFF2711C–0xFFF2711F	reserved		
– 0xFFF27120	Hardware Semaphore Register 4	HSMPR4	**page 23-2**
– 0xFFF27124–0xFFF27127	reserved		
– 0xFFF27128	Hardware Semaphore Register 5	HSMPR5	**page 23-2**
– 0xFFF2712C–0xFFF2712F	reserved		
– 0xFFF27130	Hardware Semaphore Register 6	HSMPR6	**page 23-2**
– 0xFFF27134–0xFFF27137	reserved		
– 0xFFF27138	Hardware Semaphore Register 7	HSMPR7	**page 23-2**
– 0xFFF2713C–0xFFF271FF	reserved		
• 0xFFF27200–0xFFF272FF	GPIO		
– 0xFFF27200	Pin Open-Drain Register	PODR	**page 22-7**
– 0xFFF27204–0xFFF27207	reserved		
– 0xFFF27208	Pin Data Register	PDAT	**page 22-8**
– 0xFFF2720C–0xFFF2720F	reserved		
– 0xFFF27210	Pin Data Direction Register	PDIR	**page 22-9**
– 0xFFF27214–0xFFF27217	reserved		
– 0xFFF27218	Pin Assignment Register	PAR	**page 22-9**
– 0xFFF2721C–0xFFF2721F	reserved		
– 0xFFF27220	Pin Special Options Register	PSOR	**page 22-10**
– 0xFFF27224–0xFFF272FF	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
• 0xFFF27300–0xFFF29FFF	reserved		
• 0xFFF2A000–0xFFF2AFFF	L2 ICache CLASS Initiator		
– 0xFFF2A000–0xFFF2AC03	reserved		
– 0xFFF2AC04	L2 ICache Cacheable Area Start Address	L2IC_CSA	**page 11-34**
– 0xFFF2AC08–0xFFF2AC43	reserved		
– 0xFFF2AC44	L2 ICache Cacheable Area End Address	L2IC_CEA	**page 11-35**
– 0xFFF2AC48–0xFFF2AC83	reserved		
– 0xFFF2AC84	L2 ICache Cacheable Area Enable	L2IC_CEN	**page 11-36**
– 0xFFF2AC88–0xFFF2AFFF	reserved		
• 0xFFF2B000–0xFFF2BFFF	L2 CLASS Target		
• 0xFFF2C000–0xFFF2C01F	L2 ICache Controller		
– 0xFFF2C000	L2 ICache Control Register 0	L2IC_CR0	**page 11-26**
– 0xFFF2C004	L2 ICache Control Register 1	L2IC_CR1	**page 11-26**
– 0xFFF2C008	L2 ICache Control Register 2	L2IC_CR2	**page 11-28**
– 0xFFF2C00C	L2 ICache LRM State Register	L2IC_LRM	**page 11-29**
– 0xFFF2C010	L2 ICache TAG State Register	L2IC_TAG	**page 11-31**
– 0xFFF2C014	L2 ICache Valid State Register	L2IC_VALID	**page 11-32**
– 0xFFF2C018	L2 ICache Data Register	L2IC_DBG_DATA	**page 11-32**
– 0xFFF2C01C	L2 ICache Debug Access Register	L2IC_DBG_ACS	**page 11-33**
• 0xFFF2C020–0xFFF2FFFF	reserved		
• 0xFFF30000–0xFFF33FFF	TDM0		
– 0xFFF30000–0xFFF307FF	TDM0 Receive Local Memory		
– 0xFFF30800–0xFFF30FFF	reserved		
– 0xFFF31000–0xFFF313FC	TDM0 Receive Channel Parameters Register 0–255	TDM0RCPR[0–255]	**page 19-62**
– 0xFFF31400–0xFFF317FF	reserved		
– 0xFFF31800–0xFFF31FFF	TDM0 Transmit Local Memory		
– 0xFFF32000–0xFFF327FF	reserved		
– 0xFFF32800–0xFFF32BFC	TDM0 Transmit Channel Parameters Register 0–255	TDM0TCPR[0–255]	**page 19-63**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF32C00–0xFFF33EFF	reserved		
– 0xFFF33F00	TDM0 Parity Control Register	TDM0PCR	**page 19-57**
– 0xFFF33F04–0xFFF33F07	reserved		
– 0xFFF33F08	TDM0 Parity Error Register	TDM0PER	**page 19-73**
– 0xFFF33F0C–0xFFF33F0F	reserved		
– 0xFFF33F10	TDM0 Transmit Force Register	TDM0TFR	**page 19-55**
– 0xFFF33F14–0xFFF33F17	reserved		
– 0xFFF33F18	TDM0 Receive Force Register	TDM0RFR	**page 19-56**
– 0xFFF33F1C–0xFFF33F1F	reserved		
– 0xFFF33F20	TDM0 Transmit Status Register	TDM0TSR	**page 19-73**
– 0xFFF33F24–0xFFF33F27	reserved		
– 0xFFF33F28	TDM0 Receive Status Register	TDM0RSR	**page 19-72**
– 0xFFF33F2C–0xFFF33F2F	reserved		
– 0xFFF33F30	TDM0 Adaptation Status Register	TDM0ASR	**page 19-71**
– 0xFFF33F34–0xFFF33F37	reserved		
– 0xFFF33F38	TDM0 Transmit Event Register	TDM0TER	**page 19-70**
– 0xFFF33F3C–0xFFF33F3F	reserved		
– 0xFFF33F40	TDM0 Receive Event Register	TDM0RER	**page 19-69**
– 0xFFF33F44–0xFFF33F47	reserved		
– 0xFFF33F48	TDM0 Transmit Number of Buffers	TDM0TNB	**page 19-69**
– 0xFFF33F4C–0xFFF33F4F	reserved		
– 0xFFF33F50	TDM0 Receive Number of Buffers	TDM0RNB	**page 19-68**
– 0xFFF33F54–0xFFF33F57	reserved		
– 0xFFF33F58	TDM0 Transmit Data Buffer Displacement Register	TDM0TDBDR	**page 19-67**
– 0xFFF33F5C–0xFFF33F5F	reserved		
– 0xFFF33F60	TDM0 Receive Data Buffer Displacement Register	TDM0RDBDR	**page 19-67**
– 0xFFF33F64–0xFFF33F67	reserved		
– 0xFFF33F68	TDM0 Adaptation Sync Distance Register	TDM0ASDR	**page 19-66**
– 0xFFF33F6C–0xFFF33F6F	reserved		
– 0xFFF33F70	TDM0 Transmit Interrupt Enable Register	TDM00TIER	**page 19-65**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF33F74–0xFFF33F77	reserved		
– 0xFFF33F78	TDM0 Receive Interrupt Enable Register	TDM0RIER	**page 19-64**
– 0xFFF33F7C–0xFFF33F7F	reserved		
– 0xFFF33F80	TDM0 Transmit Data Buffer Second Threshold	TDM0TDBST	**page 19-61**
– 0xFFF33F84–0xFFF33F87	reserved		
– 0xFFF33F88	TDM0 Receive Data Buffer Second Threshold	TDM0RDBST	**page 19-61**
– 0xFFF33F8C–0xFFF33F8F	reserved		
– 0xFFF33F90	TDM0 Transmit Data Buffer First Threshold	TDM0TDBFT	**page 19-60**
– 0xFFF33F94–0xFFF33F97	reserved		
– 0xFFF33F98	TDM0 Receive Data Buffer First Threshold	TDM0RDBFT	**page 19-59**
– 0xFFF33F9C–0xFFF33F9F	reserved		
– 0xFFF33FA0	TDM0 Transmit Control Register	TDM0TCR	**page 19-59**
– 0xFFF33FA4–0xFFF33FA7	reserved		
– 0xFFF33FA8	TDM0 Receive Control Register	TDM0RCR	**page 19-58**
– 0xFFF33FAC–0xFFF33FAF	reserved		
– 0xFFF33FB0	TDM0 Adaptation Control Register	TDM0ACR	**page 19-57**
– 0xFFF33FB4–0xFFF33FB7	reserved		
– 0xFFF33FB8	TDM0 Transmit Global Base Address	TDM0TGBA	**page 19-55**
– 0xFFF33FBC–0xFFF33FBF	reserved		
– 0xFFF33FC0	TDM0 Receive Global Base Address	TDM0RGBA	**page 19-54**
– 0xFFF33FC4–0xFFF33FC7	reserved		
– 0xFFF33FC8	TDM0 Transmit Data Buffer Size	TDM0TDBS	**page 19-54**
– 0xFFF33FCC–0xFFF33FCF	reserved		
– 0xFFF33FD0	TDM0 Receive Data Buffer Size	TDM0RDBS	**page 19-53**
– 0xFFF33FD4–0xFFF33FD7	reserved		
– 0xFFF33FD8	TDM0 Transmit Frame Parameters	TDM0TFP	**page 19-51**
– 0xFFF33FDC–0xFFF33FDF	reserved		
– 0xFFF33FE0	TDM0 Receive Frame Parameters	TDM0RFP	**page 19-48**
– 0xFFF33FE4–0xFFF33FE7	reserved		
– 0xFFF33FE8	TDM0 Transmit Interface Register	TDM0TIR	**page 19-46**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF33FEC–0xFFF33FEF	reserved		
– 0xFFF33FF0	TDM0 Receive Interface Register	TDM0RIR	**page 19-44**
– 0xFFF33FF4–0xFFF33FF7	reserved		
– 0xFFF33FF8	TDM0 General Interface Register	TDM0GIR	**page 19-36**
– 0xFFF33FFC–0xFFF33FFF	reserved		
• 0xFFF34000–0xFFF37FFF	TDM1		
– 0xFFF34000–0xFFF347FF	TDM1 Receive Local Memory		
– 0xFFF34800–0xFFF34FFF	reserved		
– 0xFFF35000–0xFFF353FC	TDM1 Receive Channel Parameters Register 0–255	TDM1RCPR[0–255]	**page 19-62**
– 0xFFF35400–0xFFF357FF	reserved		
– 0xFFF35800–0xFFF35FFF	TDM1 Transmit Local Memory		
– 0xFFF36000–0xFFF367FF	reserved		
– 0xFFF36800–0xFFF36BFC	TDM1 Transmit Channel Parameters Register 0–255	TDM1TCPR[0–255]	**page 19-63**
– 0xFFF36C00–0xFFF37EFF	reserved		
– 0xFFF37F00	TDM1 Parity Control Register	TDM1PCR	**page 19-57**
– 0xFFF37F04–0xFFF37F07	reserved		
– 0xFFF37F08	TDM1 Parity Error Register	TDM1PER	**page 19-73**
– 0xFFF37F0C–0xFFF37F0F	reserved		
– 0xFFF37F10	TDM1 Transmit Force Register	TDM1TFR	**page 19-55**
– 0xFFF37F14–0xFFF37F17	reserved		
– 0xFFF37F18	TDM1 Receive Force Register	TDM1RFR	**page 19-56**
– 0xFFF37F1C–0xFFF37F1F	reserved		
– 0xFFF37F20	TDM1 Transmit Status Register	TDM1TSR	**page 19-73**
– 0xFFF37F24–0xFFF37F27	reserved		
– 0xFFF37F28	TDM1 Receive Status Register	TDM1RSR	**page 19-72**
– 0xFFF37F2C–0xFFF37F2F	reserved		
– 0xFFF37F30	TDM1 Adaptation Status Register	TDM1ASR	**page 19-71**
– 0xFFF37F34–0xFFF37F37	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF37F38	TDM1 Transmit Event Register	TDM1TER	**page 19-70**
– 0xFFF37F3C–0xFFF37F3F	reserved		
– 0xFFF37F40	TDM1 Receive Event Register	TDM1RER	**page 19-69**
– 0xFFF37F44–0xFFF37F47	reserved		
– 0xFFF37F48	TDM1 Transmit Number of Buffers	TDM1TNB	**page 19-69**
– 0xFFF37F4C–0xFFF37F4F	reserved		
– 0xFFF37F50	TDM1 Receive Number of Buffers	TDM1RNB	**page 19-68**
– 0xFFF37F54–0xFFF37F57	reserved		
– 0xFFF37F58	TDM1 Transmit Data Buffer Displacement Register	TDM1TDBDR	**page 19-67**
– 0xFFF37F5C–0xFFF37F5F	reserved		
– 0xFFF37F60	TDM1 Receive Data Buffer Displacement Register	TDM1RDBDR	**page 19-67**
– 0xFFF37F64–0xFFF37F67	reserved		
– 0xFFF37F68	TDM1 Adaptation Sync Distance Register	TDM1ASDR	**page 19-66**
– 0xFFF37F6C–0xFFF37F6F	reserved		
– 0xFFF37F70	TDM1 Transmit Interrupt Enable Register	TDM10TIER	**page 19-65**
– 0xFFF37F74–0xFFF37F77	reserved		
– 0xFFF37F78	TDM1 Receive Interrupt Enable Register	TDM1RIER	**page 19-64**
– 0xFFF37F7C–0xFFF37F7F	reserved		
– 0xFFF37F80	TDM1 Transmit Data Buffer Second Threshold	TDM1TDBST	**page 19-61**
– 0xFFF37F84–0xFFF37F87	reserved		
– 0xFFF37F88	TDM1 Receive Data Buffer Second Threshold	TDM1RDBST	**page 19-61**
– 0xFFF37F8C–0xFFF37F8F	reserved		
– 0xFFF37F90	TDM1 Transmit Data Buffer First Threshold	TDM1TDBFT	**page 19-60**
– 0xFFF37F94–0xFFF37F97	reserved		
– 0xFFF37F98	TDM1 Receive Data Buffer First Threshold	TDM1RDBFT	**page 19-59**
– 0xFFF37F9C–0xFFF37F9F	reserved		
– 0xFFF37FA0	TDM1 Transmit Control Register	TDM1TCR	**page 19-59**
– 0xFFF37FA4–0xFFF37FA7	reserved		
– 0xFFF37FA8	TDM1 Receive Control Register	TDM1RCR	**page 19-58**
– 0xFFF37FAC–0xFFF37FAF	reserved		
– 0xFFF37FB0	TDM1 Adaptation Control Register	TDM1ACR	**page 19-57**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF37FB4–0xFFF37FB7	reserved		
– 0xFFF37FB8	TDM1 Transmit Global Base Address	TDM1TGBA	**page 19-55**
– 0xFFF37FBC–0xFFF37FBF	reserved		
– 0xFFF37FC0	TDM1 Receive Global Base Address	TDM1RGBA	**page 19-54**
– 0xFFF37FC4–0xFFF37FC7	reserved		
– 0xFFF37FC8	TDM1 Transmit Data Buffer Size	TDM1TDBS	**page 19-54**
– 0xFFF37FCC–0xFFF37FCF	reserved		
– 0xFFF37FD0	TDM1 Receive Data Buffer Size	TDM1RDBS	**page 19-53**
– 0xFFF37FD4–0xFFF37FD7	reserved		
– 0xFFF37FD8	TDM1 Transmit Frame Parameters	TDM1TFP	**page 19-51**
– 0xFFF37FDC–0xFFF37FDF	reserved		
– 0xFFF37FE0	TDM1 Receive Frame Parameters	TDM1RFP	**page 19-48**
– 0xFFF37FE4–0xFFF37FE7	reserved		
– 0xFFF37FE8	TDM1 Transmit Interface Register	TDM1TIR	**page 19-46**
– 0xFFF37FEC–0xFFF37FEF	reserved		
– 0xFFF37FF0	TDM1 Receive Interface Register	TDM1RIR	**page 19-44**
– 0xFFF37FF4–0xFFF37FF7	reserved		
– 0xFFF37FF8	TDM1 General Interface Register	TDM1GIR	**page 19-36**
– 0xFFF37FFC–0xFFF37FFF	reserved		
• 0xFFF38000–0xFFF3BFFF	TDM2		
– 0xFFF38000–0xFFF387FF	TDM2 Receive Local Memory		
– 0xFFF38800–0xFFF38FFF	reserved		
– 0xFFF39000–0xFFF393FC	TDM2 Receive Channel Parameters Register 0–255	TDM2RCPR[0–255]	**page 19-62**
– 0xFFF39400–0xFFF397FF	reserved		
– 0xFFF39800–0xFFF39FFF	TDM2 Transmit Local Memory		
– 0xFFF3A000–0xFFF3A7FF	reserved		
– 0xFFF3A800–0xFFF3ABFC	TDM2 Transmit Channel Parameters Register 0–255	TDM2TCPR[0–255]	**page 19-63**
– 0xFFF3AC00–0xFFF3BEFF	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF3BF00	TDM2 Parity Control Register	TDM2PCR	**page 19-57**
– 0xFFF3BF04–0xFFF3BF07	reserved		
– 0xFFF3BF08	TDM2 Parity Error Register	TDM2PER	**page 19-73**
– 0xFFF3BF0C–0xFFF3BF0F	reserved		
– 0xFFF3BF10	TDM2 Transmit Force Register	TDM2TFR	**page 19-55**
– 0xFFF3BF14–0xFFF3BF17	reserved		
– 0xFFF3BF18	TDM2 Receive Force Register	TDM2RFR	**page 19-56**
– 0xFFF3BF1C–0xFFF3BF1F	reserved		
– 0xFFF3BF20	TDM2 Transmit Status Register	TDM2TSR	**page 19-73**
– 0xFFF3BF24–0xFFF3BF27	reserved		
– 0xFFF3BF28	TDM2 Receive Status Register	TDM2RSR	**page 19-72**
– 0xFFF3BF2C–0xFFF3BF2F	reserved		
– 0xFFF3BF30	TDM2 Adaptation Status Register	TDM2ASR	**page 19-71**
– 0xFFF3BF34–0xFFF38F37	reserved		
– 0xFFF3BF38	TDM2 Transmit Event Register	TDM2TER	**page 19-70**
– 0xFFF3BF3C–0xFFF3BF3F	reserved		
– 0xFFF3BF40	TDM2 Receive Event Register	TDM2RER	**page 19-69**
– 0xFFF3BF44–0xFFF3BF47	reserved		
– 0xFFF3BF48	TDM2 Transmit Number of Buffers	TDM2TNB	**page 19-69**
– 0xFFF3BF4C–0xFFF3BF4F	reserved		
– 0xFFF3BF50	TDM2 Receive Number of Buffers	TDM2RNB	**page 19-68**
– 0xFFF3BF54–0xFFF3BF57	reserved		
– 0xFFF3BF58	TDM2 Transmit Data Buffer Displacement Register	TDM2TDBDR	**page 19-67**
– 0xFFF3BF5C–0xFFF3BF5F	reserved		
– 0xFFF3BF60	TDM2 Receive Data Buffer Displacement Register	TDM2RDBDR	**page 19-67**
– 0xFFF3BF64–0xFFF3BF67	reserved		
– 0xFFF3BF68	TDM2 Adaptation Sync Distance Register	TDM2ASDR	**page 19-66**
– 0xFFF3BF6C–0xFFF3BF6F	reserved		
– 0xFFF3BF70	TDM2 Transmit Interrupt Enable Register	TDM20TIER	**page 19-65**
– 0xFFF3BF74–0xFFF3BF77	reserved		
– 0xFFF3BF78	TDM2 Receive Interrupt Enable Register	TDM2RIER	**page 19-64**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF3BF7C–0xFFF3BF7F	reserved		
– 0xFFF3BF80	TDM2 Transmit Data Buffer Second Threshold	TDM2TDBST	**page 19-61**
– 0xFFF3BF84–0xFFF3BF87	reserved		
– 0xFFF3BF88	TDM2 Receive Data Buffer Second Threshold	TDM2RDBST	**page 19-61**
– 0xFFF3BF8C–0xFFF3BF8F	reserved		
– 0xFFF3BF90–0xFFF38F94	TDM2 Transmit Data Buffer First Threshold	TDM2TDBFT	**page 19-60**
– 0xFFF3BF98	TDM2 Receive Data Buffer First Threshold	TDM2RDBFT	**page 19-59**
– 0xFFF3BF9C–0xFFF3BF9F	reserved		
– 0xFFF3BFA0	TDM2 Transmit Control Register	TDM2TCR	**page 19-59**
– 0xFFF3BFA4–0xFFF3BFA7	reserved		
– 0xFFF3BFA8	TDM2 Receive Control Register	TDM2RCR	**page 19-58**
– 0xFFF3BFAC–0xFFF3BFAF	reserved		
– 0xFFF3BFB0	TDM2 Adaptation Control Register	TDM2ACR	**page 19-57**
– 0xFFF3BFB4–0xFFF3BFB7	reserved		
– 0xFFF3BFB8	TDM2 Transmit Global Base Address	TDM2TGBA	**page 19-55**
– 0xFFF3BFBC–0xFFF3BFBF	reserved		
– 0xFFF3BFC0	TDM2 Receive Global Base Address	TDM2RGBA	**page 19-54**
– 0xFFF3BFC4–0xFFF3BFC7	reserved		
– 0xFFF3BFC8	TDM2 Transmit Data Buffer Size	TDM2TDBS	**page 19-54**
– 0xFFF3BFCC–0xFFF3BFCF	reserved		
– 0xFFF3BFD0	TDM2 Receive Data Buffer Size	TDM2RDBS	**page 19-53**
– 0xFFF3BFD4–0xFFF3BFD7	reserved		
– 0xFFF3BFD8	TDM2 Transmit Frame Parameters	TDM2TFP	**page 19-51**
– 0xFFF3BFDC–0xFFF3BFDF	reserved		
– 0xFFF3BFE0	TDM2 Receive Frame Parameters	TDM2RFP	**page 19-48**
– 0xFFF3BFE4–0xFFF3BFE7	reserved		
– 0xFFF3BFE8	TDM2 Transmit Interface Register	TDM2TIR	**page 19-46**
– 0xFFF3BFEC–0xFFF3BFEF	reserved		
– 0xFFF3BFF0	TDM2 Receive Interface Register	TDM2RIR	**page 19-44**
– 0xFFF3BFF4–0xFFF3BFF7	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF3BFF8	TDM2 General Interface Register	TDM2GIR	**page 19-36**
– 0xFFF3BFFC–0xFFF3BFFF	reserved		
• 0xFFF3C000–0xFFF3FFFF	TDM3		
– 0xFFF3C000–0xFFF3C7FF	TDM3 Receive Local Memory		
– 0xFFF3C800–0xFFF3CFFF	reserved		
– 0xFFF3D000–0xFFF3D3FC	TDM3 Receive Channel Parameters Register 0–255	TDM3RCPR[0–255]	**page 19-62**
– 0xFFF3D400–0xFFF3D7FF	reserved		
– 0xFFF3D800–0xFFF3DFFF	TDM3 Transmit Local Memory		
– 0xFFF3E000–0xFFF3E7FF	reserved		
– 0xFFF3E800–0xFFF3EBFC	TDM3 Transmit Channel Parameters Register 0–255	TDM3TCPR[0–255]	**page 19-63**
– 0xFFF3EC00–0xFFF3FEFF	reserved		
– 0xFFF3FF00	TDM3 Parity Control Register	TDM3PCR	**page 19-57**
– 0xFFF3FFF04–0xFFF3FF07	reserved		
– 0xFFF3FF08	TDM3 Parity Error Register	TDM3PER	**page 19-73**
– 0xFFF3FF0C–0xFFF3FF0F	reserved		
– 0xFFF3FF10	TDM3 Transmit Force Register	TDM3TFR	**page 19-55**
– 0xFFF3FF14–0xFFF3FF17	reserved		
– 0xFFF3FF18	TDM3 Receive Force Register	TDM3RFR	**page 19-56**
– 0xFFF3FF1C–0xFFF3FF1F	reserved		
– 0xFFF3FF20	TDM3 Transmit Status Register	TDM3TSR	**page 19-73**
– 0xFFF3FF24–0xFFF3FF27	reserved		
– 0xFFF3FF28	TDM3 Receive Status Register	TDM3RSR	**page 19-72**
– 0xFFF3FF2C–0xFFF3FF2F	reserved		
– 0xFFF3FF30	TDM3 Adaptation Status Register	TDM3ASR	**page 19-71**
– 0xFFF3FF34–0xFFF3FF37	reserved		
– 0xFFF3FF38	TDM3 Transmit Event Register	TDM3TER	**page 19-70**
– 0xFFF3FF3C–0xFFF3FF3F	reserved		
– 0xFFF3FF40	TDM3 Receive Event Register	TDM3RER	**page 19-69**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF3FF44–0xFFF3FF47	reserved		
– 0xFFF3FF48	TDM3 Transmit Number of Buffers	TDM3TNB	**page 19-69**
– 0xFFF3FF4C–0xFFF3FF4F	reserved		
– 0xFFF3FF50	TDM3 Receive Number of Buffers	TDM3RNB	**page 19-68**
– 0xFFF3FF54–0xFFF3FF57	reserved		
– 0xFFF3FF58	TDM3 Transmit Data Buffer Displacement Register	TDM3TDBDR	**page 19-67**
– 0xFFF3FF5C–0xFFF3FF5F	reserved		
– 0xFFF3FF60	TDM3 Receive Data Buffer Displacement Register	TDM3RDBDR	**page 19-67**
– 0xFFF3FF64–0xFFF3FF67	reserved		
– 0xFFF3FF68	TDM3 Adaptation Sync Distance Register	TDM3ASDR	**page 19-66**
– 0xFFF3FF6C–0xFFF3FF6F	reserved		
– 0xFFF3FF70	TDM3 Transmit Interrupt Enable Register	TDM30TIER	**page 19-65**
– 0xFFF3FF74–0xFFF3FF77	reserved		
– 0xFFF3FF78	TDM3 Receive Interrupt Enable Register	TDM3RIER	**page 19-64**
– 0xFFF3FF7C–0xFFF3FF7F	reserved		
– 0xFFF3FF80	TDM3 Transmit Data Buffer Second Threshold	TDM3TDBST	**page 19-61**
– 0xFFF3FF84–0xFFF3FF87	reserved		
– 0xFFF3FF88	TDM3 Receive Data Buffer Second Threshold	TDM3RDBST	**page 19-61**
– 0xFFF3FF8C–0xFFF3FF8F	reserved		
– 0xFFF3FF90	TDM3 Transmit Data Buffer First Threshold	TDM3TDBFT	**page 19-60**
– 0xFFF3FF94–0xFFF3FF97	reserved		
– 0xFFF3FF98	TDM3 Receive Data Buffer First Threshold	TDM3RDBFT	**page 19-59**
– 0xFFF3FF9C–0xFFF3FF9F	reserved		
– 0xFFF3FFA0	TDM3 Transmit Control Register	TDM3TCR	**page 19-59**
– 0xFFF3FFA4–0xFFF3FFA7	reserved		
– 0xFFF3FFA8	TDM3 Receive Control Register	TDM3RCR	**page 19-58**
– 0xFFF3FFAC–0xFFF3FFAF	reserved		
– 0xFFF3FFB0	TDM3 Adaptation Control Register	TDM3ACR	**page 19-57**
– 0xFFF3FFB4–0xFFF3FFB7	reserved		
– 0xFFF3FFB8	TDM3 Transmit Global Base Address	TDM3TGBA	**page 19-55**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF3FFBC–0xFFF3FFBF	reserved		
– 0xFFF3FFC0	TDM3 Receive Global Base Address	TDM3RGBA	**page 19-54**
– 0xFFF3FFC4–0xFFF3FFC7	reserved		
– 0xFFF3FFC8	TDM3 Transmit Data Buffer Size	TDM3TDBS	**page 19-54**
– 0xFFF3FFCC–0xFFF3FFCF	reserved		
– 0xFFF3FFD0	TDM3 Receive Data Buffer Size	TDM3RDBS	**page 19-53**
– 0xFFF3FFD4–0xFFF3FFD7	reserved		
– 0xFFF3FFD8	TDM3 Transmit Frame Parameters	TDM3TFP	**page 19-51**
– 0xFFF3FFDC–0xFFF3FFDF	reserved		
– 0xFFF3FFE0	TDM3 Receive Frame Parameters	TDM3RFP	**page 19-48**
– 0xFFF3FFE4–0xFFF3FFE7	reserved		
– 0xFFF3FFE8	TDM3 Transmit Interface Register	TDM3TIR	**page 19-46**
– 0xFFF3FFEC–0xFFF3FFEF	reserved		
– 0xFFF3FFF0	TDM3 Receive Interface Register	TDM3RIR	**page 19-44**
– 0xFFF3FFF4–0xFFF3FFF7	reserved		
– 0xFFF3FFF8	TDM3 General Interface Register	TDM3GIR	**page 19-36**
– 0xFFF3FFFC–0xFFF3FFFF	reserved		
• 0xFFF40000–0xFFF43FFF	TDM4		
– 0xFFF40000–0xFFF407FF	TDM4 Receive Local Memory		
– 0xFFF40800–0xFFF40FFF	reserved		
– 0xFFF41000–0xFFF413FC	TDM4 Receive Channel Parameters Register 0–255	TDM4RCPR[0–255]	**page 19-62**
– 0xFFF41400–0xFFF417FF	reserved		
– 0xFFF41800–0xFFF41FFF	TDM4 Transmit Local Memory		
– 0xFFF42000–0xFFF427FF	reserved		
– 0xFFF42800–0xFFF42BFC	TDM4 Transmit Channel Parameters Register 0–255	TDM4TCPR[0–255]	**page 19-63**
– 0xFFF42C00–0xFFF43EFF	reserved		
– 0xFFF43F00	TDM4 Parity Control Register	TDM4PCR	**page 19-57**
– 0xFFF43F04–0xFFF43F07	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF43F08	TDM4 Parity Error Register	TDM4PER	**page 19-73**
– 0xFFF43F0C–0xFFF43F0F	reserved		
– 0xFFF43F10	TDM4 Transmit Force Register	TDM4TFR	**page 19-55**
– 0xFFF43F14–0xFFF43F17	reserved		
– 0xFFF43F18	TDM4 Receive Force Register	TDM4RFR	**page 19-56**
– 0xFFF43F1C–0xFFF43F1F	reserved		
– 0xFFF43F20	TDM4 Transmit Status Register	TDM4TSR	**page 19-73**
– 0xFFF43F24–0xFFF43F27	reserved		
– 0xFFF43F28	TDM4 Receive Status Register	TDM4RSR	**page 19-72**
– 0xFFF43F2C–0xFFF43F2F	reserved		
– 0xFFF43F30	TDM4 Adaptation Status Register	TDM4ASR	**page 19-71**
– 0xFFF43F34–0xFFF43F37	reserved		
– 0xFFF43F38	TDM4 Transmit Event Register	TDM4TER	**page 19-70**
– 0xFFF43F3C–0xFFF43F3F	reserved		
– 0xFFF43F40	TDM4 Receive Event Register	TDM4RER	**page 19-69**
– 0xFFF43F44–0xFFF43F47	reserved		
– 0xFFF43F48	TDM4 Transmit Number of Buffers	TDM4TNB	**page 19-69**
– 0xFFF43F4C–0xFFF43F4F	reserved		
– 0xFFF43F50	TDM4 Receive Number of Buffers	TDM4RNB	**page 19-68**
– 0xFFF43F54–0xFFF43F57	reserved		
– 0xFFF43F58	TDM4 Transmit Data Buffer Displacement Register	TDM4TDBDR	**page 19-67**
– 0xFFF43F5C–0xFFF43F5F	reserved		
– 0xFFF43F60	TDM4 Receive Data Buffer Displacement Register	TDM4RDBDR	**page 19-67**
– 0xFFF43F64–0xFFF43F67	reserved		
– 0xFFF43F68	TDM4 Adaptation Sync Distance Register	TDM4ASDR	**page 19-66**
– 0xFFF43F6C–0xFFF43F6F	reserved		
– 0xFFF43F70	TDM4 Transmit Interrupt Enable Register	TDM40TIER	**page 19-65**
– 0xFFF43F74–0xFFF43F77	reserved		
– 0xFFF43F78	TDM4 Receive Interrupt Enable Register	TDM4RIER	**page 19-64**
– 0xFFF43F7C–0xFFF43F7F	reserved		
– 0xFFF43F80	TDM4 Transmit Data Buffer Second Threshold	TDM4TDBST	**page 19-61**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF43F84–0xFFF43F87	reserved		
– 0xFFF43F88	TDM4 Receive Data Buffer Second Threshold	TDM4RDBST	**page 19-61**
– 0xFFF43F8C–0xFFF43F8F	reserved		
– 0xFFF43F90	TDM4 Transmit Data Buffer First Threshold	TDM4TDBFT	**page 19-60**
– 0xFFF43F94–0xFFF43F97	reserved		
– 0xFFF43F98	TDM4 Receive Data Buffer First Threshold	TDM4RDBFT	**page 19-59**
– 0xFFF43F9C–0xFFF43F9F	reserved		
– 0xFFF43FA0	TDM4 Transmit Control Register	TDM4TCR	**page 19-59**
– 0xFFF43FA4–0xFFF43FA7	reserved		
– 0xFFF43FA8	TDM4 Receive Control Register	TDM4RCR	**page 19-58**
– 0xFFF43FAC–0xFFF43FAF	reserved		
– 0xFFF43FB0	TDM4 Adaptation Control Register	TDM4ACR	**page 19-57**
– 0xFFF43FB4–0xFFF43FB7	reserved		
– 0xFFF43FB8	TDM4 Transmit Global Base Address	TDM4TGBA	**page 19-55**
– 0xFFF43FBC–0xFFF43FBF	reserved		
– 0xFFF43FC0	TDM4 Receive Global Base Address	TDM4RGBA	**page 19-54**
– 0xFFF43FC4–0xFFF43FC7	reserved		
– 0xFFF43FC8	TDM4 Transmit Data Buffer Size	TDM4TDBS	**page 19-54**
– 0xFFF43FCC–0xFFF43FCF	reserved		
– 0xFFF43FD0	TDM4 Receive Data Buffer Size	TDM4RDBS	**page 19-53**
– 0xFFF43FD4–0xFFF43FD7	reserved		
– 0xFFF43FD8	TDM4 Transmit Frame Parameters	TDM4TFP	**page 19-51**
– 0xFFF43FDC–0xFFF43FDF	reserved		
– 0xFFF43FE0	TDM4 Receive Frame Parameters	TDM4RFP	**page 19-48**
– 0xFFF43FE4–0xFFF43F37	reserved		
– 0xFFF43FE8	TDM4 Transmit Interface Register	TDM4TIR	**page 19-46**
– 0xFFF43FEC–0xFFF43FEF	reserved		
– 0xFFF43FF0	TDM4 Receive Interface Register	TDM4RIR	**page 19-44**
– 0xFFF43FF4–0xFFF43FF7	reserved		
– 0xFFF43FF8	TDM4 General Interface Register	TDM4GIR	**page 19-36**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF43FFC–0xFFF43FFF	reserved		
• 0xFFF44000–0xFFF47FFF	TDM5		
– 0xFFF44000–0xFFF447FF	TDM5 Receive Local Memory		
– 0xFFF44800–0xFFF44FFF	reserved		
– 0xFFF45000–0xFFF453FC	TDM5 Receive Channel Parameters Register 0–255	TDM5RCPR[0–255]	**page 19-62**
– 0xFFF45400–0xFFF457FF	reserved		
– 0xFFF45800–0xFFF45FFF	TDM5 Transmit Local Memory		
– 0xFFF46000–0xFFF467FF	reserved		
– 0xFFF46800–0xFFF46BFC	TDM5 Transmit Channel Parameters Register 0–255	TDM5TCPR[0–255]	**page 19-63**
– 0xFFF46C00–0xFFF47EFF	reserved		
– 0xFFF47F00	TDM5 Parity Control Register	TDM5PCR	**page 19-57**
– 0xFFF47F04–0xFFF47F07	reserved		
– 0xFFF47F08	TDM5 Parity Error Register	TDM5PER	**page 19-73**
– 0xFFF47F0C–0xFFF47F0F	reserved		
– 0xFFF47F10	TDM5 Transmit Force Register	TDM5TFR	**page 19-55**
– 0xFFF47F14–0xFFF47F17	reserved		
– 0xFFF47F18	TDM5 Receive Force Register	TDM5RFR	**page 19-56**
– 0xFFF47F1C–0xFFF47F1F	reserved		
– 0xFFF47F20	TDM5 Transmit Status Register	TDM5TSR	**page 19-73**
– 0xFFF47F24–0xFFF47F27	reserved		
– 0xFFF47F28	TDM5 Receive Status Register	TDM5RSR	**page 19-72**
– 0xFFF47F2C–0xFFF47F2F	reserved		
– 0xFFF47F30	TDM5 Adaptation Status Register	TDM5ASR	**page 19-71**
– 0xFFF47F34–0xFFF47F37	reserved		
– 0xFFF47F38	TDM5 Transmit Event Register	TDM5TER	**page 19-70**
– 0xFFF47F3C–0xFFF47F3F	reserved		
– 0xFFF47F40	TDM5 Receive Event Register	TDM5RER	**page 19-69**
– 0xFFF47F44–0xFFF47F47	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF47F48	TDM5 Transmit Number of Buffers	TDM5TNB	**page 19-69**
– 0xFFF47F4C–0xFFF47F4F	reserved		
– 0xFFF47F50	TDM5 Receive Number of Buffers	TDM5RNB	**page 19-68**
– 0xFFF47F54–0xFFF47F57	reserved		
– 0xFFF47F58	TDM5 Transmit Data Buffer Displacement Register	TDM5TDBDR	**page 19-67**
– 0xFFF47F5C–0xFFF47F5F	reserved		
– 0xFFF47F60	TDM5 Receive Data Buffer Displacement Register	TDM5RDBDR	**page 19-67**
– 0xFFF47F64–0xFFF47F67	reserved		
– 0xFFF47F68	TDM5 Adaptation Sync Distance Register	TDM5ASDR	**page 19-66**
– 0xFFF47F6C–0xFFF47F6F	reserved		
– 0xFFF47F70	TDM5 Transmit Interrupt Enable Register	TDM50TIER	**page 19-65**
– 0xFFF47F74–0xFFF47F77	reserved		
– 0xFFF47F78	TDM5 Receive Interrupt Enable Register	TDM5RIER	**page 19-64**
– 0xFFF47F7C–0xFFF47F7F	reserved		
– 0xFFF47F80	TDM5 Transmit Data Buffer Second Threshold	TDM5TDBST	**page 19-61**
– 0xFFF47F84–0xFFF47F87	reserved		
– 0xFFF47F88	TDM5 Receive Data Buffer Second Threshold	TDM5RDBST	**page 19-61**
– 0xFFF47F8C–0xFFF47F8F	reserved		
– 0xFFF47F90	TDM5 Transmit Data Buffer First Threshold	TDM5TDBFT	**page 19-60**
– 0xFFF47F94–0xFFF47F97	reserved		
– 0xFFF47F98	TDM5 Receive Data Buffer First Threshold	TDM5RDBFT	**page 19-59**
– 0xFFF47F9C–0xFFF47F9F	reserved		
– 0xFFF47FA0	TDM5 Transmit Control Register	TDM5TCR	**page 19-59**
– 0xFFF47FA4–0xFFF47FA7	reserved		
– 0xFFF47FA8	TDM5 Receive Control Register	TDM5RCR	**page 19-58**
– 0xFFF47FAC–0xFFF47FAF	reserved		
– 0xFFF47FB0	TDM5 Adaptation Control Register	TDM5ACR	**page 19-57**
– 0xFFF47FB4–0xFFF47FB7	reserved		
– 0xFFF47FB8	TDM5 Transmit Global Base Address	TDM5TGBA	**page 19-55**
– 0xFFF47FBC–0xFFF47FBF	reserved		
– 0xFFF47FC0	TDM5 Receive Global Base Address	TDM5RGBA	**page 19-54**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF47FC4–0xFFF47FC7	reserved		
– 0xFFF47FC8	TDM5 Transmit Data Buffer Size	TDM5TDBS	**page 19-54**
– 0xFFF47FCC–0xFFF47FCF	reserved		
– 0xFFF47FD0	TDM5 Receive Data Buffer Size	TDM5RDBS	**page 19-53**
– 0xFFF47FD4–0xFFF47FD7	reserved		
– 0xFFF47FD8	TDM5 Transmit Frame Parameters	TDM5TFP	**page 19-51**
– 0xFFF47FDC–0xFFF47FDF	reserved		
– 0xFFF47FE0	TDM5 Receive Frame Parameters	TDM5RFP	**page 19-48**
– 0xFFF47FE4–0xFFF47FE7	reserved		
– 0xFFF47FE8	TDM5 Transmit Interface Register	TDM5TIR	**page 19-46**
– 0xFFF47FEC–0xFFF47FEF	reserved		
– 0xFFF47FF0	TDM5 Receive Interface Register	TDM5RIR	**page 19-44**
– 0xFFF47FF4–0xFFF47FF7	reserved		
– 0xFFF47FF8	TDM5 General Interface Register	TDM5GIR	**page 19-36**
– 0xFFF47FFC–0xFFF47FFF	reserved		
• 0xFFF48000–0xFFF4BFFF	TDM6		
– 0xFFF48000–0xFFF487FF	TDM6 Receive Local Memory		
– 0xFFF48800–0xFFF48FFF	reserved		
– 0xFFF49000–0xFFF493FC	TDM6 Receive Channel Parameters Register 0–255	TDM6RCPR[0–255]	**page 19-62**
– 0xFFF49400–0xFFF497FF	reserved		
– 0xFFF49800–0xFFF49FFF	TDM6 Transmit Local Memory		
– 0xFFF4A000–0xFFF4A7FF	reserved		
– 0xFFF4A800–0xFFF4ABFC	TDM6 Transmit Channel Parameters Register 0–255	TDM6TCPR[0–255]	**page 19-63**
– 0xFFF4AC00–0xFFF4BEFF	reserved		
– 0xFFF4BF00	TDM6 Parity Control Register	TDM6PCR	**page 19-57**
– 0xFFF4BF04–0xFFF4BF07	reserved		
– 0xFFF4BF08	TDM6 Parity Error Register	TDM6PER	**page 19-73**
– 0xFFF4BF0C–0xFFF4BF0F	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF4BF10	TDM6 Transmit Force Register	TDM6TFR	**page 19-55**
– 0xFFF4BF14–0xFFF4BF17	reserved		
– 0xFFF4BF18	TDM6 Receive Force Register	TDM6RFR	**page 19-56**
– 0xFFF4BF1C–0xFFF4BF1F	reserved		
– 0xFFF4BF20	TDM6 Transmit Status Register	TDM6TSR	**page 19-73**
– 0xFFF4BF24–0xFFF4BF27	reserved		
– 0xFFF4BF28	TDM6 Receive Status Register	TDM6RSR	**page 19-72**
– 0xFFF4BF2C–0xFFF4BF2F	reserved		
– 0xFFF4BF30	TDM6 Adaptation Status Register	TDM6ASR	**page 19-71**
– 0xFFF4BF34–0xFFF4BF37	reserved		
– 0xFFF4BF38	TDM6 Transmit Event Register	TDM6TER	**page 19-70**
– 0xFFF4BF3C–0xFFF4BF3F	reserved		
– 0xFFF4BF40	TDM6 Receive Event Register	TDM6RER	**page 19-69**
– 0xFFF4BF44–0xFFF4BF47	reserved		
– 0xFFF4BF48	TDM6 Transmit Number of Buffers	TDM6TNB	**page 19-69**
– 0xFFF4BF4C–0xFFF4BF4F	reserved		
– 0xFFF4BF50	TDM6 Receive Number of Buffers	TDM6RNB	**page 19-68**
– 0xFFF4BF54–0xFFF4BF57	reserved		
– 0xFFF4BF58	TDM6 Transmit Data Buffer Displacement Register	TDM6TDBDR	**page 19-67**
– 0xFFF4BF5C–0xFFF4BF5F	reserved		
– 0xFFF4BF60	TDM6 Receive Data Buffer Displacement Register	TDM6RDBDR	**page 19-67**
– 0xFFF4BF64–0xFFF4BF67	reserved		
– 0xFFF4BF68	TDM6 Adaptation Sync Distance Register	TDM6ASDR	**page 19-66**
– 0xFFF4BF6C–0xFFF4BF6F	reserved		
– 0xFFF4BF70	TDM6 Transmit Interrupt Enable Register	TDM60TIER	**page 19-65**
– 0xFFF4BF74–0xFFF4BF77	reserved		
– 0xFFF4BF78	TDM6 Receive Interrupt Enable Register	TDM6RIER	**page 19-64**
– 0xFFF4BF7C–0xFFF4BF7F	reserved		
– 0xFFF4BF80	TDM6 Transmit Data Buffer Second Threshold	TDM6TDBST	**page 19-61**
– 0xFFF4BF84–0xFFF4BF87	reserved		
– 0xFFF4BF88	TDM6 Receive Data Buffer Second Threshold	TDM6RDBST	**page 19-61**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF4BF8C–0xFFF4BF8F	reserved		
– 0xFFF4BF90	TDM6 Transmit Data Buffer First Threshold	TDM6TDBFT	**page 19-60**
– 0xFFF4BF94–0xFFF4BF97	reserved		
– 0xFFF4BF98	TDM6 Receive Data Buffer First Threshold	TDM6RDBFT	**page 19-59**
– 0xFFF4BF9C–0xFFF4BF9F	reserved		
– 0xFFF4BFA0	TDM6 Transmit Control Register	TDM6TCR	**page 19-59**
– 0xFFF4BFA4–0xFFF4BFA7	reserved		
– 0xFFF4BFA8	TDM6 Receive Control Register	TDM6RCR	**page 19-58**
– 0xFFF4BFAC–0xFFF4BFAF	reserved		
– 0xFFF4BFB0	TDM6 Adaptation Control Register	TDM6ACR	**page 19-57**
– 0xFFF4BFB4–0xFFF4BFB7	reserved		
– 0xFFF4BFB8	TDM6 Transmit Global Base Address	TDM6TGBA	**page 19-55**
– 0xFFF4BFBC–0xFFF4BFBF	reserved		
– 0xFFF4BFC0	TDM6 Receive Global Base Address	TDM6RGBA	**page 19-54**
– 0xFFF4BFC4–0xFFF4BFC7	reserved		
– 0xFFF4BFC8	TDM6 Transmit Data Buffer Size	TDM6TDBS	**page 19-54**
– 0xFFF4BFCC–0xFFF4BFCF	reserved		
– 0xFFF4BFD0	TDM6 Receive Data Buffer Size	TDM6RDBS	**page 19-53**
– 0xFFF4BFD4–0xFFF4BFD7	reserved		
– 0xFFF4BFD8	TDM6 Transmit Frame Parameters	TDM6TFP	**page 19-51**
– 0xFFF4BFDC–0xFFF4BFDF	reserved		
– 0xFFF4BFE0	TDM6 Receive Frame Parameters	TDM6RFP	**page 19-48**
– 0xFFF4BFE4–0xFFF4BFE7	reserved		
– 0xFFF4BFE8	TDM6 Transmit Interface Register	TDM6TIR	**page 19-46**
– 0xFFF4BFEC–0xFFF4BFEF	reserved		
– 0xFFF4BFF0	TDM6 Receive Interface Register	TDM6RIR	**page 19-44**
– 0xFFF4BFF4–0xFFF4BFF7	reserved		
– 0xFFF4BFF8	TDM6 General Interface Register	TDM6GIR	**page 19-36**
– 0xFFF4BFFC–0xFFF4BFFF	reserved		
• 0xFFF4C000–0xFFF4FFFF	TDM7		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF4C000–0xFFF4C7FF	TDM7 Receive Local Memory		
– 0xFFF4C800–0xFFF4CFFF	reserved		
– 0xFFF4D000–0xFFF4D3FC	TDM7 Receive Channel Parameters Register 0–255	TDM7RCPR[0–255]	**page 19-62**
– 0xFFF4D400–0xFFF4D7FF	reserved		
– 0xFFF4D800–0xFFF4DFFF	TDM7 Transmit Local Memory		
– 0xFFF4E000–0xFFF4E7FF	reserved		
– 0xFFF4E800–0xFFF4EBFC	TDM7 Transmit Channel Parameters Register 0–255	TDM7TCPR[0–255]	**page 19-63**
– 0xFFF4EC00–0xFFF4FEFF	reserved		
– 0xFFF4FF00	TDM7 Parity Control Register	TDM7PCR	**page 19-57**
– 0xFFF4FF04–0xFFF4FF07	reserved		
– 0xFFF4FF08	TDM7 Parity Error Register	TDM7PER	**page 19-73**
– 0xFFF4FF0C–0xFFF4FF0F	reserved		
– 0xFFF4FF10	TDM7 Transmit Force Register	TDM7TFR	**page 19-55**
– 0xFFF4FF14–0xFFF4FF17	reserved		
– 0xFFF4FF18	TDM7 Receive Force Register	TDM7RFR	**page 19-56**
– 0xFFF4FF1C–0xFFF4FF1F	reserved		
– 0xFFF4FF20	TDM7 Transmit Status Register	TDM7TSR	**page 19-73**
– 0xFFF4FF24–0xFFF4FF27	reserved		
– 0xFFF4FF28	TDM7 Receive Status Register	TDM7RSR	**page 19-72**
– 0xFFF4FF2C–0xFFF4FF2F	reserved		
– 0xFFF4FF30	TDM7 Adaptation Status Register	TDM7ASR	**page 19-71**
– 0xFFF4FF34–0xFFF4FF37	reserved		
– 0xFFF4FF38	TDM7 Transmit Event Register	TDM7TER	**page 19-70**
– 0xFFF4FF3C–0xFFF4FF3F	reserved		
– 0xFFF4FF40	TDM7 Receive Event Register	TDM7RER	**page 19-69**
– 0xFFF4FF44–0xFFF4FF47	reserved		
– 0xFFF4FF48	TDM7 Transmit Number of Buffers	TDM7TNB	**page 19-69**
– 0xFFF4FF4C–0xFFF4FF4F	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF4FF50	TDM7 Receive Number of Buffers	TDM7RNB	**page 19-68**
– 0xFFF4FF54–0xFFF4FF57	reserved		
– 0xFFF4FF58	TDM7 Transmit Data Buffer Displacement Register	TDM7TDBDR	**page 19-67**
– 0xFFF4FF5C–0xFFF4FF5F	reserved		
– 0xFFF4FF60	TDM7 Receive Data Buffer Displacement Register	TDM7RDBDR	**page 19-67**
– 0xFFF4FF64–0xFFF4FF67	reserved		
– 0xFFF4FF68	TDM7 Adaptation Sync Distance Register	TDM7ASDR	**page 19-66**
– 0xFFF4FF6C–0xFFF4FF6F	reserved		
– 0xFFF4FF70	TDM7 Transmit Interrupt Enable Register	TDM70TIER	**page 19-65**
– 0xFFF4FF74–0xFFF4FF77	reserved		
– 0xFFF4FF78	TDM7 Receive Interrupt Enable Register	TDM7RIER	**page 19-64**
– 0xFFF4FF7C–0xFFF4FF7F	reserved		
– 0xFFF4FF80	TDM7 Transmit Data Buffer Second Threshold	TDM7TDBST	**page 19-61**
– 0xFFF4FF84–0xFFF4FF87	reserved		
– 0xFFF4FF88	TDM7 Receive Data Buffer Second Threshold	TDM7RDBST	**page 19-61**
– 0xFFF4FF8C–0xFFF4FF8F	reserved		
– 0xFFF4FF90	TDM7 Transmit Data Buffer First Threshold	TDM7TDBFT	**page 19-60**
– 0xFFF4FF94–0xFFF4FF97	reserved		
– 0xFFF4FF98	TDM7 Receive Data Buffer First Threshold	TDM7RDBFT	**page 19-59**
– 0xFFF4FF9C–0xFFF4FF9F	reserved		
– 0xFFF4FFA0	TDM7 Transmit Control Register	TDM7TCR	**page 19-59**
– 0xFFF4FFA4–0xFFF4FFA7	reserved		
– 0xFFF4FFA8	TDM7 Receive Control Register	TDM7RCR	**page 19-58**
– 0xFFF4FFAC–0xFFF4FFAF	reserved		
– 0xFFF4FFB0	TDM7 Adaptation Control Register	TDM7ACR	**page 19-57**
– 0xFFF4FFB4–0xFFF4FFB7	reserved		
– 0xFFF4FFB8	TDM7 Transmit Global Base Address	TDM7TGBA	**page 19-55**
– 0xFFF4FFBC–0xFFF4FFBF	reserved		
– 0xFFF4FFC0	TDM7 Receive Global Base Address	TDM7RGBA	**page 19-54**
– 0xFFF4FFC4–0xFFF4FFC7	reserved		
– 0xFFF4FFC8	TDM7 Transmit Data Buffer Size	TDM7TDBS	**page 19-54**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF4FFCC–0xFFF4FFCF	reserved		
– 0xFFF4FFD0	TDM7 Receive Data Buffer Size	TDM7RDBS	**page 19-53**
– 0xFFF4FFD4–0xFFF4FFD7	reserved		
– 0xFFF4FFD8	TDM7 Transmit Frame Parameters	TDM7TFP	**page 19-51**
– 0xFFF4FFDC–0xFFF4FFDF	reserved		
– 0xFFF4FFE0	TDM7 Receive Frame Parameters	TDM7RFP	**page 19-48**
– 0xFFF4FFE4–0xFFF4FFE7	reserved		
– 0xFFF4FFE8	TDM7 Transmit Interface Register	TDM7TIR	**page 19-46**
– 0xFFF4FFEC–0xFFF4FFEF	reserved		
– 0xFFF4FFF0	TDM7 Receive Interface Register	TDM7RIR	**page 19-44**
– 0xFFF4FFF4–0xFFF4FFF7	reserved		
– 0xFFF4FFF8	TDM7 General Interface Register	TDM7GIR	**page 19-36**
– 0xFFF4FFFC–0xFFF4FFFF	reserved		
• 0xFFF50000–0xFFF77FFF	reserved		
• 0xFFF78000–0xFFF7807F	General Configuration.		
– 0xFFF78000	General Configuration Register 1	GCR1	**page 8-2**
– 0xFFF78004	General Configuration Register 2	GCR2	**page 8-3**
– 0xFFF78008	General Status Register 1	GSR1	**page 8-4**
– 0xFFF7800C	Lynx General Configuration Register	L_GCR	**page 8-6**
– 0xFFF78010	DDR General Control Register	DDR_GCR	**page 8-7**
– 0xFFF78014	RapidIO Control Register	RIO_CR	**page 8-8**
– 0xFFF78018	SGMII Control Register	SGMII_CR	**page 8-9**
– 0xFFF7801C	QUICC Engine Control Register	CECTLR	**page 8-10**
– 0xFFF78020	reserved		
– 0xFFF78024	GPIO Input Enable Register	GIER	**page 8-11**
– 0xFFF78028	System Part and Revision ID Register	SPRIDR	**page 8-12**
– 0xFFF78030	General Configuration Register 4	GCR4	**page 8-13**
– 0xFFF78034–0xFFF7803F	reserved		
– 0xFFF78040	General Interrupt Register 1	GIR1	**page 8-14**
– 0xFFF78044	General Interrupt Enable Register 1 for Core 0	GIER1_0	**page 8-16**
– 0xFFF78048	General Interrupt Enable Register 1 for Core 1	GIER1_1	**page 8-16**
– 0xFFF7804C	General Interrupt Enable Register 1 for Core 2	GIER1_2	**page 8-16**
– 0xFFF78050	General Interrupt Enable Register 1 for Core 3	GIER1_3	**page 8-16**
– 0xFFF78054	General Interrupt Register 2	GIR2	**page 8-17**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF78058	General Interrupt Enable Register 2 for Core 0	GIER2_0	**page 8-19**
– 0xFFF7805C	General Interrupt Enable Register 2 for Core 1	GIER2_1	**page 8-19**
– 0xFFF78060	General Interrupt Enable Register 2 for Core 2	GIER2_2	**page 8-19**
– 0xFFF78064	General Interrupt Enable Register 2 for Core 3	GIER2_3	**page 8-19**
– 0xFFF78068	General Interrupt Register 3	GIR3	**page 8-21**
– 0xFFF7806C	General Interrupt Enable Register 3 for Core 0	GIER3_0	**page 8-22**
– 0xFFF78070	General Interrupt Enable Register 3 for Core 1	GIER3_1	**page 8-22**
– 0xFFF78074	General Interrupt Enable Register 3 for Core 2	GIER3_2	**page 8-22**
– 0xFFF78078	General Interrupt Enable Register 3 for Core 3	GIER3_3	**page 8-22**
– 0xFFF7807C–0xFFF7807F	reserved		
• 0xFFF78080–0xFFF79FFF	reserved		
• 0xFFF7A000–0xFFF7A1FF	PCI		
– 0xFFF7A000	PCI Error Status Register	PCI_ESR	**page 15-34**
– 0xFFF7A004	PCI Error Capture Disable Register	PCI_ECDR	**page 15-35**
– 0xFFF7A008	PCI Error Enable Register	PCI_EER	**page 15-36**
– 0xFFF7A00C	PCI Error Attributes Capture Register	PCI_EATCR	**page 15-37**
– 0xFFF7A010	PCI Error Address Capture Register	PCI_EACR	**page 15-39**
– 0xFFF7A014	PCI Error Extended Address Capture Register	PCI_EEACR	**page 15-39**
– 0xFFF7A018	PCI Error Data Low Capture Register	PCI_EDCR	**page 15-40**
– 0xFFF7A020–0xFFF7A037	reserved		
– 0xFFF7A038	PCI Inbound Translation Address Register 2	PITAR2	**page 15-40**
– 0xFFF7A03C–0xFFF7A03F	reserved		
– 0xFFF7A040	PCI Inbound Base Address Register 2	PIBAR2	**page 15-41**
– 0xFFF7A044	PCI Inbound Extended Base Address Register 2	PIEBAR2	**page 15-41**
– 0xFFF7A048	PCI Inbound Window Attribute Register 2	PIWAR2	**page 15-42**
– 0xFFF7A050	PCI Inbound Translation Address Register 1	PITAR1	**page 15-40**
– 0xFFF7A054–0xFFF7A057	reserved		
– 0xFFF7A058	PCI Inbound Base Address Register 1	PIBAR1	**page 15-41**
– 0xFFF7A05C	PCI Inbound Extended Base Address Register 1	PIEBAR1	**page 15-41**
– 0xFFF7A060	PCI Inbound Window Attributes Register 1	PIWAR1	**page 15-42**
– 0xFFF7A068	PCI Inbound Translation Address Register 0	PITAR0	**page 15-40**
– 0xFFF7A06C–0xFFF7A06F	reserved		
– 0xFFF7A070	PCI Inbound Base Address Register 0	PIBAR0	**page 15-41**
– 0xFFF7A078	PCI Inbound Window Attribute Register 0	PIWAR0	**page 15-41**
– 0xFFF7A07C–0xFFF7A0FF	reserved		
– 0xFFF7A100	PCI Outbound Translation Address Register 0	POTAR0	**page 15-43**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF7A104–0xFFF7A107	reserved		
– 0xFFF7A108	PCI Outbound Base Address Register 0	POBAR0	**page 15-44**
– 0xFFF7A10C–0xFFF7A10F	reserved		
– 0xFFF7A110	PCI Outbound Comparison Mask Register 0	POCMR0	**page 15-44**
– 0xFFF7A114–0xFFF7A117	reserved		
– 0xFFF7A118	PCI Outbound Translation Address Register 1	POTAR1	**page 15-43**
– 0xFFF7A11C–0xFFF7A11F	reserved		
– 0xFFF7A120	PCI Outbound Base Address Register 1	POBAR1	**page 15-44**
– 0xFFF7A124–0xFFF7A127	reserved		
– 0xFFF7A128	PCI Outbound Comparison Mask Register 1	POCMR1	**page 15-44**
– 0xFFF7A12C–0xFFF7A12F	reserved		
– 0xFFF7A130	PCI Outbound Translation Address Register 2	POTAR2	**page 15-43**
– 0xFFF7A134–0xFFF7A137	reserved		
– 0xFFF7A138	PCI Outbound Base Address Register 2	POBAR2	**page 15-44**
– 0xFFF7A13C–0xFFF7A13F	reserved		
– 0xFFF7A140	PCI Outbound Comparison Mask Register 2	POCMR2	**page 15-44**
– 0xFFF7A144–0xFFF7A147	reserved		
– 0xFFF7A148	PCI Outbound Translation Address Register 3	POTAR3	**page 15-43**
– 0xFFF7A14C–0xFFF7A14F	reserved		
– 0xFFF7A150	PCI Outbound Base Address Register 3	POBAR3	**page 15-44**
– 0xFFF7A154–0xFFF7A157	reserved		
– 0xFFF7A158	PCI Outbound Comparison Mask Register 3	POCMR3	**page 15-44**
– 0xFFF7A15C–0xFFF7A15F	reserved		
– 0xFFF7A160	PCI Outbound Translation Address Register 4	POTAR4	**page 15-43**
– 0xFFF7A164–0xFFF7A167	reserved		
– 0xFFF7A168	PCI Outbound Base Address Register 4	POBAR4	**page 15-44**
– 0xFFF7A16C–0xFFF7A16F	reserved		
– 0xFFF7A170	PCI Outbound Comparison Mask Register 4	POCMR4	**page 15-44**
– 0xFFF7A174–0xFFF7A177	reserved		
– 0xFFF7A178	PCI Outbound Translation Address Register 5	POTAR5	**page 15-43**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF7A17C–0xFFF7A17F	reserved		
– 0xFFF7A180	PCI Outbound Base Address Register 5	POBAR5	**page 15-44**
– 0xFFF7A184–0xFFF7A187	reserved		
– 0xFFF7A188	PCI Outbound Comparison Mask Register 5	POCMR5	**page 15-44**
– 0xFFF7A18C–0xFFF7A1F7	reserved		
– 0xFFF7A1F8	Discard Timer Control Register	DTCR	**page 15-46**
– 0xFFF7A1FC–0xFFF7A1FF	reserved		
• 0xFFF7A200–0xFFF7EFFF	reserved		
• 0xFFF7F000–0xFFF7F03F	UART		
– 0xFFF7F000	SCI Baud-Rate Register	SCIBR	**page 20-25**
– 0xFFF7F004–0xFFF7F007	reserved		
– 0xFFF7F008	SCI Control Register	SCICR	**page 20-26**
– 0xFFF7F00C–0xFFF7F00F	reserved		
– 0xFFF7F010	SCI Status Register	SCISR	**page 20-29**
– 0xFFF7F014–0xFFF7F017	reserved		
– 0xFFF7F018	SCI Data Register	SCIDR	**page 20-31**
– 0xFFF7F01C–0xFFF7F027	reserved		
– 0xFFF7F028	SCI Data Direction Register	SCIDDR	**page 20-32**
– 0xFFF7F02C–0xFFF7F03F	reserved		
• 0xFFF7F040–0xFFF7FFFF	reserved		
• 0xFFF80000–0xFFF9FFFF	RapidIO		
– 0xFFF80000	Device Identity Capability Register	DIDCAR	**page 16-104**
– 0xFFF80004	Device Information Capability Register	DICAR	**page 16-105**
– 0xFFF80008	Assembly Identity Capability Register	AIDCAR	**page 16-105**
– 0xFFF8000C	Assembly Information Capability Register	AICAR	**page 16-106**
– 0xFFF80010	Processing Element Features Capability Register	PEFCAR	**page 16-106**
– 0xFFF80018	Source Operations Capability Register	SOCAR	**page 16-108**
– 0xFFF8001C	Destination Operations Capability Register	DOCAR	**page 16-109**
– 0xFFF80020–0xFFF8003F	reserved		
– 0xFFF80040	Mailbox Command And Status Register	MCSR	**page 16-111**
– 0xFFF80044	Port -Write and Doorbell Command and Status Register	PWDCSR	**page 16-112**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF80048–0xFFF8004B	reserved		
– 0xFFF8004C	Processing Element Logical Layer Control Command and Status Register	PELLCCSR	**page 16-114**
– 0xFFF80050–0xFFF8005B	reserved		
– 0xFFF8005C	Local Configuration Space Base Address 1 Command and Status Register	LCSBA1SCR	**page 16-115**
– 0xFFF80060	Base Device ID Command and Status Register	BDIDCSR	**page 16-116**
– 0xFFF80064–0xFFF80067	reserved		
– 0xFFF80068	Host Base Device ID Lock Command and Status Register	HBDIDLCSR	**page 16-117**
– 0xFFF8006C	Component Tag Command and Status Register	CTCSR	**page 16-118**
– 0xFFF80070–0xFFF800FF	reserved		
– 0xFFF80100	Port Maintenance Block Header 0	PMBH0	**page 16-119**
– 0xFFF80104–0xFFF8011F	reserved		
– 0xFFF80120	Port Link Time-out Control Command and Status Register	PLTOCCSR	**page 16-120**
– 0xFFF80124	Port Response Time-out Control Command and Status Register	PRTOCCSR	**page 16-121**
– 0xFFF80128–0xFFF8013B	reserved		
– 0xFFF8013C	Port General Control Command and Status Register	PGCCSR	**page 16-122**
– 0xFFF80140–0xFFF80147	reserved		
– 0xFFF80148	Port 0 Local ackID Status Command and Status Register	P0LASCSR	**page 16-125**
– 0xFFF8014C–0xFFF80157	reserved		
– 0xFFF80158	Error and Status Command and Status Register	ESCSR	**page 16-126**
– 0xFFF8015C	Port 0 Control Command and Status Register	P0CCSR	**page 16-128**
– 0xFFF80160–0xFFF805FF	reserved		
– 0xFFF80600	Error Reporting Block Header	ERBH	**page 16-130**
– 0xFFF80604–0xFFF80607	reserved		
– 0xFFF80608	Logical/Transport Layer Error Detect Command and Status Register	LTLEDCSR	**page 16-130**
– 0xFFF8060C	Logical/Transport Layer Error Enable Command and Status Register	LTLEECSR	**page 16-132**
– 0xFFF80610–0xFFF80613	reserved		
– 0xFFF80614	Logical/Transport Layer Address Capture Command and Status Register	LTLACCSR	**page 16-133**
– 0xFFF80618	Logical/Transport Layer Device ID Capture Command and Status Register	LTLDIDCCSR	**page 16-134**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF8061C	Logical/Transport Layer Control Capture Command and Status Register	LTLCCCSR	**page 16-135**
– 0xFFF80620–0xFFF8063F	reserved		
– 0xFFF80640	Port 0 Error Detect Command and Status Register	P0EDCSR	**page 16-136**
– 0xFFF80644	Port 0 Error Rate Enable Command and Status Register	P0ERECSR	**page 16-137**
– 0xFFF80648	Port 0 Error Capture Attributes Command and Status Register	P0ECACSR	**page 16-138**
– 0xFFF8064C	Port 0 Packet/Control Symbol Error Capture Command and Status Register 0	P0PCSECCSR0	**page 16-140**
– 0xFFF80650	Port 0 Packet Error Capture Command and Status Register 1	P0PECCSR1	**page 16-141**
– 0xFFF80654	Port 0 Packet Error Capture Command and Status Register 2	P0PECCSR2	**page 16-142**
– 0xFFF80658	Port 0 Packet Error Capture Command and Status Register 3	P0PECCSR3	**page 16-143**
– 0xFFF8065C–0xFFF80667	reserved		
– 0xFFF80668	Port 0 Error Rate Command and Status Register	P0ERCSR	**page 16-144**
– 0xFFF8066C	Port 0 Error Rate Threshold Command and Status Register	P0ERTCSR	**page 16-145**
– 0xFFF80670–0xFFF90003	reserved		
– 0xFFF90004	Logical Layer Configuration Register	LLCR	**page 16-146**
– 0xFFF90008–0xFFF9000F	reserved		
– 0xFFF90010	Error/Port-Write Interrupt Status Register	EPWISR	**page 16-147**
– 0xFFF90014–0xFFF9001F	reserved		
– 0xFFF90020	Logical Retry Error Threshold Configuration Register	LRETCR	**page 16-148**
– 0xFFF90024–0xFFF9007F	reserved		
– 0xFFF90080	Physical Retry Error Threshold Configuration Register	PRETCR	**page 16-149**
– 0xFFF90084–0xFFF900FF	reserved		
– 0xFFF90100	Port 0 Alternate Device ID Command and Status Register	P0ADIDCSR	**page 16-150**
– 0xFFF90104–0xFFF9011F	reserved		
– 0xFFF90120	Port 0 Accept-All Configuration Register	P0AACR	**page 16-151**
– 0xFFF90124	Port 0 Logical Outbound Packet Time-to-Live Configuration Register	P0LOPTTLCR	**page 16-152**
– 0xFFF90128–0xFFF9012F	reserved		
– 0xFFF90130	Port 0 Implementation Error Command and Status Register	P0IECSR	**page 16-153**
– 0xFFF90134–0xFFF9013F	reserved		
– 0xFFF90140	Port 0 Physical Configuration Register	P0PCR	**page 16-154**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF90144–0xFFF90157	reserved		
– 0xFFF90158	Port 0 Serial Link Command And Status Register	P0SLCSR	**page 16-155**
– 0xFFF9015C–0xFFF9015F	reserved		
– 0xFFF90160	Port 0 Serial Link Error Injection Configuration Register	P0SLEICR	**page 16-156**
– 0xFFF90164–0xFFF90BF7	reserved		
– 0xFFF90BF8	IP Block Revision Register 1	IPBRR1	**page 16-157**
– 0xFFF90BFC	IP Block Revision Register 2	IPBRR2	**page 16-157**
– 0xFFF90C00	Port 0 RapidIO Outbound Window Translation Address Register 0	P0ROWTAR0	**page 16-158**
– 0xFFF90C04	Port 0 RapidIO Outbound Window Translation Extended Address Register 0	P0ROWTEAR0	**page 16-159**
– 0xFFF90C08	Port 0 RapidIO Outbound Window Base Address Register 0	P0ROWBAR0	**page 16-162**
– 0xFFF90C0C–0xFFF90C0F	reserved		
– 0xFFF90C10	Port 0 RapidIO Outbound Window Attributes Register 0	P0ROWAR0	**page 16-160**
– 0xFFF90C14–0xFFF90C1F	reserved		
– 0xFFF90C20	Port 0 RapidIO Outbound Window Translation Address Register 1	P0ROWTAR1	**page 16-158**
– 0xFFF90C24	Port 0 RapidIO Outbound Window Translation Extended Address Register 1	P0ROWTEAR1	**page 16-159**
– 0xFFF90C28	Port 0 RapidIO Outbound Window Base Address Register 1	P0ROWBAR1	**page 16-162**
– 0xFFF90C2C–0xFFF90C2F	reserved		
– 0xFFF90C30	Port 0 RapidIO Outbound Window Attributes Register 1	P0ROWAR1	**page 16-160**
– 0xFFF90C34–0xFFF90C3F	reserved		
– 0xFFF90C40	Port 0 RapidIO Outbound Window Translation Address Register 2	P0ROWTAR2	**page 16-158**
– 0xFFF90C44	Port 0 RapidIO Outbound Window Translation Extended Address Register 2	P0ROWTEAR2	**page 16-159**
– 0xFFF90C48	Port 0 RapidIO Outbound Window Base Address Register 2	P0ROWBAR2	**page 16-162**
– 0xFFF90C4C–0xFFF90C4F	reserved		
– 0xFFF90C50	Port 0 RapidIO Outbound Window Attributes Register 2	P0ROWAR2	**page 16-160**
– 0xFFF90C54–0xFFF90C5F	reserved		
– 0xFFF90C60	Port 0 RapidIO Outbound Window Translation Address Register 3	P0ROWTAR3	**page 16-158**
– 0xFFF90C64	Port 0 RapidIO Outbound Window Translation Extended Address Register 3	P0ROWTEAR3	**page 16-159**
– 0xFFF90C68	Port 0 RapidIO Outbound Window Base Address Register 3	P0ROWBAR3	**page 16-162**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF90C6C–0xFFF90C6F	reserved		
– 0xFFF90C70	Port 0 RapidIO Outbound Window Attributes Register 3	P0ROWAR3	**page 16-160**
– 0xFFF90C74–0xFFF90C7F	reserved		
– 0xFFF90C80	Port 0 RapidIO Outbound Window Translation Address Register 4	P0ROWTAR4	**page 16-158**
– 0xFFF90C84	Port 0 RapidIO Outbound Window Translation Extended Address Register 4	P0ROWTEAR4	**page 16-159**
– 0xFFF90C88	Port 0 RapidIO Outbound Window Base Address Register 4	P0ROWBAR4	**page 16-162**
– 0xFFF90C8C–0xFFF90C8F	reserved		
– 0xFFF90C90	Port 0 RapidIO Outbound Window Attributes Register 4	P0ROWAR4	**page 16-160**
– 0xFFF90C94–0xFFF90C9F	reserved		
– 0xFFF90CA0	Port 0 RapidIO Outbound Window Translation Address Register 5	P0ROWTAR5	**page 16-158**
– 0xFFF90CA4	Port 0 RapidIO Outbound Window Translation Extended Address Register 5	P0ROWTEAR5	**page 16-159**
– 0xFFF90CA8	Port 0 RapidIO Outbound Window Base Address Register 5	P0ROWBAR5	**page 16-162**
– 0xFFF90CAC–0xFFF90CAF	reserved		
– 0xFFF90CB0	Port 0 RapidIO Outbound Window Attributes Register 5	P0ROWAR5	**page 16-160**
– 0xFFF90CB4–0xFFF90CBF	reserved		
– 0xFFF90CC0	Port 0 RapidIO Outbound Window Translation Address Register 6	P0ROWTAR6	**page 16-158**
– 0xFFF90CC4	Port 0 RapidIO Outbound Window Translation Extended Address Register 6	P0ROWTEAR6	**page 16-159**
– 0xFFF90CC8	Port 0 RapidIO Outbound Window Base Address Register 6	P0ROWBAR6	**page 16-162**
– 0xFFF90CCC–0xFFF90CCF	reserved		
– 0xFFF90CD0	Port 0 RapidIO Outbound Window Attributes Register 6	P0ROWAR6	**page 16-160**
– 0xFFF90CD4–0xFFF90CDF	reserved		
– 0xFFF90CE0	Port 0 RapidIO Outbound Window Translation Address Register 7	P0ROWTAR7	**page 16-158**
– 0xFFF90CE4	Port 0 RapidIO Outbound Window Translation Extended Address Register 7	P0ROWTEAR7	**page 16-159**
– 0xFFF90CE8	Port 0 RapidIO Outbound Window Base Address Register 7	P0ROWBAR7	**page 16-162**
– 0xFFF90CEC–0xFFF90CEF	reserved		
– 0xFFF90CF0	Port 0 RapidIO Outbound Window Attributes Register 7	P0ROWAR7	**page 16-160**
– 0xFFF90CF4–0xFFF90CFF	reserved		
– 0xFFF90D00	Port 0 RapidIO Outbound Window Translation Address Register 8	P0ROWTAR8	**page 16-158**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF90D04	Port 0 RapidIO Outbound Window Translation Extended Address Register 8	P0ROWTEAR8	**page 16-159**
– 0xFFF90D08	Port 0 RapidIO Outbound Window Base Address Register 8	P0ROWBAR8	**page 16-162**
– 0xFFF90D0C–0xFFF90D0F	reserved		
– 0xFFF90D10	Port 0 RapidIO Outbound Window Attributes Register 8	P0ROWAR8	**page 16-160**
– 0xFFF90D14–0xFFF90D5F	reserved		
– 0xFFF90D60	RapidIO Inbound Window Translation Address Register 4	RIWTAR4	**page 16-164**
– 0xFFF90D64–0xFFF90D67	reserved		
– 0xFFF90D68	RapidIO Inbound Window Base Address Register 4	RIWBAR4	**page 16-165**
– 0xFFF90D6C–0xFFF90D6F	reserved		
– 0xFFF90D70	RapidIO Inbound Window Attributes Register 4	RIWAR4	**page 16-166**
– 0xFFF90D74–0xFFF90D7F	reserved		
– 0xFFF90D80	RapidIO Inbound Window Translation Address Register 3	RIWTAR3	**page 16-164**
– 0xFFF90D84–0xFFF90D87	reserved		
– 0xFFF90D88	RapidIO Inbound Window Base Address Register 3	RIWBAR3	**page 16-165**
– 0xFFF90D8C–0xFFF90D8F	reserved		
– 0xFFF90D90	RapidIO Inbound Window Attributes Register 3	RIWAR3	**page 16-166**
– 0xFFF90D94–0xFFF90D9F	reserved		
– 0xFFF90DA0	RapidIO Inbound Window Translation Address Register 2	RIWTAR2	**page 16-164**
– 0xFFF90DA4–0xFFF90DA7	reserved		
– 0xFFF90DA8	RapidIO Inbound Window Base Address Register 2	RIWBAR2	**page 16-165**
– 0xFFF90DAC–0xFFF90DAF	reserved		
– 0xFFF90DB0	RapidIO inbound window attributes register 2	RIWAR2	**page 16-166**
– 0xFFF90DB4–0xFFF90DBF	reserved		
– 0xFFF90DC0	RapidIO Inbound Window Translation Address Register 1	RIWTAR1	**page 16-164**
– 0xFFF90DC4–0xFFF90DC7	reserved		
– 0xFFF90DC8	RapidIO Inbound Window Base Address Register 1	RIWBAR1	**page 16-165**
– 0xFFF90DCC–0xFFF90DCF	reserved		
– 0xFFF90DD0	RapidIO Inbound Window Attributes Register 1	RIWAR1	**page 16-166**
– 0xFFF90DD4–0xFFF90DDF	reserved		
– 0xFFF90DE0	RapidIO Inbound Window Translation Address Register 0	RIWTAR0	**page 16-164**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF90DE4–0xFFF90DEF	reserved		
– 0xFFF90DF0	RapidIO Inbound Window Attributes Register 0	RIWAR0	**page 16-166**
– 0xFFF90DF4–0xFFF92FFF	reserved		
– 0xFFF93000	Outbound Message 0 Mode Register	OM0MR	**page 16-167**
– 0xFFF93004	Outbound Message 0 Status Register	OM0SR	**page 16-169**
– 0xFFF93008–0xFFF9300B	reserved		
– 0xFFF9300C	Outbound Message 0 Descriptor Queue Dequeue Pointer Address Register	OM0DQDPAR	**page 16-171**
– 0xFFF93010–0xFFF93013	reserved		
– 0xFFF93014	Outbound Message 0 Source Address Register	OM0SAR	**page 16-172**
– 0xFFF93018	Outbound Message 0 Destination Port Register	OM0DPR	**page 16-173**
– 0xFFF9301C	Outbound Message 0 Destination Attributes Register	OM0DATR	**page 16-174**
– 0xFFF93020	Outbound Message 0 Double-word Count Register	OM0DCR	**page 16-175**
– 0xFFF93024–0xFFF93027	reserved		
– 0xFFF93028	Outbound Message 0 Descriptor Queue Enqueue Pointer Address Register	OM0DQEPAR	**page 16-176**
– 0xFFF9302C	Outbound Message 0 Retry Error Threshold Configuration Register	OM0RETCR	**page 16-177**
– 0xFFF93030	Outbound Message 0 Multicast Group Register	OM0MGR	**page 16-178**
– 0xFFF93034	Outbound Message 0 Multicast List Register	OM0MLR	**page 16-179**
– 0xFFF93038–0xFFF9305F	reserved		
– 0xFFF93060	Inbound Message 0 Mode Register	IM0MR	**page 16-180**
– 0xFFF93064	Inbound Message 0 Status Register	IM0SR	**page 16-182**
– 0xFFF93068–0xFFF9306B	reserved		
– 0xFFF9306C	Inbound Message 0 Frame Queue Dequeue Pointer Address Register	IM0FQDPAR	**page 16-184**
– 0xFFF93070–0xFFF93073	reserved		
– 0xFFF93074	Inbound Message 0 Frame Queue Enqueue Pointer Address Register	IM0FQEPAR	**page 16-185**
– 0xFFF93078	Inbound Message 0 Maximum Interrupt Report Interval Register	IM0MIRIR	**page 16-186**
– 0xFFF9307C–0xFFF930FF	reserved		
– 0xFFF93100	Outbound Message 1 Mode Register	OM1MR	**page 16-167**
– 0xFFF93104	Outbound Message 1 Status Register	OM1SR	**page 16-169**
– 0xFFF93108–0xFFF9310B	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF9310C	Outbound Message 1 Descriptor Queue Dequeue Pointer Address Register	OM1DQDPAR	**page 16-171**
– 0xFFF93110–0xFFF93113	reserved		
– 0xFFF93114	Outbound Message 1 Source Address Register	OM1SAR	**page 16-172**
– 0xFFF93118	Outbound Message 1 Destination Port Register	OM1DPR	**page 16-173**
– 0xFFF9311C	Outbound Message 1 Destination Attributes Register	OM1DATR	**page 16-174**
– 0xFFF93120	Outbound Message 1 Double-word Count Register	OM1DCR	**page 16-175**
– 0xFFF93124–0xFFF93127	reserved		
– 0xFFF93128	Outbound Message 1 Descriptor Queue Enqueue Pointer Address Register	OM1DQEPAR	**page 16-176**
– 0xFFF9312C	Outbound Message 1 Retry Error Threshold Configuration Register	OM1RETCR	**page 16-177**
– 0xFFF93130	Outbound Message 1 Multicast Group Register	OM1MOR	**page 16-178**
– 0xFFF93134	Outbound Message 1 Multicast List Register	OM1MLR	**page 16-179**
– 0xFFF93138–0xFFF9315F	reserved		
– 0xFFF93160	Inbound Message 1 Mode Register	IM1MR	**page 16-180**
– 0xFFF93164	Inbound Message 1 Status Register	IM1SR	**page 16-182**
– 0xFFF93168–0xFFF9316B	reserved		
– 0xFFF9316C	Inbound Message 1 Frame Queue Dequeue Pointer Address Register	IM1FQDPAR	**page 16-184**
– 0xFFF93170–0xFFF93173	reserved		
– 0xFFF93174	Inbound Message 1 Frame Queue Enqueue Pointer Address Register	IM1FQEPAR	**page 16-185**
– 0xFFF93178	Inbound Message 1 Maximum Interrupt Report Interval Register	IM1MIRIR	**page 16-186**
– 0xFFF9317C–0xFFF933FF	reserved		
– 0xFFF93400	Outbound Doorbell Mode Register	ODMR	**page 16-187**
– 0xFFF93404	Outbound Doorbell Status Register	ODSR	**page 16-188**
– 0xFFF93408–0xFFF93417	reserved		
– 0xFFF93418	Outbound Doorbell Destination Port Register	ODDPR	**page 16-189**
– 0xFFF9341C	Outbound Doorbell Destination Attributes Register	ODDATR	**page 16-190**
– 0xFFF93420–0xFFF9342B	reserved		
– 0xFFF9342C	Outbound Doorbell Retry Error Threshold Configuration Register	ODRETCR	**page 16-191**
– 0xFFF93430–0xFFF9345F	reserved		
– 0xFFF93460	Inbound Doorbell Mode Register	IDMR	**page 16-192**
– 0xFFF93464	Inbound Doorbell Status Register	IDSR	**page 16-194**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFF93468–0xFFF9346B	reserved		
– 0xFFF9346C	Inbound Doorbell Queue Dequeue Pointer Address Register	IDQDPAR	**page 16-195**
– 0xFFF93470–0xFFF93473	reserved		
– 0xFFF93474	Inbound Doorbell Queue Enqueue Pointer Address Register	IDQEPAR	**page 16-196**
– 0xFFF93478	Inbound Doorbell Maximum Interrupt Report Interval Register	IDMIRIR	**page 16-197**
– 0xFFF9347C–0xFFF934DF	reserved		
– 0xFFF934E0	Inbound Port-write Mode Register	IPWMR	**page 16-198**
– 0xFFF934E4	Inbound Port-write Status Register	IPWSR	**page 16-199**
– 0xFFF934EC	Inbound Port-write Queue Base Address Register	IPWQBAR	**page 16-200**
• 0xFFF934F0–0xFFF9FFFF	reserved		
• 0xFFFA0000–0xFFFA00FF	OCN Crossbar Switch		
• 0xFFFA0100–0xFFFA0FFF	reserved		
• 0xFFFA1000–0xFFFA103F	OCN Crossbar Switch to MBus		
• 0xFFFA1040–0xFFFA1FFF	reserved		
• 0xFFFA2000–0xFFFA3FFF	Dedicated DMA Controller for RapidIO Interface Registers		
– 0xFFFA2000–0xFFFA20FF	reserved		
– 0xFFFA2100	DMA 0 Mode Register	MR0	
– 0xFFFA2104	DMA 0 Status Register	SR0	
– 0xFFFA2108	DMA 0 Current Link Descriptor Extended Address Register	ECLNDAR0	
– 0xFFFA210C	DMA 0 Current Link Descriptor Address Register	CLNDAR0	
– 0xFFFA2110	DMA 0 Source Attributes Register	SATR0	
– 0xFFFA2114	DMA 0 Source Address Register	SAR0	
– 0xFFFA2118	DMA 0 Destination Attributes Register	DATR0	
– 0xFFFA211C	DMA 0 Destination Address Register	DAR0	
– 0xFFFA2120	DMA 0 Byte Count Register	BCR0	
– 0xFFFA2124	DMA 0 Next Link Descriptor Extended Address Register	ENLNDAR0	
– 0xFFFA2128	DMA 0 Next Link Descriptor Address Register	NLNDAR0	
– 0xFFFA212C–0xFFFA212F	reserved		
– 0xFFFA2130	DMA 0 Current List Descriptor Extended Address Register	ECLSDAR0	
– 0xFFFA2134	DMA 0 Current List Descriptor Address Register	CLSDAR0	
– 0xFFFA2138	DMA 0 Next List Descriptor Extended Address Register	ENLSDAR0	

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFFA213C	DMA 0 Next List Descriptor Address Register	NLSDAR0	
– 0xFFFA2140	DMA 0 Source Stride Register	SSR0	
– 0xFFFA2144	DMA 0 Destination Stride Register	DSR0	
– 0xFFFA2148–0xFFFA217F	reserved		
– 0xFFFA2180	DMA 1 Mode Register	MR1	
– 0xFFFA2184	DMA 1 Status Register	SR1	
– 0xFFFA2188	DMA 1 Current Link Descriptor Extended Address Register	ECLNDAR1	
– 0xFFFA218C	DMA 1 Current Link Descriptor Address Register	CLNDAR1	
– 0xFFFA2190	DMA 1 Source Attributes Register	SATR1	
– 0xFFFA2194	DMA 1 Source Address Register	SAR1	
– 0xFFFA2198	DMA 1 Destination Attributes Register	DATR1	
– 0xFFFA219C	DMA 1 Destination Address Register	DAR1	
– 0xFFFA21A0	DMA 1 Byte Count Register	BCR1	
– 0xFFFA21A4	DMA 1 Next Link Descriptor Extended Address Register	ENLNDAR1	
– 0xFFFA21A8	DMA 1 Next Link Descriptor Address Register	NLNDAR1	
– 0xFFFA21AC–0xFFFA21AF	reserved		
– 0xFFFA21B0	DMA 1 Current List Descriptor Extended Address Register	ECLSDAR1	
– 0xFFFA21B4	DMA 1 Current List Descriptor Address Register	CLSDAR1	
– 0xFFFA21B8	DMA 1 Next List Descriptor Extended Address Register	ENLSDAR1	
– 0xFFFA21BC	DMA 1 Next List Descriptor Address Register	NLSDAR1	
– 0xFFFA21C0	DMA 1 Source Stride Register	SSR1	
– 0xFFFA21C4	DMA 1 Destination Stride Register	DSR1	
– 0xFFFA21C8–0xFFFA21FF	reserved		
– 0xFFFA2200	DMA 2 Mode Register	MR2	
– 0xFFFA2204	DMA 2 Status Register	SR2	
– 0xFFFA2208	DMA 2 Current Link Descriptor Extended Address Register	ECLNDAR2	
– 0xFFFA220C	DMA 2 Current Link Descriptor Address Register	CLNDAR2	
– 0xFFFA2210	DMA 2 Source Attributes Register	SATR2	
– 0xFFFA2214	DMA 2 Source Address Register	SAR2	
– 0xFFFA2218	DMA 2 Destination Attributes Register	DATR2	
– 0xFFFA221C	DMA 2 Destination Address Register	DAR2	
– 0xFFFA2220	DMA 2 Byte Count Register	BCR2	
– 0xFFFA2224	DMA 2 Next Link Descriptor Extended Address Register	ENLNDAR2	
– 0xFFFA2228	DMA 2 Next Link Descriptor Address Register	NLNDAR2	
– 0xFFFA222C–0xFFFA222F	reserved		
– 0xFFFA2230	DMA 2 Current List Descriptor Extended Address Register	ECLSDAR2	
– 0xFFFA2234	DMA 2 Current List Descriptor Address Register	CLSDAR2	
– 0xFFFA2238	DMA 2 Next List Descriptor Extended Address Register	ENLSDAR2	
– 0xFFFA223C	DMA 2 Next List Descriptor Address Register	NLSDAR2	

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFFA2240	DMA 2 Source Stride Register	SSR2	
– 0xFFFA2244	DMA 2 Destination Stride Register	DSR2	
– 0xFFFA2248–0xFFFA227F	reserved		
– 0xFFFA2280	DMA 3 Mode Register	MR3	
– 0xFFFA2284	DMA 3 Status Register	SR3	
– 0xFFFA2288	DMA 3 Current Link Descriptor Extended Address Register	ECLNDAR3	
– 0xFFFA228C	DMA 3 Current Link Descriptor Address Register	CLNDAR3	
– 0xFFFA2290	DMA 3 Source Attributes Register	SATR3	
– 0xFFFA2294	DMA 3 Source Address Register	SAR3	
– 0xFFFA2298	DMA 3 Destination Attributes Register	DATR3	
– 0xFFFA229C	DMA 3 Destination Address Register	DAR3	
– 0xFFFA22A0	DMA 3 Byte Count Register	BCR3	
– 0xFFFA22A4	DMA 3 Next Link Descriptor Extended Address Register	ENLNDAR3	
– 0xFFFA22A8	DMA 3 Next Link Descriptor Address Register	NLNDAR3	
– 0xFFFA22AC–0xFFFA22AF	reserved		
– 0xFFFA22B0	DMA 3 Current List Descriptor Extended Address Register	ECLSDAR3	
– 0xFFFA22B4	DMA 3 Current List Descriptor Address Register	CLSDAR3	
– 0xFFFA22B8	DMA 3 Next List Descriptor Extended Address Register	ENLSDAR3	
– 0xFFFA22BC	DMA 3 Next List Descriptor Address Register	NLSDAR3	
– 0xFFFA22C0	DMA 3 Source Stride Register	SSR3	
– 0xFFFA22C4	DMA 3 Destination Stride Register	DSR3	
– 0xFFFA22C8–0xFFFA22FF	reserved		
– 0xFFFA2300	DMA General Status Register	DGSR	
– 0xFFFA2304–0xFFFA3C07	reserved		
– 0xFFFA3C08	Local Access Window Base Address Register 0	LAWBAR0	
– 0xFFFA3C0C–0xFFFA3C0F	reserved		
– 0xFFFA3C10	Local Access Window Attributes Register 0	LAWAR0	
– 0xFFFA3C14–0xFFFA3C27	reserved		
– 0xFFFA3C28	Local Access Window Base Address Register 1	LAWBAR1	
– 0xFFFA3C2C–0xFFFA3C2F	reserved		
– 0xFFFA3C30	Local Access Window Attributes Register 1	LAWAR1	
– 0xFFFA3C34–0xFFFA3C47	reserved		
– 0xFFFA3C48	Local Access Window Base Address Register 2	LAWBAR2	
– 0xFFFA3C4C–0xFFFA3C4F	reserved		
– 0xFFFA3C50	Local Access Window Attributes Register 2	LAWAR2	

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFFA3C54–0xFFFA3C67	reserved		
– 0xFFFA3C68	Local Access Window Base Address Register 3	LAWBAR3	
– 0xFFFA3C6C–0xFFFA3C6F	reserved		
– 0xFFFA3C70	Local Access Window Attributes Register 3	LAWAR3	
– 0xFFFA3C74–0xFFFA3C87	reserved		
– 0xFFFA3C88	Local Access Window Base Address Register 4	LAWBAR4	
– 0xFFFA3C8C–0xFFFA3C8F	reserved		
– 0xFFFA3C90	Local Access Window Attributes Register 4	LAWAR4	
– 0xFFFA3C94–0xFFFA3CA7	reserved		
– 0xFFFA3CA8	Local Access Window Base Address Register 5	LAWBAR5	
– 0xFFFA3CAC–0xFFFA3CAF	reserved		
– 0xFFFA3CB0	Local Access Window Attributes Register 5	LAWAR5	
– 0xFFFA3CB4–0xFFFA3CC7	reserved		
– 0xFFFA3CC8	Local Access Window Base Address Register 6	LAWBAR6	
– 0xFFFA3CCC–0xFFFA3CCF	reserved		
– 0xFFFA3CD0	Local Access Window Attributes Register 6	LAWAR6	
– 0xFFFA3CD4–0xFFFA3CE7	reserved		
– 0xFFFA3CE8	Local Access Window Base Address Register 7	LAWBAR7	
– 0xFFFA3CEC–0xFFFA3CEF	reserved		
– 0xFFFA3CF0	Local Access Window Attributes Register 7	LAWAR7	
– 0xFFFA3CF4–0xFFFA3D07	reserved		
– 0xFFFA3D08	Local Access Window Base Address Register 8	LAWBAR8	
– 0xFFFA3D0C–0xFFFA3D0F	reserved		
– 0xFFFA3D10	Local Access Window Attributes Register 8	LAWAR8	
– 0xFFFA3D14–0xFFFA3D27	reserved		
– 0xFFFA3D28	Local Access Window Base Address Register 9	LAWBAR9	
– 0xFFFA3D2C–0xFFFA3D2F	reserved		
– 0xFFFA3D30	Local Access Window Attributes Register 9	LAWAR9	
– 0xFFFA3D34–0xFFFA3FFF	reserved		
• 0xFFFA4000–0xFFFC00FF	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
• 0xFFFC0100–0xFFFC01FF	Performance Monitor		
– 0xFFFC0100	Performance Monitor Global Control Register	PMGCR	**page 25-80**
– 0xFFFC0104–0xFFFC010F	reserved		
– 0xFFFC0110	Performance Monitor Local Control Register A0	PMLCA0	**page 25-81**
– 0xFFFC0114	Performance Monitor Local Control Register B0	PMLCB0	**page 25-83**
– 0xFFFC0118	Performance Monitor Counter 0	PMC0	**page 25-85**
– 0xFFFC011C–0xFFFC011F	reserved		
– 0xFFFC0120	Performance Monitor Local Control Register A1	PMLCA1	**page 25-81**
– 0xFFFC0124	Performance Monitor Local Control Register B1	PMLCB1	**page 25-83**
– 0xFFFC0128	Performance Monitor Counter 1	PMC1	**page 25-85**
– 0xFFFC012C–0xFFFC012F	reserved		
– 0xFFFC0130	Performance Monitor Local Control Register A2	PMLCA2	**page 25-81**
– 0xFFFC0134	Performance Monitor Local Control Register B2	PMLCB2	**page 25-83**
– 0xFFFC0138	Performance Monitor Counter 2	PMC2	**page 25-85**
– 0xFFFC013C–0xFFFC013F	reserved		
– 0xFFFC0140	Performance Monitor Local Control Register A3	PMLCA3	**page 25-81**
– 0xFFFC0144	Performance Monitor Local Control Register B3	PMLCB3	**page 25-83**
– 0xFFFC0148	Performance Monitor Counter 3	PMC3	**page 25-85**
– 0xFFFC014C–0xFFFC014F	reserved		
– 0xFFFC0150	Performance Monitor Local Control Register A4	PMLCA4	**page 25-81**
– 0xFFFC0154	Performance Monitor Local Control Register B4	PMLCB4	**page 25-83**
– 0xFFFC0158	Performance Monitor Counter 4	PMC4	**page 25-85**
– 0xFFFC015C–0xFFFC015F	reserved		
– 0xFFFC0160	Performance Monitor Local Control Register A5	PMLCA5	**page 25-81**
– 0xFFFC0164	Performance Monitor Local Control Register B5	PMLCB5	**page 25-83**
– 0xFFFC0168	Performance Monitor Counter 5	PMC5	**page 25-85**
– 0xFFFC016C–0xFFFC016F	reserved		
– 0xFFFC0170	Performance Monitor Local Control Register A6	PMLCA6	**page 25-81**
– 0xFFFC0174	Performance Monitor Local Control Register B6	PMLCB6	**page 25-83**
– 0xFFFC0178	Performance Monitor Counter 6	PMC6	**page 25-85**
– 0xFFFC017C–0xFFFC017F	reserved		
– 0xFFFC0180	Performance Monitor Local Control Register A7	PMLCA7	**page 25-81**
– 0xFFFC0184	Performance Monitor Local Control Register B7	PMLCB7	**page 25-83**
– 0xFFFC0188	Performance Monitor Counter 7	PMC7	**page 25-85**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFFC018C–0xFFFC018F	reserved		
– 0xFFFC0190	Performance Monitor Local Control Register A8	PMLCA8	**page 25-81**
– 0xFFFC0194	Performance Monitor Local Control Register B8	PMLCB8	**page 25-83**
– 0xFFFC0198	Performance Monitor Counter 8	PMC8	**page 25-85**
– 0xFFFC019C–0xFFFC01FF	reserved		
• 0xFFFC0200–0xFFFCFFFF	reserved		
• 0xFFFD0000–0xFFFDEFFF	Security Engine		
– 0xFFFD0000–0xFFFD1007	reserved		
– 0xFFFD1008	Controller Interrupt Enable Register	CIER	**page 26-81**
– 0xFFFD1010	Controller Interrupt Status Register	CISR	**page 26-84**
– 0xFFFD1018	Controller Interrupt Clear Register	CICR	**page 26-87**
– 0xFFFD1020	Controller Identification Register	CIDR	**page 26-79**
– 0xFFFD1028	EU Assignment Status Register	EUASR	**page 26-80**
– 0xFFFD1030	Master Control Register	MCR	**page 26-77**
– 0xFFFD1038–0xFFFD1107	reserved		
– 0xFFFD1108	Channel 1 Configuration Register	CCR1	**page 26-89**
– 0xFFFD1110	Channel 1 Pointer Status Register	CPSR1	**page 26-92**
– 0xFFFD1118–0xFFFD113F	reserved		
– 0xFFFD1140	Channel 1 Current Descriptor Pointer Register	CDPR1	**page 26-98**
– 0xFFFD1148	Channel 1 Fetch FIFO	CFF1	**page 26-99**
– 0xFFFD1150–0xFFFD117F	reserved		
– 0xFFFD1180–0xFFFD11BF	Channel 1 Descriptor Buffer	DB1	**page 26-100**
– 0xFFFD11C0–0xFFFD11DF	Channel 1 Gather Link Tables	—	**page 26-76**
– 0xFFFD11E0–0xFFFD11FF	Channel 1 Scatter Link Tables	—	**page 26-76**
– 0xFFFD1200–0xFFFD1207	reserved		
– 0xFFFD1208	Channel 2 Configuration Register	CCR2	**page 26-89**
– 0xFFFD1210	Channel 2 Pointer Status Register	CPSR2	**page 26-92**
– 0xFFFD1218–0xFFFD123F	reserved		
– 0xFFFD1240	Channel 2 Current Descriptor Pointer Register	CDPR2	**page 26-98**
– 0xFFFD1248	Channel 2 Fetch FIFO	CFF2	**page 26-99**
– 0xFFFD1250–0xFFFD127F	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFFD1280–0xFFFD12BF	Channel 2 Descriptor Buffer	DB2	**page 26-100**
– 0xFFFD12C0–0xFFFD12DF	Channel 2 Gather Link Tables	—	**page 26-76**
– 0xFFFD12E0–0xFFFD12FF	Channel 2 Scatter Link Tables	—	**page 26-76**
– 0xFFFD1300–0xFFFD1307	reserved		
– 0xFFFD1308	Channel 3 Configuration Register	CCR3	**page 26-89**
– 0xFFFD1310	Channel 3 Pointer Status Register	CPSR3	**page 26-92**
– 0xFFFD1318–0xFFFD133F	reserved		
– 0xFFFD1340	Channel 3 Current Descriptor Pointer Register	CDPR3	**page 26-98**
– 0xFFFD1348	Channel 3 Fetch FIFO	CFF3	**page 26-99**
– 0xFFFD1350–0xFFFD137F	reserved		
– 0xFFFD1380–0xFFFD13BF	Channel 3 Descriptor Buffer	DB3	**page 26-100**
– 0xFFFD13C0–0xFFFD13DF	Channel 3 Gather Link Tables	—	**page 26-76**
– 0xFFFD13E0–0xFFFD13FF	Channel 3 Scatter Link Tables	—	**page 26-76**
– 0xFFFD1400–0xFFFD1407	reserved		
– 0xFFFD1408	Channel 4 Configuration Register	CCR4	**page 26-89**
– 0xFFFD1410	Channel 4 Pointer Status Register	CPSR4	**page 26-92**
– 0xFFFD1418–0xFFFD143F	reserved		
– 0xFFFD1440	Channel 4 Current Descriptor Pointer Register	CDPR4	**page 26-98**
– 0xFFFD1448	Channel 4 Fetch FIFO	CFF4	**page 26-99**
– 0xFFFD1450–0xFFFD147F	reserved		
– 0xFFFD1480–0xFFFD14BF	Channel 4 Descriptor Buffer	DB4	**page 26-100**
– 0xFFFD14C0–0xFFFD14DF	Channel 4 Gather Link Tables	—	**page 26-76**
– 0xFFFD14E0–0xFFFD14FF	Channel 4 Scatter Link Tables	—	**page 26-76**
– 0xFFFD1400–0xFFFD1BF7	reserved		
– 0xFFFD1BF8	Controller IP Block Revision Register	CIPBRR	**page 26-79**
– 0xFFFD1C00–0xFFFD1FFF	reserved		
– 0xFFFD2000	DEU Mode Register	DEUMR	**page 26-113**
– 0xFFFD2008	DEU Key Size Register	DEUKSR	**page 26-114**
– 0xFFFD2010	DEU Data Size Register	DEUDSR	**page 26-115**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFFD2018	DEU Reset Control Register	DEURCR	**page 26-116**
– 0xFFFD2020–0xFFFD2027	reserved		
– 0xFFFD2028	DEU Status Register	DEUSR	**page 26-117**
– 0xFFFD2030	DEU Interrupt Status Register	DEUISR	**page 26-118**
– 0xFFFD2038	DEU Interrupt Mask Register	DEUIMR	**page 26-118**
– 0xFFFD2040–0xFFFD204F	reserved		
– 0xFFFD2050	DEU End_of_Message Register	DEUEOMR	**page 26-122**
– 0xFFFD2058–0xFFFD20FF	reserved		
– 0xFFFD2100	DEU IV Register	DEUIVR	**page 26-122**
– 0xFFFD2108–0xFFFD23FF	reserved		
– 0xFFFD2400	DEU Key 1 Register	DEUKR1	**page 26-123**
– 0xFFFD2408	DEU Key 2 Register	DEUKR2	**page 26-123**
– 0xFFFD2410	DEU Key 3 Register	DEUKR3	**page 26-123**
– 0xFFFD2418–0xFFFD27FF	reserved		
– 0xFFFD2800–0xFFFD2FFF	DEU Input FIFO/Output FIFO	—	**page 26-123**
– 0xFFFD3000–0xFFFD3FFF	reserved		
– 0xFFFD4000	AESU Mode Register	AESUMR	**page 26-124**
– 0xFFFD4008	AESU Key Size Register	AESUKSR	**page 26-126**
– 0xFFFD4010	AESU Data Size Register	AESUDSR	**page 26-127**
– 0xFFFD4018	AESU Reset Control Register	AESURCR	**page 26-128**
– 0xFFFD4020–0xFFFD4027	reserved		
– 0xFFFD4028	AESU Status Register	AESUSR	**page 26-129**
– 0xFFFD4030	AESU Interrupt Status Register	AESUISR	**page 26-130**
– 0xFFFD4038	AESU Interrupt Mask Register	AESUIMR	**page 26-132**
– 0xFFFD4040–0xFFFD404F	reserved		
– 0xFFFD4050	AESU End_of_Message Register	AESUEOMR	**page 26-134**
– 0xFFFD4058–0xFFFD40FF	reserved		
– 0xFFFD4100	AESU Context Register 1	AESUCR1	**page 26-134**
– 0xFFFD4108	AESU Context Register 2	AESUCR2	**page 26-134**
– 0xFFFD4110	AESU Context Register 3	AESUCR3	**page 26-134**
– 0xFFFD4118	AESU Context Register 4	AESUCR4	**page 26-134**
– 0xFFFD4120	AESU Context Register 5	AESUCR5	**page 26-134**
– 0xFFFD4128	AESU Context Register 6	AESUCR6	**page 26-134**
– 0xFFFD4130	AESU Context Register 7	AESUCR7	**page 26-134**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFFD4138–0xFFFD43FF	reserved		
– 0xFFFD4400	AESU Key Register 1	AESUKR1	**page 26-135**
– 0xFFFD4408	AESU Key Register 2	AESUKR2	**page 26-135**
– 0xFFFD4410	AESU Key Register 3	AESUKR3	**page 26-135**
– 0xFFFD4418–0xFFFD47FF	reserved		
– 0xFFFD4800–0xFFFD4FFF	AESU Input FIFO/Output FIFO	—	**page 26-136**
– 0xFFFD5000–0xFFFD5FFF	reserved		
– 0xFFFD6000	MDEU Mode Register	MDEUMR	**page 26-137**
– 0xFFFD6008	MDEU Key Size Register	MDEUKSR	**page 26-139**
– 0xFFFD6010	MDEU Data Size Register	MDEUDSR	**page 26-139**
– 0xFFFD6018	MDEU Reset Control Register	MDEURCR	**page 26-140**
– 0xFFFD6020–0xFFFD6027	reserved		
– 0xFFFD6028	MDEU Status Register	MDEUSR	**page 26-141**
– 0xFFFD6030	MDEU Interrupt Status Register	MDEUISR	**page 26-142**
– 0xFFFD6038	MDEU Interrupt Mask Register	MDEUIMR	**page 26-144**
– 0xFFFD6040	MDEU ICV Size Register	MDEUICVSR	**page 26-146**
– 0xFFFD6048–0xFFFD604F	reserved		
– 0xFFFD6050	MDEU End_of_Message Register	MDEUEOMR	**page 26-147**
– 0xFFFD6058–0xFFFD60FF	reserved		
– 0xFFFD6100–0xFFFD6120	MDEU Context Registers	MDEUCR	**page 26-147**
– 0xFFFD6128–0xFFFD63FF	reserved		
– 0xFFFD6400–0xFFFD647F	MDEU Key Registers	MDEUKR	**page 26-149**
– 0xFFFD6480–0xFFFD67FF	reserved		
– 0xFFFD6800–0xFFFD6FFF	MDEU Input FIFO	—	**page 26-149**
– 0xFFFD7000–0xFFFD7FFF	reserved		
– 0xFFFD8000	AFEU Mode Register	AFEUMR	**page 26-150**
– 0xFFFD8008	AFEU Key Size Register	AFEUKSR	**page 26-151**
– 0xFFFD8010	AFEU Context/Data Size Register	AFEUCDSR	**page 26-152**
– 0xFFFD8018	AFEU Reset Control Register	AFEURCR	**page 26-153**
– 0xFFFD8020–0xFFFD8027	reserved		
– 0xFFFD8028	AFEU Status Register	AFEUSR	**page 26-154**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFFD8030	AFEU Interrupt Status Register	AFEUISR	**page 26-155**
– 0xFFFD8038	AFEU Interrupt Mask Register	AFEUIMR	**page 26-157**
– 0xFFFD8040–0xFFFD804F	reserved		
– 0xFFFD8050	AFEU End_of_Message Register	AFEUEOMR	**page 26-159**
– 0xFFFD8058–0xFFFD80FF	reserved		
– 0xFFFD8100–0xFFFD81FF	AFEU Context Memory	—	**page 26-159**
– 0xFFFD8200	AFEU Context Memory Pointer Register	AFEUCMPR	**page 26-160**
– 0xFFFD8208–0xFFFD83FF	reserved		
– 0xFFFD8400	AFEU Key Register 1	AFEUKR1	**page 26-160**
– 0xFFFD8408	AFEU Key Register 2	AFEUKR2	**page 26-160**
– 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.	—	**page 26-160**
– 0xFFFD9000–0xFFFD9FFF	reserved		
– 0xFFFDA000	RNG Mode Register	RNGMR	**page 26-179**
– 0xFFFDA008–0xFFFDA00F	reserved		
– 0xFFFDA010	RNG Data Size Register	RNGDSR	**page 26-180**
– 0xFFFDA018	RNG Reset Control Register	RNGRCR	**page 26-181**
– 0xFFFDA020–0xFFFDA027	reserved		
– 0xFFFDA028	RNG Status Register	RNGSR	**page 26-182**
– 0xFFFDA030	RNG Interrupt Status Register	RNGISR	**page 26-183**
– 0xFFFDA038	RNG Interrupt Mask Register	RNGIMR	**page 26-184**
– 0xFFFDA040–0xFFFDA04F	reserved		
– 0xFFFDA050	RNG End_of_Message Register	RNGEOMR	**page 26-186**
– 0xFFFDA058–0xFFFDA0FF	reserved		
– 0xFFFDA800–0xFFFDAFFF	RNG Output FIFO	—	**page 26-186**
– 0xFFFDB000–0xFFFDBFFF	reserved		
– 0xFFFDC000	PKEU Mode Register	PKEUMR	**page 26-101**
– 0xFFFDC008	PKEU Key Size Register	PKEUKSR	**page 26-103**
– 0xFFFDC010	PKEU Data Size Register	PKEUDSR	**page 26-105**
– 0xFFFDC018	PKEU Reset Control Register	PKEURCR	**page 26-106**

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFFDC020–0xFFFDC027	reserved		
– 0xFFFDC028	PKEU Status Register	PKEUSR	**page 26-107**
– 0xFFFDC030	PKEU Interrupt Status Register	PKEUISR	**page 26-108**
– 0xFFFDC038	PKEU Interrupt Mask Register	PKEUIMR	**page 26-110**
– 0xFFFDC040	PKEU AB Size Register	PKEUIABSR	**page 26-104**
– 0xFFFDC048–0xFFFDC04F	reserved		
– 0xFFFDC050	PKEU End_of_Message Register	PKEUEOMR	**page 26-111**
– 0xFFFDC058–0xFFFDC1FF	reserved		
– 0xFFFDC200–0xFFFDC23F	PKEU Parameter Memory A0	—	**page 26-111**
– 0xFFFDC240–0xFFFDC27F	PKEU Parameter Memory A1	—	**page 26-111**
– 0xFFFDC280–0xFFFDC2BF	PKEU Parameter Memory A2	—	**page 26-111**
– 0xFFFDC2C0–0xFFFDC2FF	PKEU Parameter Memory A3	—	**page 26-111**
– 0xFFFDC300–0xFFFDC33F	PKEU Parameter Memory B0	—	**page 26-111**
– 0xFFFDC340–0xFFFDC37F	PKEU Parameter Memory B1	—	**page 26-111**
– 0xFFFDC380–0xFFFDC3BF	PKEU Parameter Memory B2	—	**page 26-111**
– 0xFFFDC3C0–0xFFFDC3FF	PKEU Parameter Memory B3	—	**page 26-111**
– 0xFFFDC400–0xFFFDC4FF	PKEU Parameter Memory E	—	**page 26-111**
– 0xFFFDC500–0xFFFDC7FF	reserved		
– 0xFFFDC800–0xFFFDC8FF	PKEU Parameter Memory N	—	**page 26-111**
– 0xFFFDC900–0xFFFDDFFF	reserved		
– 0xFFFDE000	KEU Mode Register	KEUMR	**page 26-161**
– 0xFFFDE008	KEU Key Size Register	KEUKSR	**page 26-163**
– 0xFFFDE010	KEU Data Size Register	KEUDSR	**page 26-164**
– 0xFFFDE018	KEU Reset Control Register	KEURCR	**page 26-165**
– 0xFFFDE020–0xFFFDE027	reserved		
– 0xFFFDE028	KEU Status Register	KEUSR	**page 26-166**
– 0xFFFDE030	KEU Interrupt Status Register	KEUISR	**page 26-167**
– 0xFFFDE038	KEU Interrupt Mask Register	KEUIMR	**page 26-169**
– 0xFFFDE040–0xFFFDE047	reserved		

Table 9-9. Consolidated Memory Map (Continued)

Address	Name/Status	Acronym	Reference
– 0xFFFDE048	KEU Data Out Register	KEUDOR	**page 26-171**
– 0xFFFDE050	KEU End_of_Message Register	KEUEOMR	**page 26-172**
– 0xFFFDE058–0xFFFDE0FF	reserved		
– 0xFFFDE100	KEU IV1 Register	KEUIVR1	**page 26-173**
– 0xFFFDE108	KEU ICV_IN Register	KEUICVIR	**page 26-174**
– 0xFFFDE110	KEU IV2 Register	KEUIVR2	**page 26-175**
– 0xFFFDE118	KEU Context Register 1	KEUCR1	**page 26-176**
– 0xFFFDE120	KEU Context Register 2	KEUCR2	**page 26-176**
– 0xFFFDE128	KEU Context Register 3	KEUCR3	**page 26-176**
– 0xFFFDE130	KEU Context Register 4	KEUCR4	**page 26-176**
– 0xFFFDE138	KEU Context Register 5	KEUCR5	**page 26-176**
– 0xFFFDE140	KEU Context Register 6	KEUCR6	**page 26-176**
– 0xFFFDE148–0xFFFDE3FF	reserved		
– 0xFFFDE400	KEU Key Data Register 1	KEUKDR1	**page 26-177**
– 0xFFFDE408	KEU Key Data Register 2	KEUKDR2	**page 26-177**
– 0xFFFDE410	KEU Key Data Register 3	KEUKDR3	**page 26-178**
– 0xFFFDE418	KEU Key Data Register 4	KEUKDR4	**page 26-178**
– 0xFFFDE420–0xFFFDE7FF	reserved		
– 0xFFFDE800–0xFFFDEFFF	KEU Input FIFO/Output FIFO	—	**page 26-178**
• 0xFFFDF000–0xFFFFEFFF	reserved		
0xFFFFF000–0xFFFFFFFF	reserved		

MSC8144E SC3400 DSP Subsystem 10

Each SC3400 core is embedded in an DSP subsystem that enhances the power of the SC3400 core and provides a simple interface to each SC3400 core. The DSP core subsystem includes:

- SC3400 core
- Internal memory subsystem:
  - Memory management unit (MMU)
  - Instruction channel and L1 Instruction cache (16 KB, 8-way)
  - Data channel, Data cache (32 KB, 8-way), and write queue
- Interrupt processing using the Embedded Programmable Interrupt Controller (EPIC)
- Real-time debug support with a JTAG controller and test access port (TAP), on-chip emulator (OCE), and debug and profiling unit (DPU)
- Dual timer
- Two interfaces (instruction and data) to link the QBus structure supported by the SC3400 core with the MBus used to interface with the other functional blocks in the MSC8144 device
- Five operating states including two core power saving modes

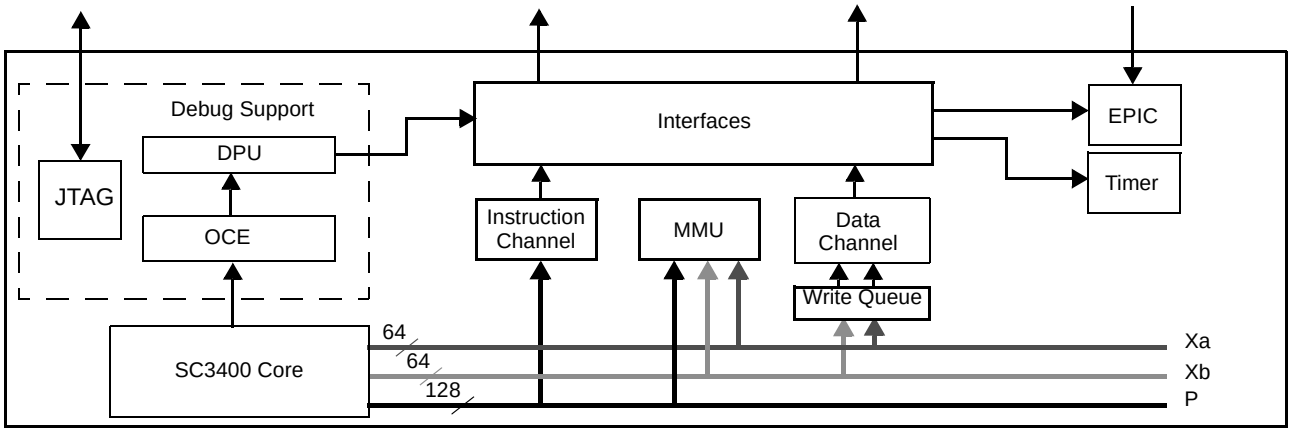


Figure 10-1. DSP core subsystem

Note: *The SC3400 DSP Core Reference Manual* and the *MSC8144 SC3400 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 SC3400 DSP Core

The SC3400 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 and the SC1400 core used in the MSC711x DSP family and delivers up to 3200/4000 16-bit MMACS using an internal 800 MHz/1 GHz clock at 1 V. The SC3400 core efficiently deploys a novel variable-length execution set (VLES) execution model, maximizing parallelism by allowing multiple address generation and data arithmetic logic units to execute multiple operations in a single clock cycle. The SC3400 can sustain this high performance over time, owing to the flexibility of its data execution units. The four data execution units can operate simultaneously in any combination. For example, the core can execute four multiply-accumulate operations in a single clock, or one MAC, two arithmetic/logical operations and one bit field operation. All four data ALUs are identical. This permits great flexibility in the assignment and execution of instructions, increasing the likelihood that four execution units can be kept busy on any given cycle and enabling programs to take better advantage of the parallel architecture of the core.

The SC3400 core has the same organization as the SC140 core and has the following functional units:

- Data Arithmetic and Logic Unit (DALU) that includes a Data Register File and four Arithmetic Logic Units (ALU). Each ALU includes a MAC unit and a Bit Field Unit (BFU)
- Address Generation Unit (AGU), including an Address Register File, two Address Arithmetic Units (AAU) and a Bit Mask Unit (BMU)
- A Program Sequencer and Controller (PSEQ)
- To provide data exchange between the core and the other internal blocks, the following buses are implemented:
  - Two data memory buses, Xa and Xb (with their respective address and data pairs: XaBA, XDBA and XaBB, XDBB), used for all data transfers between the core and the DSP core subsystem MMU, instruction channel, and data channel.
  - Program bus (P bus) with its address and data buses (PAB, PDB) for fetching program words from memory via the instruction channel to the core

Note: See **Chapter 2**, *SC3400 Core Overview* and the *SC3400 DSP Core Reference Manual* for details on the SC3400 core.

10.2 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, programming 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, pre-fetch enable, write-policy, cacheability, and so forth.

The MMU performs address translation on external (to the DSP core subsystem) addresses, from virtual addresses (used by the software that runs on the core) to physical addresses (used by the MBuses). Address translation is needed for several reasons, including:

- Enabling the 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 re-use
- Allowing true dynamic code relocation without any performance cost or overhead

The same virtual addresses can be re-used between tasks, with no need to flush the caches between tasks. This is because the caches also 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 be set (among other things) to cacheable/non-cacheable, pre-fetch policy, shared/non-shared, and more. The MMU controls up to 20 data and up to 12 program segment descriptors through the memory attributes and translation table (MATT).

The MMU also handles all memory protection. If an attempted memory access is not permitted, the MMU aborts the memory accesses by signalling the relevant DSP core subsystem components via an MMU exception indication back to the core. The MMU also stores the address and attributes of accesses that are not permitted. The MMU registers are memory-mapped on the DQBus and are programmed using core move and bit-mask instructions. Memory protection is required to increase the reliability of the system, so that errant tasks are not allowed to ruin the privileged state and the state of other tasks. Program and data accesses from

the core can be in either user or supervisor level. The MMU checks each access to determine if it matches the permissions defined for this task in the MATT. If it does not, the access is killed and a memory exception is generated.

The MMU controls memory accesses and translates virtual addresses to physical addresses. It serves these main functions:

- Supplying program and data access hardware protection for two privilege levels (User and Supervisor) in order to enable multi task protected system.
- Implementing a high-speed address translation mechanism that translates from virtual to physical addresses in order to support memory relocation.
- Providing cache and bus controls for advanced memory management.
- Provides task protection
- Supports multi-tasking
- Defines the memory and access attributes of memory regions

Note: See the *MSC8144 SC3400 DSP Core Subsystem Reference Manual* for details on the MMU.

10.3 Instruction Channel

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 (off-platform) memory by the IFU (ICache miss). The IFU operates in parallel to the core to implement a pre-fetch 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 Cache (ICache):
- 16 KB
- 8 ways with 8 lines per way
- Multi-task support
- Real-time support through locking flexible boundaries
- Pre-fetch capability
- Software coherency support

Note: See the *MSC8144 SC3400 DSP Core Subsystem Reference Manual* for details on the Instruction Channel and the L1 ICache.

10.4 Data Channel and Write Queue

The Data Channel comprises the:

- Data cache (DCache)
- Data fetch unit (DFU)
- Data control unit (DCU)
- Write-back buffer (WBB)
- Data write buffer (DWB)

This channel 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 to. 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 (off-platform) memory by the DFU, and driven to the core. The DFU operates in parallel with the core and implements a pre-fetch algorithm to load to the DCache. Because there is a high probability that the information will be needed again, the loading of the data can reduce the number of data cache misses. The channel differentiates between cacheable and non-cacheable addresses. For cacheable addresses, it supports the write-back allocate writing policy. The selection is made on an address segment basis, as programmed in the MMU. The Data Channel supports the arrangement of data in both big- and little-endian formats. Core data types can be byte, word, long (4 byte) or 2 long (8 byte) wide. The data channel has the following features:

- 32 KB
- 8 ways with 16 lines per way
- Capable of serving two data accesses in parallel (Xa, Xb)
- Multi-task support
- Real-time support through locking flexible boundaries
- Software coherency support
- Write-back writing policy
- Pre-fetch capability

Note: See the *MSC8144 SC3400 DSP Core Subsystem Reference Manual* for details on the Data Channel and the L1 DCache.

10.5 Interrupt Processing

The DSP core subsystem processes interrupts and exceptions generated by conditions either inside the SC3400 core, the DSP core subsystem modules, or external sources. In general, the prioritization and arbitration between the DSP core subsystem and the external interrupt sources is done in the Embedded Program Interrupt Controller (EPIC) module.

The source of interrupts in the DSP core subsystem are:

- SC3400 core internal exceptions. The SC3400 core has several internal interrupt sources such as trap, illegal instruction, debug exceptions, and DALU overflow. For details on the SC3400 core interrupts, see the *SC3400 DSP Core Reference Manual*.
- MMU interrupts. The MMU has eight dedicated interrupt lines to the core that are treated as internal core interrupts even though the MMU is in the DSP core subsystem. The MMU itself has a number of interrupt sources, some internal to the MMU (such as protection violation and MATT) and some external to the MMU (such as EDC error). The MMU interrupts are synchronized to the core accesses and are precise interrupts. For more information about the MMU interrupts, see the *MSC8144 SC3400 DSP Core Subsystem Reference Manual.*
- External interrupts. This includes interrupts generated by the MSC8144E internal peripherals and external interrupt input lines.

The role of the EPIC module is to manage the interrupt inputs. The EPIC manages the interrupts using a fixed set of priority rules and passes interrupts with the highest priority at any given time to the core. The EPIC also manages the acknowledgment of edge-triggered interrupts.

The EPIC handles up to 256 interrupt inputs, including 222 interrupts external to the DSP core subsystem that can be independently configured as maskable or non-maskable interrupts (NMIs). Interrupts external to the device use the standard interrupt interface protocol.

Note: See **Chapter 13**, *Interrupt Handling* and the *MSC8144 SC3400 DSP Core Subsystem Reference Manual* for details about interrupt processing.

10.6 Real-Time Debug Support

The debug support includes an internal JTAG controller and test access port (TAP), a debug and profiling unit (DPU), and the on-chip emulator (OCE) module. The TAP supplies the **IEEE**-specified external interface and standard JTAG debugging capability. The DPU supports the debugging and profiling of the DSP core subsystem in cooperation with the OCE block. The main debug and profiling requirements that the DPU helps support include:

- *Cache support*. Cache usage can have a significant effect on the overall performance of the DSP core subsystem. Application developers require observability of the internal state and cumulative statistics for the cache performance of their applications.

- *Multi-tasking support*. The DSP core subsystem supports a multi-task software environment under the control of an RTOS. There is a need to debug and profile the application in this environment.
- *High data collection capacity*. Frequently in DSP applications, successive runs are not identical, due to changes in input data and the like. Therefore, it is important to accumulate as much information as possible during a single run.
- *Real-time support*. Debugging real-time systems requires that the application run for extended periods of time. Profiling activity, in particular, can be performed without stopping the application.
- *Non-intrusive operation*. DSP applications work in a tightly-coupled environment that includes other processors, peripherals and components that operate in parallel. To the extent possible, debug and profile activity should be performed in a way that does not change the timing behavior of the application during this process.

The DPU has three major debug and profiling functions:

- Counting system events.
- Managing the Virtual Trace Buffer (VTB).
- Generating data.

The DPU has the following characteristics:

- Enables parallel counting of platform events in 6 dedicated counters, from more than 40 events.
- Filter, process and add task ID and profiling information on the OCE PC trace information.

The OCE has the following characteristics:

- Communicates with the host debugger through the JTAG Test Access Port (TAP) controller
- Makes it possible for the SC3400 core to enter the debug processing state upon a varied set of conditions, during which it is possible to:
  - Single step
  - Execute core commands inserted from the host debugger, enabling for example to upload and download memory and core registers.
  - Set up to 6 address related breakpoints, on either PC or a data address
  - Set a data breakpoint on a data value, optionally combined with a data address
  - Generate the PC tracing flow, optionally filtered to a subset of events such as only jumps/returns from subroutine, interrupts, and so forth.

See **Chapter 25**, *Debugging, Profiling, and Performance Monitoring* for details.

10.7 Dual Timer

Each DSP core subsystem includes two 32-bit general-purpose counters with pre-loading capability. It counts clocks at the core frequency. The timers are intended mainly for operating system use.

Note: See **Chapter 21**, *Timers* for details about the timers.

10.8 Interfaces

The DSP core subsystem supports separate buses for data and instructions and uses two bridges (IQ2M and DQ2M) to connect between the core instruction QBus (P lines) and the data QBus (Xa and Xb lines) used by the SC3400 core and the MBus that connects to the other blocks in the MSC8144 device. The bridges are placed between two clock domains: internal and external. The two clock domains are asynchronous. The internal clock domain runs at the core frequency and the external clock is slower to comply with the other block requirements.

The interfaces include the following:

- Bus widths
  - 128-bit data buses (read/write) for both buses
  - 32-bit address bus for both buses
  - 8-bit line byte count
- Frequency reduction from the QBus frequency and the system frequency
- Bus error interrupt generation
- Idle indications that allow the extend core to enter STOP mode
- Burst transactions up to 128 bytes
- Burst wraparounds on both sides of the bridges (for bursts of two or more beats)

10.9 DSP Core Subsystem Operating States

This section describes the DSP core subsystem processing states. These states are related to the SC3400 processing states, but include additional platform-specific attributes. The processing states include:

- Reset
- Execution
- Debug
- WAIT
- STOP

10.9.1 Reset State

The Reset processing state is the initial state of the platform, entered upon power up. While in this state, all DSP core subsystem internal modules receive the reset signal and initialize their internal logic to a predefined state. While the platform is in this state, the SC3400 is in its Reset processing state. Reset can be entered from all states.

10.9.2 Execution State

In Execution state, the SC3400 core is free-running and executing instructions. The core operates in one of three working modes (Normal, Exception or User), having one of two privilege levels (Supervisor, User) while in the Execution state. See the *SC3000 Core Reference Manual* for details.

10.9.3 Debug State

When the platform is in the Debug state, the SC3400 core enters its Debug processing state, and instruction processing is halted. After a delay, all subsequent platform 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 platform registers and memory locations in order to develop and debug the application that is intended to run on the architecture. See the *SC3000 Core Reference Manual* for details about Debug state. The platform 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 commands by the core (depending on the configuration of the OCE)
- An event in the DPU (depending on the configuration of the DPU and OCE)

The platform exits the Debug state when receiving the proper transaction from the external debugging agent through the JTAG port, or a reset signal. The ICache and DCache blocks have block-specific Debug modes, which can be activated in only one manner: the Platform is in 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. The state of the cache array could be written to as well. See **Chapter 11**, *Internal Memory Subsystem* for details.

10.9.4 WAIT State

This state is entered by executing the WAIT instruction on the SC3400 core, which enters the WAIT processing state. During this state, all the SC3400 core clocks are stopped. In addition, the clocks of most of the DSP core subsystem components are stopped. The only DSP core subsystem blocks that receive an active clock during WAIT state are those that are involved in waking up the DSP core subsystem from this state. These modules are: the EPIC, the Timer, the DPU.

The WAIT State is an intermediate power-saving state, consuming more power than the STOP state, but less than Execution state. The platform exits WAIT state when an enabled interrupt request is asserted, or the platform is transferred to Debug state, or to Reset state.

10.9.5 STOP State

This state is entered by executing the STOP instruction on the SC3400 core, which enters the STOP processing state. During the STOP state, all DSP core subsystem clocks are stopped. Only very limited logic that is required to wake up from STOP state receives an active clock. This state has the lowest power consumption for the DSP core subsystem.

Whenever the STOP command is issued to the SC3400 core, a hardware request-acknowledge signal protocol is initiated between the blocks of DSP core subsystem and the external environment in order to ensure that the platform is idle, and for the external environment to acknowledge that the platform can enter STOP state. The platform exits STOP state when a dedicated input is asserted from outside the platform, or when the platform is transferred to Debug state or to Reset state. See the *SC3000 DSP Core Reference Manual* for details about the STOP processing state and conditions for exiting it.

Note: In STOP processing state, the EPIC clocks are stopped. The EPIC serves as the combinatorial collector of all the enabled interrupts with priority larger than 0. The EPIC performs a logic OR operation for all the interrupt inputs. The output from OR operand is used to wake the core from STOP mode. Therefore, any enabled level interrupt causes the platform to wake from STOP state. You cannot use edge interrupts to wake the DSP core subsystem from STOP mode; therefore, disable all edge interrupts prior to entering the STOP state.

Note: The STOP instruction only stops operation of the SC3400 core. To stop the entire SC3400 DSP core subsystem, you must set the respective GCR2[COREn_STP_EN] bit (see **Section 8.2.2**, *General Configuration Register 2 (GCR2)*, on page 8-3 for details). After setting the bit, the core subsystem does not stop until the corresponding GCR1[COREn_STP_ACK] bit is set (see **Section 8.2.3**, *General Status Register 1 (GSR1)*, on page 8-4 for details).

10.9.6 State Transitions

Figure 10-2 describes the transitions between the functional states of the DSP core subsystem.

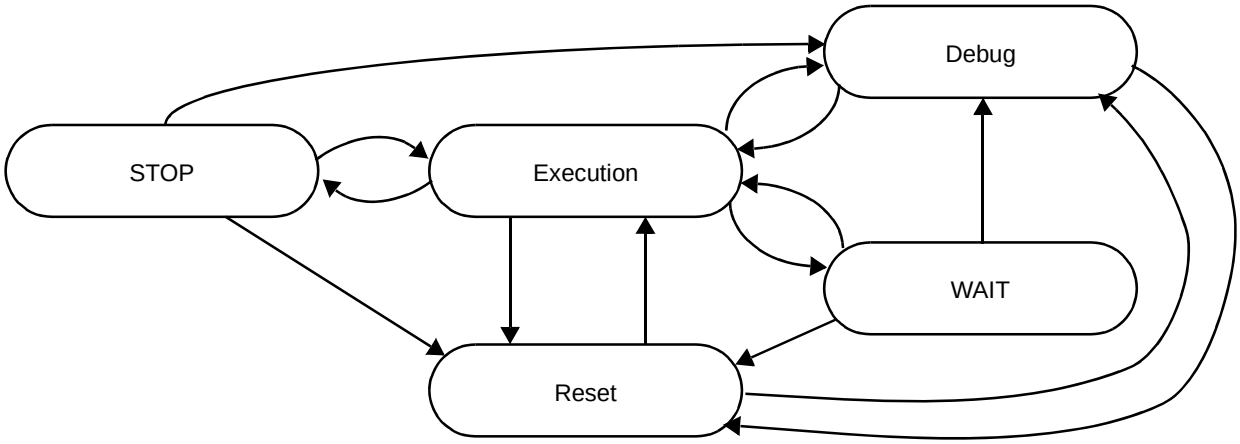


Figure 10-2. State Transition Diagram

10.9.6.1 Transition from Reset to Execution State

This transition is initiated by the assertion of the DSP core subsystem reset input port. The transition will occur assuming that the EE0 input port is not asserted. Several configuration inputs of the DSP core subsystem are sampled during reset to configure the DSP core subsystem. For example, the SC3400 core samples a dedicated input bus to determine the reset vector (first address) during the reset signal deassertion.

Note: The SC3400 performs the transition from the Reset state to the Execution state and Exception working mode. Please refer to the *SC3000 Core Reference Manual* for details on this transition.

10.9.6.2 Transition from Reset to Debug State

This transition is initiated by the negation of the DSP core subsystem reset input port while the EE0 input port is asserted.

10.9.6.3 Transition to Reset State

This transition is initiated by the assertion of the DSP core subsystem reset input port in any of the other states.

10.9.6.4 Transition from Execution to Debug State

This transition is initiated in one of the following ways:

- The JTAG DEBUG REQUEST instruction is requested by the host debugger through the JTAG TAP controller.
- EE0 is asserted while configured as input (this is the default configuration of this port).
- An OCE or DPU Debug event occurs (depends on proper configuration of the OCE and DPU registers)
- The DEBUG or DEBUGEV instruction is executed by the SC3400 core, with matching OCE configuration of its control registers.

10.9.6.5 Transition from Execution to STOP State

The DSP core subsystem can switch to the STOP state only in one of the non-protected modes (Exception or Normal) of the Execution state. The transition is done by executing the STOP instruction by the SC3400 core.

10.9.6.6 Transition from Execution to WAIT State

The DSP core subsystem can switch to the WAIT state only in one of the non-protected working modes (Execution or Normal) of the Execution state. The transition is done by executing the WAIT instruction by the SC3400 core.

10.9.6.7 Transition from STOP/WAIT to Debug state

This transition is initiated in one of the following conditions:

- The JTAG DEBUG REQUEST instruction is requested by the host debugger through the JTAG TAP controller.
- The EE0 port is asserted while configured as an input (the default configuration of this port).

Note: Entering Debug state is the default behavior of the OCE for the described events, but the OCE can be programmed to respond by entering the Execution state and process the event as a Debug exception. Therefore, before entering a STOP/WAIT state, make sure that the SC3400 core OCE is configured to exit the STOP/WAIT state and enter the Debug state and not the Execution state. See **Section 10.9.6.8** and **Section 10.9.6.9** for details. Please refer to the **Emulation and Debug (OCE)** chapter in the *SC3400 DSP Core Reference Manual* for OCE programming details.

10.9.6.8 Transition from STOP to Execution state

The transition from the STOP state is done by the assertion of one of the following exit from STOP signals:

- A dedicated external signal is asserted.
- The JTAG DEBUG REQUEST instruction is requested by the host debugger through the JTAG TAP controller (requires that the OCE is pre-programmed as described in the note below).
- The EE0 input is asserted (requires that the OCE is pre-programmed as described in the note below).

Note: To exit the STOP state via a debug event and enter Execution state (servicing the Debug exception), the OCE must be pre-programmed before entering the STOP state to respond to the above events as a Debug exception and not as a request to enter the Debug processing state. Please refer to the *OCE Reference Manual* for more information on the OCE programming.

10.9.6.9 Transition from WAIT to Execution state

The transition from the WAIT state is done by the assertion of one of the following *exit from WAIT signals*:

- An interrupt request, that is enabled by the core, is asserted.
- A non-maskable interrupt (NMI) request is issued.
- The JTAG DEBUG REQUEST command is issued to the external JTAG controller (requires that the OCE is pre-programmed as described in the note below).
- The EE0 pin is asserted (requires that the OCE is pre-programmed as described in the note below).

Note: In order to exit the WAIT state via a debug event and enter Execution state (servicing the Debug exception), the OCE must be pre-programmed before entering the WAIT state to respond to the above events as a Debug exception and not as a request to enter the Debug processing state. Please refer to the **Emulation and Debug (OCE)** chapter in the *SC3400 DSP Core Reference Manual* for OCE programming details.

10.9.6.10 Transition from Debug to Execution State

The transition is initiated by the external JTAG controller through its interface to the OCE module. The JTAG controller sets the dedicated exit bit in the OCE command register.

Internal Memory Subsystem 11

The internal memory system supports 10.96 MB of internal memory and includes:

- Memory management unit (MMU) per core.
- Instruction channel with 16 KB L1 ICache per core.
- Data channel with 32 KB L1 DCache per core.
- 128 KB L2 shared ICache.
- 512 KB M2 low-latency memory for critical data and temporary data buffering. Accessible from all DSP subsystems and all CLASS initiators via four interleaved ports.
- 10 MB 128-bit wide M3 memory accessed at up to 400 MHz. Accessible from all DSP subsystems and all CLASS initiators. Most applications run with no external memory.
- 96 KB of boot ROM accessible from the cores.

Note: The MMU, L1 ICache, and L1 DCache are part of the MSC8144 SC3400 DSP core subsystem. For detailed programming and functional information, refer to the *MSC8144 SC3400 DSP Core Reference Manual*, available with a signed non-disclosure agreement. Contact your local Freescale dealer or sales representative for more information.

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, pre-fetch 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 or 4 for the data fetch unit (DFU) and instruction fetch unit (IFU).
  - Pre-fetch line enable.
  - System/shared attributes
  - Global attributes
  - Write policy for data memory
  - L2 cache policy data memory

- 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 supervisor-level read allowed/not allowed access; for the User level, provides read allowed/not allowed access
  - 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, enables allocating 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.
- General Purpose registers 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.
- Platform error detection code (EDC) recovery scheme.
- Enable/disable EDC exception mechanism.
- Peripherals error handling.

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 pre-fetch 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 P 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 pre-fetch accesses upon cache miss that fetches data to the end of the cache line (256 bytes).
  - Pre-fetching is aborted in case of a new miss on a burst size boundary.
  - Programmable burst size of 1 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.
- Pseudo-LRU (PLRU) as the cache line replacement mechanism (LRM).
- Partial lock allows locking of a subset of cache lines based on ways boundaries to reduce the cache restoration penalty of a restored task. Instructions can be locked in the cache, preventing the thrashing of instructions that have expected reuse. This mechanism is useful during rapid task switching and prevents a situation in which a task thrashes important instructions associated with other tasks.
- Global lock allows locking of all cache lines to reduce the cache restoration penalty of a restored task. In this case, the cache does not serve cache misses.

- User-initiated cache sweep operations for coherency support. These operations are performed on each line in a user-specified address range. An invalidate discards the cache line (clears the valid bits).
- Cache debug mode in which the cache state (ETAG values, Valid, PLRU state) can be read and the memory array can be read or written.
- Dedicated programmable cache control registers that control or reflect its operation.
- EDC (error detection) support.
- Dedicated exceptions for each of the following events:
  - *End of sweep operation*. This exception indicates the completion of the sweep operation.
  - *XP non-cacheable hit access*. This exception indicates that an access is a hit access, even though the MMU classifies it as a non-cacheable access. This type of situation can occur if the memory space attributes changed in the MMU without invalidating the appropriate cache lines
  - *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 to. 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 to the core. The DFU operates in parallel with the core and implements a pre-fetch 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. For cacheable addresses, it supports the write-back allocate writing policy. The selection is made on an address segment basis, as programmed in the MMU. The data channel has the following features:

- Handling 2 parallel core accesses Xa/Xb 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 in this chapter.
- 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 pre-fetch accesses that will bring in data until the end of the cache line (256 bytes).
  - Pre-fetching is aborted in case of a new miss, on a burst size boundary.
  - Programmable burst size of 1 or 4 VBRs.
- Miss access that is identified as being pre-fetched (pre-fetch hit), will stall the core for reduced number of wait states relative to a simple miss.
- Cache miss upon a cacheable write performs a read first (write-allocate policy).
- Supports write back policy for updating higher level memory through the WBB.
- Hazard detection for reads that use data that was flushed and is still in the Write-Back Buffer; The access is stalled until the write back is complete.
- Supports Pseudo-LRU (PLRU) as the cache Line Replacement Mechanism (LRM).
- Partial lock allows locking of a subset of cache lines based on ways boundaries, to reduce cache restoration penalty of a restored task. Data can be locked in the cache, thus

preventing the thrashing of data expected to be used again. This mechanism is useful when rapid task switching is required, thereby preventing a situation in which a task thrashes important data associated with other tasks.

- Global lock allows locking of all cache lines to reduce cache restoration penalty of a restored task. Miss accesses are not served by the cache in this case.
- Supports user-initiated “cache sweep” operations for coherency support. These operations are performed on each line in a user-specified address range:
  - Synchronize: write back the cache line if it was modified, clearing its “dirty” bit without affecting its validity.
  - Flush: write back any cache line (and clear the “dirty” bit), and also invalidate it in the cache (clearing the “valid” bit).
  - Invalidate: discard the cache line without writing it back (clear both “dirty” and “valid” bits).
- Cache debug mode where the cache state (ETAG values, Valid-Dirty, PLRU state) could be read and the memory array could be read or written to.
- 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.
  - Xa/Xb non-cacheable hit access: This exception indicates that an access is a 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 were changed in the MMU without flushing/invalidating the appropriate cache lines
  - Xa/Xb 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 Instruction Cache

The shared level 2 multi-port interleaved ICache is highly optimized for multi-core DSP applications and minimizes miss ratio, latencies, and bus bandwidth requirements. The 8-way associative 128 KB L2 ICache contains two 64 KB banks and uses a 256-byte line size. When a cache miss occurs, it can fetch new data in a burst or as single accesses from the target memory. The optional fetch to the end of the line (prefetch) takes advantages of the spatial locality of the code. L2 ICache entries are invalidated via a cache invalidate command in the same way as in the SC3400 L1 ICache, which is useful when new code is written to M2, M3, or DDR memory that is already cached. All DSP subsystem instruction addresses are interleaved to two L2 ICache ports, so they can service multiple requesters concurrently if they access different interleaved banks. **Figure 11-1** shows the L2 ICache block diagram. The two memory banks (Bank 1 and Bank 2) both include a 64 KB instruction cache memory, an instruction fetch unit, and two bridges (one to the MBus for memory access and one to the IQBuses for all four cores).

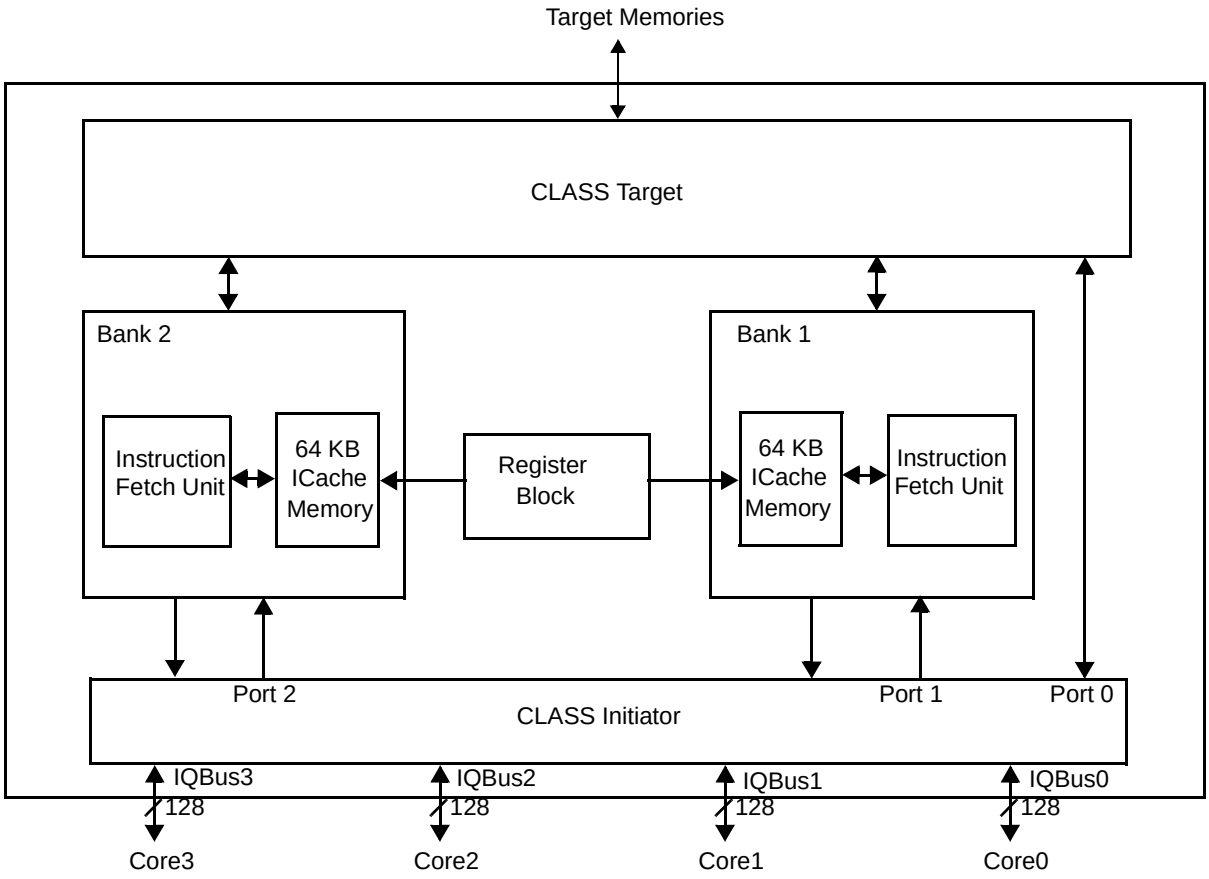


Figure 11-1. L2 ICache Block Diagram

The L2 ICache fetches code from a higher hierarchy memory and enables the cores (in case of a miss in their L1 ICache) to fetch a code from level 2 cache memory with reasonable degradation, instead of the high miss penalty to access higher hierarchy memory directly. Cacheable accesses activate the cache, and in case of a hit, return the data to the DSP core subsystem. In case of a miss, the instruction cache issues fetch requests to the higher-level memory, and can prefetch more data than requested (limited by the end of the cache line) to reduce the performance impact due to subsequent accesses.

The L2 ICache includes the following:

- 2×64 KB interleaved cache bank memories (total of 128 KB) with an interleave resolution of 256 bytes (line size).
- 8 way associative cache.
- 32 cache indexes (5 bits) in each bank.
- 128-bit valid bit resolution (VBR).
- 256-byte cache line size (16 VBRs).
- $8 \times 32 = 256$ cache lines (TAGs) per bank.
- L2 ICache initiator and target buses are 128 bits wide. The L2 ICache connects to four initiators (the 4 cores) and one target (the internal MBus).
- Supports both cacheable and non-cacheable accesses determined by a user configured address range.
- Supports wrap transactions.
- Supports big-endian data.
- Cache contains dedicated registers for programming and debug through the MBus.
- Supports cache sweep operations for coherency support, that is, you can invalidate cache lines within a specified address range.
- Hits are identified in pre-fetched data (pre-fetch hit).
- Supports pre-fetch accesses to the end of the cache line (256 bytes).
- Partial lock allows you to lock a subset of cache lines based on way boundaries.
- Supports a pseudo-LRU (PLRU) cache line replacement algorithm.
- Provides dedicated exceptions for each of the following events:
  - — Profiling interrupts:
    - Watch point event.
    - Overflow.
  - — Error detection and correction in L2 ICache memory (single-bit detection and correction with no assurance for multi-bit detection.
- Automatic built-in-self-test (ABIST) of the ICache memory.

11.4.1 CLASS

The Chip Level Arbitration and Switching System (CLASS) non blocking, interconnect fabric used in the L2 ICache is the same model as that used for linking the DSP core subsystems to the MSC8144E peripherals and MBus (see **Chapter 4**, *Chip-Level Arbitration and Switching System (CLASS)* for details). There are two CLASS modules in the L2 ICache, referred to as the CLASS initiator and the CLASS target. The CLASS initiator supports arbitration between multiple initiators and multiple targets. The CLASS initiator normalizes transactions using specified L2 ICache parameters. Non-cacheable accesses are supported by CLASS initiator address decoding.

The L2 ICache CLASS initiator supports four initiators (the four DSP core subsystems) and three targets (ports 1 and 2 for cache memory interleaving and port 0 for the CLASS target for a non-cacheable path). The CLASS initiator gets fetch requests from the DSP core subsystems and either

- passes cacheable requests to the relevant L1 ICache, or
- passes non-cacheable requests (single or burst) to the CLASS target.

When the requested data is ready, the CLASS initiator transfers it back to the DSP core subsystem that initiated the access.

The interleave between the two L2 ICache banks is done by the CLASS initiator using the lsb of the 6-bit cache index. After interleaving, the lsb of the 6 bit index is removed and each bank uses a 5-bit index, as shown in **Figure 11-2**.

Each cache way includes 32 cache lines (5 bit index after the 6 bit index is used to interleave between the banks). Each line includes 256 bytes of data, 16 bytes in each VBR. To locate the instruction cache line that matches the access, the tag is compared to those which are stored in the cache lines. All lines are compared simultaneously. In case of line match and valid status of the appropriate VBR in the cache, the access is served by cache memory without any latency penalty (unless there is a memory conflict).

Each access is received by the CLASS initiator and is transferred (as single transfers) to the bus bridge according to its address range. The bridge accesses activate the CLASS port, and in case of a hit, the port returns the data toward the bridge. In case of a miss, the fetch unit in the bank issues fetch requests to the higher-level memory, and can pre-fetch more data than requested, limited at most by the end of the cache line (the user can enable or disable the prefetch), to reduce the performance impact due to subsequent accesses.

The CLASS initiator port 0 is enabled out of reset and gets the full address range (0x00000–0xFFFFF). Port 1 should be enabled and gets an address range defined by the user in L2IC_CSA and L2IC_CEA; the reset range is 0x00000–0xFFFFF, but the cacheable window is disabled, which means that after reset all accesses are treated as non-cacheable. Port 2 is disabled after reset and accesses pass through it according to the interleaving.

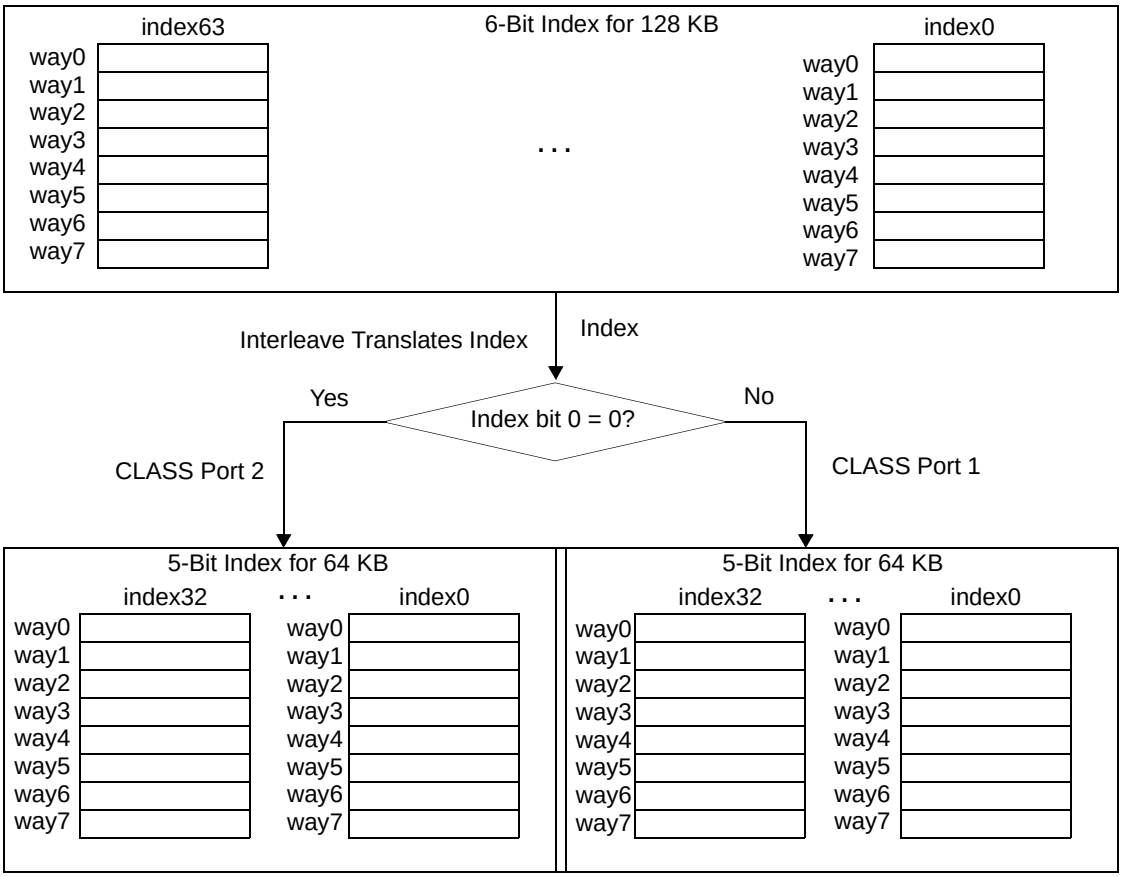


Figure 11-2. L2 ICache Interleaving

The CLASS target supports three initiators (the three CLASS initiator ports) and one target that connects to the device-level CLASS module. The target can support sixteen opened accesses because some of the L2 ICache targets (for example, M3 Memory and DDR) have a long latency and pipelined accesses to them must be supported.

11.4.2 ICache Modules

Each of the two ICache modules is a 64 KB 8-way associative cache. The module is designed to improve the SC3400 core average access time to data located in areas with high access latencies. The cache module includes a dedicated cache memory that serves cores hit accesses and to which data is fetched due to miss accesses.The memory unit inside each module is four single precompiled memories of 2048 72-bit words.

11.4.3 Instruction Fetch Unit

The instruction fetch unit is active only for L2 ICache cacheable accesses. It requests fetch sets from devices connected to the MBus and drives that data to the L2 ICache memory.

11.4.4 L2 ICache Register Block

This block contains L2 ICache registers which are configured through the MBus. These registers control:

- Sweep (invalidate) operation by address range or global sweep.
- Way locking by boundaries or global.
- The ICache memory on/off condition.
- Debug mode on/off and registers for reading during debug.
- Pre-Fetch on/off.
- User configured burst size for accesses from L2 ICache fetch unit toward higher level memory.

Note: See **Section 11.8.1** through **Section 11.8.8** for register details and programming model.

11.4.5 Global Attributes Support

Global attributes are used as additional external access attributes. These attributes are determined per MMU segment descriptor and are issued to the system. For the L2 ICache, these signals can be used to determine the cache policies.

11.4.6 Functional Mode of Operation

This is the normal processing state in which each DSP core subsystem freely executes instructions, and initiates memory accesses.

11.4.6.1 Enabling the L2 ICache

The CLASS does not return request acknowledge signals during reset. Therefore, the L2 ICache cannot respond to core requests during reset. After reset, the L2 ICache is in non-cacheable mode only. Use the following steps to activate the L2 ICache:

1. Exit the reset state.
2. Enable the two banks of the cache memory by setting the L2IC_CR2[CE] bit (see **Section 11.8.3**).
3. Configure the required cacheable address range in L2IC_CSA and L2IC_CEA (see **Section 11.8.9** and **Section 11.8.10**), which control port 1 of the CLASS initiator. If you do not configure these values, the initiator uses the default values. Set the L2IC_CEN[DEN] bit (see **Section 11.8.11**) to enable the cacheable range. Because the default value disables the address range, you must enable the window to enable the cache.
4. The L2 ICache can begin normal operation for cacheable and non-cacheable requests.

Note: Port 2 of the CLASS initiator is disabled at reset and must not be enabled by the user (otherwise the interleaving between the two banks will not work).

Do not disable the cache memory modules after they are enabled. To disable L2 ICache operation, disable the L2 Cache window by clearing the L2IC_CEN[DEN] bit and then write start and end addresses of 0x000000 to L2IC_CSA and L2IC_CEA.

If the cache is disabled after initialization, you must set the L2IC_CR2[CE] bit before setting the L2IC_CEN[DEN] bit to enable the cache.

11.4.6.2 L2 ICache Global Invalidation Command

The L2 ICache global invalidation command clears the valid bits in both cache modules in a single clock cycle. To execute a global invalidate, enter the sweep/invalidate command in the CC field and set the CGS bit in of L2IC_CR1 (see **Section 11.8.2**). The command also resets the PLRU machine.

11.4.6.3 L2 ICache Global Lock Mode

To enter global lock mode, set the L2IC_CR2[CGL] bit (see **Section 11.8.3**). When globally locked, new data fetches are not written to the L2 ICache and no lines are trashed. Hit accesses are served as usual. Miss accesses are served without updating the cache memory. The PLRU machine is not updated while the L2 ICache is locked. The L2 ICache Global Lock mode may be useful to improve the performance of some applications that may be sensitive to interrupts, for example. After asserting this mode, all open miss accesses are served and updated as if L2 ICache is not globally locked. The lock mode disables only new miss data updates that are initiated after the L2 ICache enters global lock mode.

The sweep operation overrides the global lock status. The value of the global lock bit is ignored during the cache sweep operation.

11.4.6.4 L2 ICache Partial Lock

L2 ICache can be partially locked to protect certain data. The lock is done in line resolution (16 VBRs). The replace mechanisms are disabled for data that does not correlate with the active cache lock boundaries. In case of TAG match, the data of a locked line can be updated with new fetches to this line, but L2 ICache lines are not replaced. The locking is done in way boundaries.

- The reduced open boundaries contain 2 or 4 ways.
- The exact partial lock configuration is programmed via dedicated register.

11.4.6.5 L2 ICache Line Replacement

When a line needs to be replaced in the L2 ICache, the pseudo-least-recently-used (PLRU) replacement algorithm is used. When a cache line is accessed, it is tagged as the most recently used line of the index. When a line miss occurs, the pseudo least recently used line is replaced by a new line, provided the cache is not completely locked or disabled. The PLRU bits in the L2 ICache are updated each time a cache line hit occurs based on the most recently used cache line.

11.4.6.5.1 PLRU Replacement

Line replacement is performed using a binary decision-tree, PLRU algorithm. There is an identifying bit for each cache way, L[0–7]. There are seven PLRU bits, B[0–6] for each index in the cache to determine the line to be replaced. The PLRU bits are updated when a new line is allocated or replaced and when there is a line hit. A line is selected for replacement according to the PLRU bit encoding shown in **Table 11-1** (initial values of B[6–0] is all zeros).

Table 11-1. PLRU Replacement Way Selection

PLRU Bits						Way Selected for Replacement
B0	0	**B1**	0	**B3**	0	L0
	0		0		1	L1
	0		1	**B4**	0	L2
	0		1		1	L3
	1	**B2**	0	**B5**	0	L4
	1		0		1	L5
	1		1	**B6**	0	L6
	1		1		1	L7

Figure 11-3 shows the decision tree used to generate the victim line in the PLRU algorithm.

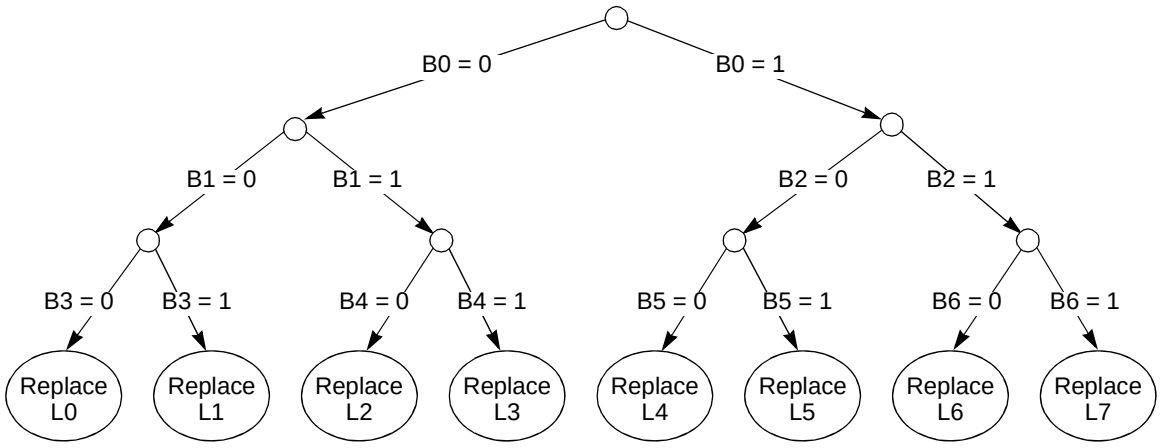


Figure 11-3. PLRU Replacement Algorithm

11.4.6.5.2 PLRU Bit Updates

Each time a cache line is accessed, it is tagged as the most recently used way of the index. For every line hit in the cache or when a new line is allocated, the PLRU bits for the index are updated using the rules specified in **Table 11-2**.

Table 11-2. PLRU Bit Update Rules

Line access	New State of the PLRU Bits						
	B0	B1	B2	B3	B4	B5	B6
L0	1	1	No change	1	No change	No change	No change
L1	1	1	No change	0	No change	No change	No change
L2	1	0	No change	No change	1	No change	No change
L3	1	0	No change	No change	0	No change	No change
L4	0	No change	1	No change	No change	1	No change
L5	0	No change	1	No change	No change	0	No change
L6	0	No change	0	No change	No change	No change	1
L7	0	No change	0	No change	No change	No change	0

Note: Only three PLRU bits are updated for any access.

11.4.6.5.3 PLRU Bits In Partial Lock

Partial lock is supported by freezing some of the PLRU bits according to the lock configuration. The frozen bits prevent some of the L2 ICache lines to be replaced and re-allocated for new miss access. At least two cache ways (2 lines in each index) are open in any partial lock configuration. In case of global lock all ways are locked.

11.4.6.5.4 PLRU Inverse Mechanism

PLRU bit status determines which line is next to allocate. The user can change this status (and by that change the order in which the lines are allocated) by using the inverse mechanism. The inverse operation is initiated by setting the L2IC_CR1[INV] bit. When you initiate inverse operation, all the PLRU status bit change their state, causing the allocation sequence to be inverted.

11.4.6.6 L2 ICache Sweep Operation

The L2 ICache sweep operation supports software coherency by enabling data invalidation for a specific address space programmed in the cache registers. Invalidation is useful if an instruction in the cache array will not be used, and required if the data in the system level memory was updated subsequent to the last cache fetch. The L2 ICache sweep operation uses a dedicated hardware counter and address comparators. The counter checks each and every cache valid line cycle by cycle (one line per cycle), and compares the line address, which is a combination of its Tag and Index, to an address range defined in dedicated registers. The sweep operation

invalidates a line if the line address matches the defined address range. The L2 ICache sweep operation is typically done as a background operation. To minimize the sweep operation period, you should prevent accesses during the sweep operation period.

Sweep programming has an effect only at the cycle in which a new sweep operation begins. The address boundaries are also sampled at this cycle. Once the sweep process is finished over all cache lines, the Cache Command Enable bit in L2IC_CR1 is cleared by hardware.

Different processes that are not aware of each other might try to initiate a sweep operation while another is still performing. The hardware supports only one sweep operation at a time. Also, since sweep operation requires preprogramming, nested software sweep operations might override other programming. L2 ICache does not support nested sweeps. The programer should use semaphore and follow the next steps to avoid nested sweeps.

- Verify that 0 is written to the dedicated semaphore.
- Use a BMTSET command to write 1 to the semaphore.
- Initiate the sweep command by writing 1 to the L2IC_CR1[CE] bit.
- Poll the sweep status (the L2IC_CR1[CE] bit) until it is low (this bit is cleared by hardware at the end of the sweep) and then write 0 to the dedicated semaphore.
- While using sweep operation, it is recommended to prevent situations in which there is an accesses to the memory space on which sweep operation is executed. If such situation does occur, however, sweep command or an access may override each other (a later event overrides an earlier one).
- You should use a central routine to manage the sweep operations that is called whenever a sweep is needed.
- An L2 ICache sweep operation initiation attempt is ignored when cache debug mode is enabled.

11.4.6.6.1 Invalidation Sweep

Invalidation of a line is executed by clearing all VBRs valid bits in the line. This operation is executed in a single clock cycle.

Note: The PLRU mechanism takes into consideration line invalidation by sweep command, and mark invalidated lines as least recently used, unless this line is locked by the PLRU lock mechanism. An update of VBRs valid bit during Fetch/Prefetch operation is done only if the line Tag is valid. This is to ignore Prefetch on invalidated lines.

11.4.6.6.2 L2 ICache Global Sweep Operation

If Cache Global Sweep bit in L2IC_CR1 is set, the sweep command ignores the programmed sweep address space and the address comparators are disabled. In such a case, the sweep command is performed on all cache lines. Note that global invalidation is performed in parallel on all cache lines in one cycle, not serially.

11.4.6.7 Fetch Operation

Fetch is an operation in which data is read from higher level memory and written to the cache memory. A miss access initiates the fetch operation. When the transferred data is not located in the same burst of the data which is required by the miss access, the operation is called a prefetch (this is a predictive fetch). If a fetch operation is required, and the appropriate cache memory location is full, a thrash operation is executed. This operation frees cache memory locations for the use of the required fetch of recent miss access.

Basic fetch sequence is as follows:

- A miss access invokes a fetch operation.
- A cache line is allocated and invalidated.
- Once the missed VBR is fetched, the initiator releases the fetch unit. Fetch and prefetch operations continue according to the fetch unit configuration.

11.4.7 Debug Mode

Cache debug mode is a special state that allows observation and control over the cache internal state. It allows access to the PLRU bits, TAG and VALID bits, and ICache memory.

Note: You can only invoke the L2 ICache debug mode when all the DSP core subsystems and all the L1 ICaches inside the DSP core subsystems are in the debug state.

The L2 ICache debug mode is initiated by setting a dedicated bit in the cache control registers. The L2 ICache state registers are accessible through the internal MBus when the cache debug mode bit is set. Any attempt to invoke the cache debug mode while a sweep operation is ongoing is ignored and discarded. In addition, you should only invoke the debug mode while the cache is idle. Use the following steps to invoke the L2 ICache debug mode:

1. Place all DSP core subsystems into debug mode.
2. Place all L1 ICaches in all DSP core subsystems into debug mode.
3. Verify that four DSP core subsystems are in debug mode by checking the core debug status bits in the General Status Register 1 (GSR1[CORE_DEBUG_STS]). All bits should be set to 1. See **Chapter 8**, *General Configuration Registers* for details.
4. Make sure that no accesses (read or write) are made to the L2 ICache and verify that the ICache is idle by checking the L2 ICache idle bit in GSR1 (GSR1[L2I_IDLE]).
5. Place the L2 ICache into debug mode by writing a 0b1 to L2IC_CR2[CDM].

You must make sure that the DSP core subsystems are all in the debug state before exiting the cache debug mode. Do not clear the Debug mode bit directly by setting the ECR[EX] bit to force the DSP core subsystem into its execution state.

Use the following steps to exit the L2 ICache debug mode:

1. Make sure that all DSP core subsystems are in Debug mode (the CORE3_DBE_STS, CORE2_DBE_STS, CORE1_DBE_STS, and CORE0_DBE_STS bits in the GSR are all set; see **Section 8.2.3**, *General Status Register 1 (GSR1)*, on page 8-4 for details).
2. Clear the L2IC_CR2[CDM] bit in the L2 ICache to exit debug mode.

If the cache debug mode bit is cleared, all cache operations that do not relate to debug behave as usual. Cache update mechanisms are disabled once cache debug mode bit is set by the user.

When L2 ICache is in debug mode, you can perform the following options:

- Read the dedicated L2 ICache debug registers:
  - Valid State Register (L2IC_VALID)
  - TAG State Register (L2IC_TAG)
  - Line Replacement Mechanism (LRM) State Register (L2IC_LRM).
- Read and write the cache memory array using the dedicated registers:
  - Debug data register (L2IC_DBGDATA)
  - Debug access register (L2IC_DBGACS).

11.4.7.1 Initialize/Read and Update State Registers

The following memory mapped state registers are located in the L2 ICache module: Tag State Register, Line Replacement Mechanism State Register, and Valid State Register. In cache debug mode, the content of the registers can be read via the MBus. This operation also updates the content of the accessed register according to a dedicated counter that selects the appropriate information to be sampled by the register. Each state register has a dedicated counter. The state registers and their dedicated counters are initialized writing a 0x100 to the L2IC_CR1[CC] field and then writing a 1 to the L2IC_CR1[CE] field (see **Section 11.8.2**). This command resets the debug counters to zero and loads the state registers with the information related to first way and index. Each read access to a state register increments by one the correlating counter and reloads the state registers with a new data. To enable cache debug mode, set the L2IC_CR2[CDM] bit. **Table 11-3**, **Table 11-4**, and **Table 11-5** are examples of debug sequences by accessing the state registers.

Table 11-3. Tag State Reading Sequence

Description	Debug Register
State initialization command	Initial load
Read tag array state register: way 0, index0, ETAG bits (in L2IC_B1)	re-load
Read tag array state register: way 0, index1, ETAG bits (in L2IC_B1)	re-load
Continue Reading from same address	
Read tag array state register: way 0, index 31, ETAG bits (in L2IC_B1)	re-load
Read tag array state register: way 1, index0, ETAG bits (in L2IC_B1)	re-load
Read tag array state register: way 1, index1, ETAG bits (in L2IC_B1)	re-load

Table 11-3. Tag State Reading Sequence (Continued)

Description	Debug Register
Continue Reading from same address	
Read tag array state register: way 1, index 31, ETAG bits (in L2IC_B1)	re-load
Continue Reading from same address	
Read tag array state register: way 8, index0, ETAG bits (in L2IC_B1)	re-load
Read tag array state register: way 8, index1, ETAG bits (in L2IC_B1)	re-load
Continue Reading from same address	
Read tag array state register: way 8, index 31, ETAG bits (in L2IC_B1)	re-load
Read tag array state register: way 0, index0, ETAG bits (in L2IC_B2)	re-load (way0,index0,L2IC_B2)
Read tag array state register: way 0, index1, ETAG bits (in L2IC_B2)	re-load
Continue Reading from same address	
Read tag array state register: way 0, index 31, ETAG bits (in L2IC_B2)	re-load
Read tag array state register: way 1, index0, ETAG bits (in L2IC_B2)	re-load
Read tag array state register: way 1, index1, ETAG bits (in L2IC_B2)	re-load
Continue Reading from same address	
Read tag array state register: way 1, index 31, ETAG bits (in L2IC_B2)	re-load
Continue Reading from same address	
Read tag array state register: way 8, index0, ETAG bits (in L2IC_B2)	re-load
Read tag array state register: way 8, index1, ETAG bits (in L2IC_B2)	re-load
Continue Reading from same address	
Read tag array state register: way 8, index 31, ETAG bits (in L2IC_B2)	re-load
Read tag array state register: way 0, index0, ETAG bits (in L2IC_B1)	re-load (way0,index0,L2IC_B1)
Continue Reading from same address	

Table 11-4. Line Replacement Mechanism State Reading Sequence

Description	Debug Register
State initialization command	Initial load
Read PLRU state register: index 0–index 3, PLRU bits (in L2IC_B1)	re-load
Read PLRU state register: index 4–index 7, PLRU bits (in L2IC_B1)	re-load
Read PLRU state register: index 8–index 11, PLRU bits (in L2IC_B1)	re-load
Read PLRU state register: index 12–index 15, PLRU bits (in L2IC_B1)	re-load
Read PLRU state register: index 16–index 19, PLRU bits (in L2IC_B1)	re-load
Read PLRU state register: index 20–index 23, PLRU bits (in L2IC_B1)	re-load
Read PLRU state register: index 24–index 27, PLRU bits (in L2IC_B1)	re-load
Read PLRU state register: index 28–index 31, PLRU bits (in L2IC_B1)	re-load
Read PLRU state register: index 0–index 3, PLRU bits (in L2IC_B2)	re-load
Read PLRU state register: index 4–index 7, PLRU bits (in L2IC_B2)	re-load

Table 11-4. Line Replacement Mechanism State Reading Sequence

Description	Debug Register
Read PLRU state register: index 8–index 11, PLRU bits (in L2IC_B2)	re-load
Read PLRU state register: index 12–index 15, PLRU bits (in L2IC_B2)	re-load
Read PLRU state register: index 16–index 19, PLRU bits (in L2IC_B2)	re-load
Read PLRU state register: index 20–index 23, PLRU bits (in L2IC_B2)	re-load
Read PLRU state register: index 24–index 27, PLRU bits (in L2IC_B2)	re-load
Read PLRU state register: index 28–index 31, PLRU bits (in L2IC_B2)	re-load
Read PLRU state register: index 0–index 3, PLRU bits (in L2IC_B1)	re-load
Continue Reading from same address	

Table 11-5. Valid State Reading Sequence

Description	Debug Register
State initialization command	Initial load
Read Valid state register: way 0, index0, valid bits (in L2IC_B1)	re-load
Read Valid state register: way 0, index1, valid bits (in L2IC_B1)	re-load
Continue Reading from same address	
Read Valid state register: way 0, index 31, valid bits (in L2IC_B1)	re-load
Read Valid state register: way 1, index0, valid bits (in L2IC_B1)	re-load
Read Valid state register: way 1, index1, valid bits (in L2IC_B1)	re-load
Continue Reading from same address	
Read Valid state register: way 1, index 31, valid bits (in L2IC_B1)	re-load
Continue Reading from same address	
Read Valid state register: way 8, index0, valid bits (in L2IC_B1)	re-load
Read Valid state register: way 8, index1, valid bits (in L2IC_B1)	re-load
Continue Reading from same address	
Read Valid state register: way 8, index 31, valid bits (in L2IC_B1)	re-load
Read Valid state register: way 0, index0, valid bits (in L2IC_B2)	re-load (way0,index0,L2IC_B2)
Read Valid state register: way 0, index1, valid bits (in L2IC_B2)	re-load
Continue Reading from same address	
Read Valid state register: way 0, index 31, valid bits (in L2IC_B2)	re-load
Read Valid state register: way 1, index0, valid bits (in L2IC_B2)	re-load
Read Valid state register: way 1, index1, valid bits (in L2IC_B2)	re-load
Continue Reading from same address	
Read Valid state register: way 1, index 31, valid bits (in L2IC_B2)	re-load
Continue Reading from same address	
Read Valid state register: way 8, index0, valid bits (in L2IC_B2)	re-load

Table 11-5. Valid State Reading Sequence (Continued)

Description	Debug Register
Read Valid state register: way 8, index1, valid bits (in L2IC_B2)	re-load
Continue Reading from same address	
Read Valid state register: way 8, index 31, valid bits (in L2IC_B2)	re-load
Read Valid state register: way 0, index0, valid bits (in L2IC_B1)	re-load (way0,index0,L2IC_B1)
Continue Reading from same address	
Note: L2IC_B1 and L2IC_B2 designate the two cache memory modules used by the L2 Cache to interleave the cached memory.	

11.4.7.2 L2 ICache Array Access During Debug

During debug mode, cache memory modules are accessed by programming L2IC_DBGACS and L2IC_DBGDATA. Cache update mechanisms (fetch, line replacement mechanism) are disabled in this mode.

Use the following steps to read an array:

1. Configure the debug read access by writing the appropriate values to the fields in L2IC_DBG_ACS (see page 11-33 for details):
   - Way selection to the DW field in bits 18–16.
   - Address lsbs to the DBGAD field in bits 13–0.
   - Access size in bytes to the DABE field in bits 21–20.
   - Select read by writing a 0 to the WR field in bit 23.
   - Initiate the access by writing a 1 to the INIT field in bit 24.
2. Read the L2IC_DBG_DATA content. The cache array content should be updated with a new read value (shifted to the right).
3. Data in bytes not selected by the byte enable field are not be driven by memory and therefore are not valid.
4. After the access is complete, hardware clears the L2IC_DBG_ACS[INIT] bit to enable a new access.

Use the following steps to write to an array:

1. Write the data to be written to L2IC_DBG_DATA (up to 32 bits, shifted to the right).
2. Configure the debug write access by writing the appropriate values to the fields in L2IC_DBG_ACS (see page 11-33 for details):
   - Way selection to the DW field in bits 18–16.
   - Address lsbs to the DBGAD field in bits 13–0.
   - Access size in bytes to the DABE field in bits 21–20.

- Select write by writing a 1 to the WR field in bit 23.
- Initiate the access by writing a 1 to the INIT field in bit 24.

3. Data in bytes not selected by the byte enable field are discarded (write is not performed).
4. After the access is complete, hardware clears the L2IC_DBG_ACS[INIT] bit to enable a new access.

11.5 M2 Memory

Figure 11-4 shows the M2 memory architecture. The 512 KB memory performs 128-bit wide accesses at 400 MHz and is accessible to any of the DSP core subsystems or any other initiator in the system (L2 ICache, DMA controller, TDM, QUICC Engine module subsystem, serial RapidIO subsystem, PCI). The memory is partitioned into four 128 KB memory groups to allow four simultaneous accesses. The group addresses are interleaved with 256-byte interleave resolution which is optimized to minimize contentions between accesses. The M2 memory uses SRAM technology and supports both single and burst accesses with a single wait state. The M2 memory is fully ECC protected. The M2 memory supports partial accesses. Automatic read-modify-write accesses are generated to maintain ECC protection. The M2 memory is volatile after reset.

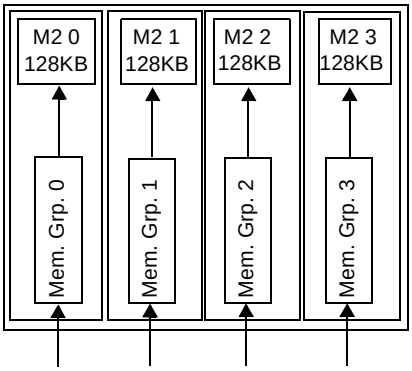


Figure 11-4. M2 Memory Architecture

The 512 KB M2 memory contains four interleaved banks of 128 KB each, operating at the system frequency supporting 128 bits data bus width. The M2 is a unified memory that stores both data and program code. All of the DSP cores, as well as the DMA, PCI, TDM, RapidIO, L2 ICache, and the QUICC Engine subsystem, can access the M2 memory through the systems interconnect. The four M2 banks can be accessed simultaneously by four different initiators to accommodate the four cores. The interleaving feature improves the total system performance by reducing the probability of transaction collisions. The bank interleaving resolution is 256 bytes which is optimal for the DSP cores.

To reduce possible impact of soft error rate (SER) on systems using the MSC8144E, the M2 employs error correction code (ECC). The ECC being deployed adds 7 bits of ECC on each 64

bits of memory. The ECC is calculated for every write access. Reads from the memory use the ECC to correct the data. The ECC mechanism of the M2 can detect and correct a single error, and in that case, generate an interrupt toward the DSP cores.

The reservation atomic operation (**bmtset** instruction) is performed in two stages: a read of a certain address content and a write of the modified content back to the original address. Between these two accesses the atomic operation is defined as *open*. A write access by another DSP core subsystem or external hosts to the same address causes the atomic operation to fail, and the DSP core subsystem atomic write operation to M2 memory is not performed. The systems arbiter allows one such open atomic operation at a time. Other cores requesting an atomic operation are serviced only after the current atomic operation is closed.

Note: Atomic operations are only supported in M2 memory and not in any other memory.

11.6 M3 Memory

The 10 MB M3 memory can be used for both program and data and eliminates the need for an external memory in a variety of applications, thus reducing board space, power dissipation, and cost. The M3 memory has a 128-bit wide port and runs at 400 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. The M3 memory is fully ECC protected.

The M3 memory uses a novel embedded DRAM (eDRAM) of 20 macrocells of 4 Mb each to total 80 Mb of user memory with a bidirectional data bus of 128-bit. The memory is located between addresses 0xD0000000 and 0xD09FFFFF in the MSC8144E memory map. The eDRAM memories operate at 100 MHz frequency and there is an internal SRAM controller that enables access with bursts of 4×128-bit to generate a sustained bandwidth of 128-bit at 400 MHz between the M3 memory and the rest of the MSC8144E device.

As a DRAM, the memory must be refreshed. The whole refresh operation occurs in parallel to regular accesses and is almost completely hidden from the user. The time is partitioned to time slots of 40 cycles. For each time slot, only one row is refreshed on every 2 MB section. The refresh is delayed until there is a user access in this same time slot and then the refresh is performed in parallel to the user access and on a different port. If the line to be refreshed in this time slot is in a different 128 KB sub-module than the user access, the actions are performed in parallel. If they are to the same 128 KB sub-module, the refresh is delayed until there is another user access in the same time slot. If there is no user access during the time slot, the controller forces a refresh condition. If a user access is requested during the refresh period, it is delayed by up to 4 cycles. The result is that the access to the M3 memory is not deterministic and a refresh conflict can increase the latency by up to 4 cycles.

The M3 memory features include:

- Memory size: 10 MB
- Maximum bandwidth: 6.4 GB/s
- ECC support
- Access types:
  - Read aligned burst of 4 beats of 128-bit each (64 bytes). The data returns to the bus as an aligned burst: D0, D1, D2, D3.
  - Read non-aligned burst of 4 beats of 128-bit each. Assuming that the read is to address 0xD0000.0010, then the data returns to the bus as: D1, D2, D3, D0 (critical word first).
  - Write aligned burst of 4 beats of 128-bit each.
  - Read single of 16 bytes.
  - Write single of 16 bytes.
  - Read partial of between 1 and 15 bytes aligned to a 16-byte boundary.
  - Write partial of between 1 and 15 bytes aligned to a 16-byte boundary. In this case, the M3 controller performs a read-modify-write sequence to update the ECC bits in the memory.

The CLASS in the MSC8144E is responsible for making sure that any access that is different from the above is sliced to a few accesses each of which complies with the above requirements.

The eDRAM memory can fix faulty bits by using both redundancy columns and ECC bits. There are two redundancy columns in each 4 Mb macrocell and the number of columns needing replacement is determined internally. In addition, there are 8 ECC bits per 128 bits in the whole memory. Using its ECC equation, the controller can detect and correct one error in each 128-bit group. The memory is initialized by an initialization sequence performed right after reset deassertion. The initialization sequence lasts up to 0.5 ms. During this period, the MSC8144E can issue accesses to the memory, but the controller keeps the access open and reports the bus as busy.

Note: Atomic operations are only supported in M2 memory and not in any other memory.

11.7 Internal Boot ROM

The MSC8144E 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.

11.8 Programming Model

Note: The are part of the MSC8144 SC3400 DSP core subsystem. For detailed programming information for the MMU, L1 ICache, and L1 DCache, refer to the *MSC8144 SC3400 DSP Core Reference Manual*

The L2 ICache registers include:

Note: The following L2 ICache registers use a base address of 0xFFF2C000.

- L2 ICache Control Register 0 (L2IC_CR0), see page 11-26.
- L2 ICache Control Register 1 (L2IC_CR1), see page 11-26.
- L2 ICache Control Register 2 (L2IC_CR2), see page 11-28.
- L2 ICache LRM State Register (L2IC_LRM), see page 11-29.
- L2 ICache TAG State Register (L2IC_TAG), see page 11-31.
- L2 ICache Valid State Register (L2IC_VALID), see page 11-32.
- L2 ICache Data Register (L2IC_DBG_DATA), see page 11-32.
- L2 ICache Debug Access Register (L2IC_DBG_ACS), see page 11-33.

Note: The following three registers use a base address of 0xFFF2A000.

- L2 ICache Cacheable Area Start Address (L2IC_CSA), see page 11-34.
- L2 ICache Cacheable Area End Address (L2IC_CEA), see page 11-35.
- L2 ICache Cacheable Area Enable (L2IC_CEN), see page 11-36.

11.8.1 L2 ICache Control Register 0 (L2IC_CR0)

L2IC_CR0 L2 ICache Control Register 0 Offset 0x00

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

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

Table 11-6 defines the L2IC_CR0 bit fields.

Table 11-6. L2IC_CR0 Bit Descriptions

Name	Reset	Description	Settings
SSPA 31–8	0	**Sweep Start Physical Address** This field indicates the start address of the memory space that correlates to the sweep operation as it appears on msb part of the address bus.	
— 7–0	0	Reserved. Write to zero for future compatibility.	

11.8.2 L2 ICache Control Register 1 (L2IC_CR1)

L2IC_CR1 L2 ICache Control Register 1 Offset 0x04

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SEPA								—	INV	CGS	—	CC			CCE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-7 defines the L2IC_CR1 bit fields.

Table 11-7. L2IC_CR1 Bit Descriptions

Name	Reset	Description	Settings
SEPA 31–8	0	**Sweep End Physical Address** Describes the end address of the memory space that correlates with the sweep operation, as it appears on msb part of the address bus. When programming this field, you must make sure that it is greater than or equal to the sweep start address (L2IC_CR0[SSPA]).	
— 7	0	Reserved. Write to zero for future compatibility.	

Table 11-7. L2IC_CR1 Bit Descriptions (Continued)

Name	Reset	Description	Settings
INV 6	0	**PLRU Mechanism Inverse Bit** Inverts all PLRU status bits of all indexes. The bit is reset by hardware one cycle after it is asserted by software. This bit is ignored while the ICache memory is disabled. In Debug mode, you can write a 1 to the bit, but it is ignored.	0 Normal PRLU status bits. 1 Inverted PLRU status bits.
CGS 5	0	**Cache Global Sweep** Determines whether a sweep command in the CC field is performed on all cache lines or only on lines that correlate with the Sweep Start Address and Sweep End Address. Cache Global Sweep for the Invalidate command (see description of DICCR1 below) is a special command that executes in parallel without performing a serial sweep. If the bit is set, the sweep command ignores the programmed sweep address space and address comparators and performs the sweep on all cache lines.	0 Sweep command is performed only on lines that correlate with Sweep Start Address and Sweep End Address. 1 Sweep command is global one and thus is performed on all cache lines.
— 4	0	Reserved. Write to zero for future compatibility.	
CC 3–1	0	**Cache Command** Indicates the cache command. Sweep commands apply only if the Cache Command Enable CE bit is set. Writing a Cache Command without setting Cache Command Enable bit has no effect.	000 Reserved 001 Invalidate Sweep Command 010 Reserved 011 Reserved 100 Initialize State Registers 101 Reserved 110 Reserved 111 Reserved
CCE 0	0	**Cache Command Enable** Writing a 1 to this bit starts the operation specified by the Cache Command field using the relevant characteristics derived by the rest of the cache register bits. As long as cache operation continues, this bit is a 1. When the cache operation completes, this bit is cleared automatically by the hardware. Any attempt to start a new cache sweep operation before a previous sweep operation is complete and CCE bit is cleared is ignored, and therefore an initiation of a new sweep. Always poll this bit and set it when it is cleared to start a new operation. Sweep operation parameters are sampled at the beginning of the operation and a change of parameter' values has no effect during the operation. Any attempt to initiate a cache command is ignored while Cache Enable Bit is clear. Any attempt to initialize a sweep command is ignored if cache debug mode is enabled. Any attempt to initialize state registers is ignored when cache debug mode is disabled.	0 Cache command disabled. 1 Cache command enabled.

11.8.3 L2 ICache Control Register 2 (L2IC_CR2)

L2IC_CR2 L2 ICache Control Register 2 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
	—		BS			LB			PFS	—			PROF	CDM	CGL	CE
Type	R/W															
Reset	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0

Table 11-9 defines the IC_CR2 bit fields.

Table 11-8. L2IC_CR2 Bit Descriptions

Name	Reset	Description	Settings
— 31–14	0	Reserved. Write to zero for future compatibility.	
BS 13–11	100	**Burst Size** Defines the burst size from L2 ICache toward the system	100 4 VBRs (default). 111 1 VBR. all others reserved.
LB 10–8	111	**Cache Way Boundaries Lock** The value of this field defines directly the lower and upper boundaries of the cache that are locked (or open),	000 reserved 001 0,1 010 2,3 011 4,5 100 6,7 101 0,1,2,3 110 4,5,6,7 111 0,1,2,3,4,5,6,7
PFS 7	1	**Prefetch Select** Enables/disables the prefetch operation.	0 Prefetch disabled. 1 Prefetch selected.
— 6–4	0	Reserved. Write to zero for future compatibility.	
PROF 3	0	**Profiling Enable** Determines whether to generate profiling signals.	0 No profiling signals generated. 1 Generate profiling signals.
CDM 2	0	**Cache Debug Mode** Indicates whether cache is in debug mode or not. During debug mode, update mechanisms are disabled and debug registers are accessible. Cache memory is accessible through cache debug registers. An attempt to set this bit while sweep operation is not complete is not allowed.	0 Normal cache mode. 1 Cache debug mode.

Table 11-8. L2IC_CR2 Bit Descriptions (Continued)

Name	Reset	Description	Settings
CGL 1	0	**Cache Global Lock** Indicates whether cache is in global lock mode or not. Assertion of global lock mode is ignored during a cache sweep operation.	0 Cache global lock mode not active. 1 Cache global lock mode active.
CE 0	0	**Cache Memory Enabled** Indicates whether the two L2 cache memory modules are enabled or disabled. At reset deassertion, the two modules are disabled. Once the cache memory is enabled. it may be disabled only by reset. In disable mode the clock inside each memory module is disabled and power is saved. **Note:** This is a sticky bit.	0 Cache memory disabled. 1 Cache memory enabled.

11.8.4 L2 ICache LRM State Register (L2IC_LRM)

L2IC_LRM L2 ICache LRM State Register Offset 0x0C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—	PLRUB3							—	PLRUB2						
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
	—	PLRUB1							—	PLRUB0						
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register holds the state of the Line Replacement Mechanism (LRM) of the cache, which is PLRU in this implementation. The PLRUB[3–0] fields hold the PLRU state of the cache lines pointed to by the debug counter. There are seven PLRU bits for each index in the cache to determine the line to be replaced. The PLRU bits are updated when a new line is allocated or replaced and when there is a line hit. A line is selected for replacement according to the PLRU bit encoding shown in **Table 11-10** (initial values of B[6–0] is all zeros). This register is read only, accessible only in L2 ICache debug mode. **Table 11-9** defines the L2IC_LRM bit fields.

Table 11-9. L2IC_LRM Bit Descriptions

Name	Reset	Description	Settings
— 31	0	Reserved. Write to zero for future compatibility.	
PLRUB3 30–24	0	**PLRU Bits 3** Holds the state bits for indexes (4 × i) + 3 (i is an integer).	0 Cache line not valid. 1 Cache line valid.
— 23	0	Reserved. Write to zero for future compatibility.	
PLRUB2 22–16	0	**PLRU Bits 2** Holds the state] bits for indexes (4 × i) + 2 (i is an integer).	0 Cache line not valid. 1 Cache line valid.

Table 11-9. L2IC_LRM Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 23	0	Reserved. Write to zero for future compatibility.	
PLRUB1 22–16	0	**PLRU Bits 1** Holds the state] bits for indexes (4 × i) + 1 (i is an integer).	0 Cache line not valid. 1 Cache line valid.
— 7	0	Reserved. Write to zero for future compatibility.	
PLRUB0 6–0	0	**PLRU Bits 0** Holds the state] bits for indexes (4 × i) (i is an integer).	0 Cache line not valid. 1 Cache line valid.

Table 11-10. PLRU Replacement Way Selection

PLRU Bits						Way Selected for Replacement
B0	0	**B1**	0	**B3**	0	L0
	0		0		1	L1
	0		1	**B4**	0	L2
	0		1		1	L3
	1	**B2**	0	**B5**	0	L4
	1		0		1	L5
	1		1	**B6**	0	L6
	1		1		1	L7

11.8.5 L2 ICache Tag State Register (L2IC_TAG)

L2IC_TAG L2 ICache Tag State Register Offset 0x10

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

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

This register is accessible only in L2 ICache debug mode. **Table 11-11** defines the L2IC_TAG bit fields.

Table 11-11. L2IC_TAG Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0xFF	Reserved. Write to ones for future compatibility.	
TVB 23	0	**Tag Valid Bit** Indicates whether the whole cache line is valid or not.	0 Cache line not valid. 1 Cache line valid.
— 22–18	0	Reserved. Write to zero for future compatibility.	
TAG 17–0	0x3FFFF	**Tag** Indicates directly the status of the debugged TAG according to a dedicated debug counter.	

11.8.6 L2 ICache Valid State Register (L2IC_VALID)

L2IC_VALID L2 ICache Valid State Register Offset 0x14

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
	VALID															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register is accessible only in L2 ICache debug mode. **Table 11-12** defines the L2IC_VALID bit fields.

Table 11-12. L2IC_VALID Bit Descriptions

Name	Reset	Description	Settings
— 31–16	0	Reserved. Write to zero for future compatibility.	
VALID 15–0	0	**VBR Valid** This field indicates the status of the valid bits of each of the reflected VBRs. The reflected VBRs are determined by a dedicated counter.	0 VBR not valid. 1 VBR valid.

11.8.7 L2 ICache Debug Data Register (L2IC_DBG_DATA)

L2IC_DBG_DATA L2 ICache Debug Data Register Offset 0x18

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

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

This register is accessible only in L2 ICache debug mode. **Table 11-13** defines the L2IC_DBG_DATA bit fields.

Table 11-13. L2IC_DBG_DATA Bit Descriptions

Name	Reset	Description	Settings
DDATA 31–0	0	**Debug Data** This field contains the read/ write data in debug mode.	

11.8.8 L2 ICache Debug Access Register (L2IC_DBG_ACS)

L2IC_DBG_ACS L2 ICache Debug Access Register Offset 0x1C

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

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

This register is accessible only in L2 ICache debug mode. **Table 11-14** defines the L2IC_DBG_ACS bit fields.

Table 11-14. L2IC_DBG_ACS Bit Descriptions

Name	Reset	Description	Settings
— 31–25	0	Reserved. Write to zero for future compatibility.	
INIT 24	0	**Debug Access Initiation** Initiates a read or write access to the cache array in debug mode. The data should be valid before setting this bit. The bit is ignored except in debug mode. The bit clears on hardware cycle after it is set by software.	0 No access. 1 Initiate access.
RW 23	0	**Read/Write** Indicates whether the access is a read or a write.	0 Read. 1 Write.
— 22	0	Reserved. Write to zero for future compatibility.	
DABE **21–20**	0	**Debug Access Byte Enable** Defines the access size.	00 Byte. 01 2 bytes. 10 4 bytes. 11 reserved.
— 19	0	Reserved. Write to zero for future compatibility.	
DW 18–16	0	**Debug Way** Defines the way number of the accessed data	000 Way 0. 001 Way 1. 010 Way 2. 011 Way 3. 100 Way 4. 101 Way 5. 110 Way 6. 111 Way 7.

Table 11-14. L2IC_DBG_ACS Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 15–14	0	Reserved. Write to zero for future compatibility.	
DBGAD 13–0	0	**Debug Access Address** This field contains the access address in debug mode (extended up to 64 KB) as bits 13–0 on the input address bus, using the following format: • Index: bits 13–8 • VBR offset: bits 7–4 • Long (32-bit) offset within the VBR: bits 3–2 • Byte offset within the long: bits 1–0 To make an aligned access to the DBG_ACS_SIZE, use only the following options: • For DBG_ACS_SIZE = 0b10 (long), the byte offset must = 0b00 only. • For DBG_ACS_SIZE = 0b01 (word), the byte offset can be 0b00 or 0b10. • For DBG_ACS_SIZE = 0b00 (byte), the byte offset can be 0b00, 0b01, 0b10, or 0b11.	

11.8.9 L2 ICache Cacheable Area Start Address (L2IC_CSA)

L2IC_CSA L2 ICache Cacheable Area Start Address Offset 0xC04

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

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

L2IC_CSA configures the start address of the cacheable area in the L2 ICache. To assure proper operation, do not change this register while the L2IC_CEN[DEN] bit is set. This register is reset by a soft reset. **Table 11-15** defines the L2IC_CSA bit fields.

Table 11-15. L2IC_CSA Bit Descriptions

Name	Reset	Description	Settings
— 31–20	0	Reserved. Write to zero for future compatibility.	
SA 19–0	0	**Start Address** Contains the 20 msbs of the L2 ICache cacheable window start address (bits 31–12). The 12 lsbs are all zeros.	

11.8.10 L2 ICache Cacheable Area End Address (L2IC_CEA)

L2IC_CEA L2 ICache Cacheable Area End Address Offset 0xC44

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

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

L2IC_CEA configures the end address of the cacheable area in the L2 ICache. To assure proper operation, do not change this register while the L2IC_CEN[DEN] bit is set. This register is reset by a soft reset. **Table 11-16** defines the L2IC_CEA bit fields.

Table 11-16. L2IC_CEA Bit Descriptions

Name	Reset	Description	Settings
— 31–20	0	Reserved. Write to zero for future compatibility.	
EA 19–0	oxFFFFF	**End Address** Contains the 20 msbs of the L2 ICache cacheable window end address. The 12 lsbs are all zeros. When programming this value, make sure it is greater than or equal to the value stored in L2IC_CSA. If the values are equal, the address window is 4 KB in size.	

11.8.11 L2 ICache Cacheable Area Enable (L2IC_CEN)

L2IC_CEN L2 ICache Cacheable Area Enable 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
	—															DEN
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

L2IC_CEN controls the activation of the L2 ICache cacheable window and therefore controls the cacheable/non-cacheable mode of the L2 ICache. When the window is not active, all transactions use the non-cacheable path even for accesses considered cacheable in the MMU. However, any transaction acknowledged before the DEN bit is cleared continues normally until completion. To assure proper operation, do not enable the L2 ICache cacheable area until after the cacheable window is defined in L2IC_CSA and L2IC_CEA. This register is reset by a soft reset. **Table 11-17** defines the L2IC_CEN bit fields.

Table 11-17. L2IC_CEN Bit Descriptions

Name	Reset	Description	Settings
— 31–1	0	Reserved. Write to zero for future compatibility.	
DEN 0	0	Cacheable Window Disable/Enable Activates/deactivates the cacheable window.	0 Window disabled (all accesses non-cacheable). 1 Window enabled.

11.9 L2 ICache Programming Limitations

The L2 ICache programming has the following limitations:

- Register access width is restricted to the actual register width. Register access attempts with an incorrect access width may cause data corruption.
- Accesses to debug-related registers: L2IC_DBGACS, L2IC_DBGDATA, L2IC_TAG, L2IC_LRM and L2IC_VALID have meaning in Cache Debug mode only.
- Initiation of sweep command by writing to L2IC_CR1, should be done according to the routine that is described in **Section 11.4.6.6**.
- To configure each of the L2 ICache CLASS registers, please refer to **Section 11.8.9** to **Section 11.8.11**.
- To configure the TL1IC in each of the two L2IC_Bs use the MBus interface to the L2RB (register file block of the L2 ICache).
- Every access to L2 ICache should start and end in the same cache line so the CLASS can handle the interleave between the L2 instruction cache banks.
- Accesses from the DSP core subsystem toward memory through the L2 ICache must be of one VBR or with a burst of 4 VBRs (byte count of 64). You must program this limitation into the MMU in the DSP core subsystem to ensure this operation.
- Accesses from the DSP core subsystem ICache toward memory via the non-cacheable route through the L2 ICache are restricted to a burst size of 1 beat or 4 beats (2-beat or 8-beat accesses are not permitted).
- Accesses from the DSP core subsystem DCache toward memory are restricted to a burst size of 1 beat or 4 beats (2-beat or 8-beat accesses are not permitted).
- L2 ICache does not support nested sweeps. Avoid it by using a dedicated semaphore and polling the relevant L2 ICache Control Register bit (see **Section 11.8.2**).
- ECC is not supported during debug mode and to L2 ICache addresses written during debug.

DDR SDRAM Memory Controller 12

The fully programmable DDR SDRAM memory controller supports most JEDEC-standard ×8 or ×16 DDR1 and DDR2 memories. However, mixing different memory types in the same system is not supported. Built-in error checking and correction (ECC) ensures very low bit-error rates for reliable high-frequency operation. Dynamic power management and auto-precharge modes simplify memory system design. A large set of special features, including ECC error injection, support rapid system debug. **Figure 12-1** shows a high-level view of the DDR memory controller with its associated interfaces.

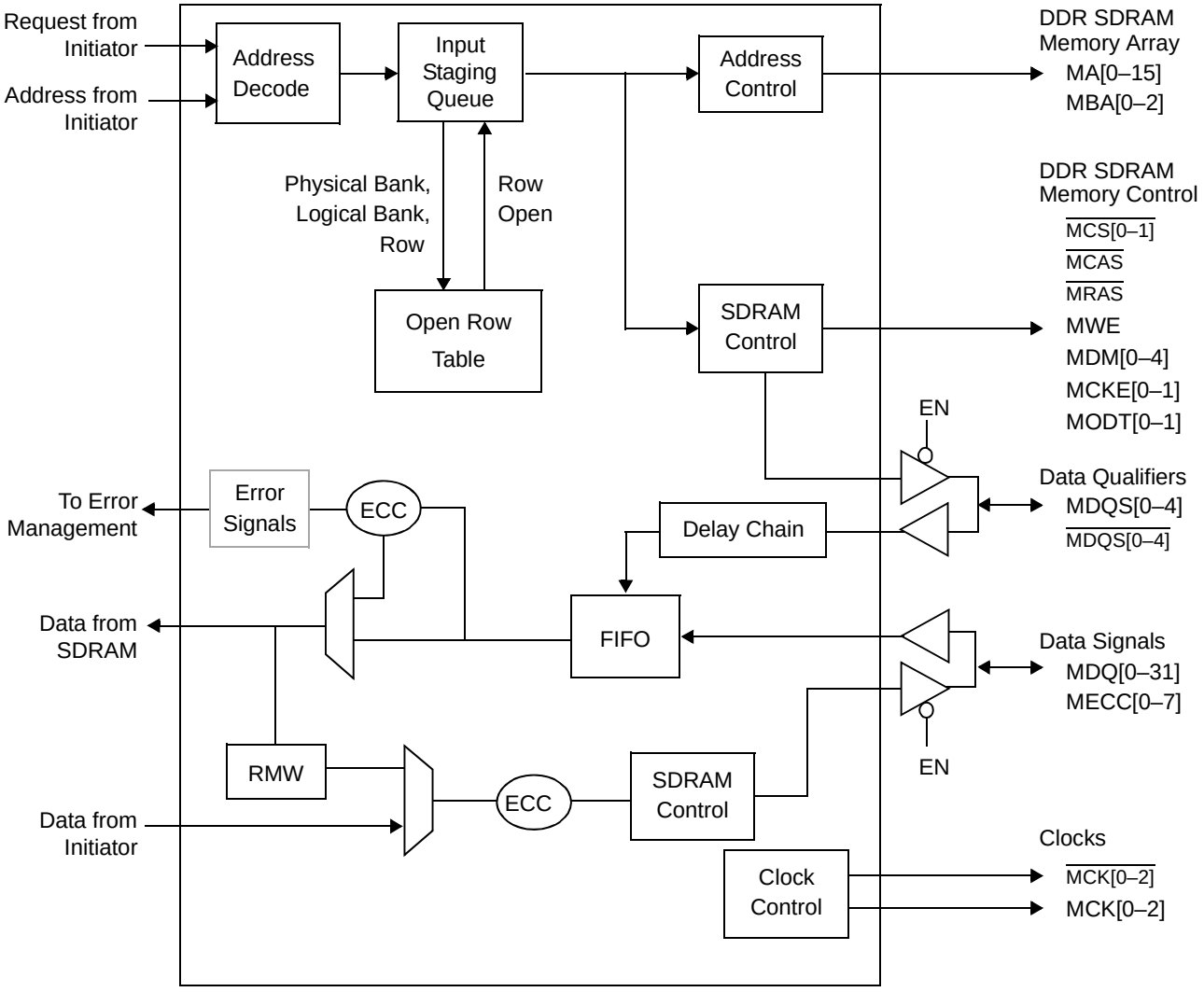


Figure 12-1. DDR SDRAM Memory Controller Block Diagram

12.1 Architecture

The DDR SDRAM controller controls processor and I/O interactions with system memory. It supports JEDEC-compliant DDR1 and DDR2 memories.

Note: A bank is a physical bank specified by a chip select; a logical bank is one of the four or eight sub-banks in each SDRAM chip. A sub-bank is specified by the two or three bits on the memory bank address (MBA) pins during a memory access. The memory interface supports two physical banks of 32/40-bit words or 16/24-bit wide memory.

As shown in **Figure 12-1**, requests are received from the internal mastering device, and the address is decoded to generate the physical bank, logical bank, row, and column addresses. The transaction is then loaded into the input staging queue with the decoded information. The lower two entries of the input queue are compared with values in the row open table to determine whether the address maps to an open page. If the address from either entry does not map to an open page, an activate command is issued for that entry, with the lowest entry having priority. Commands are always issued from the lowest input queue entry.

Programmable parameters give a variety of memory organizations and timings. Using optional error checking and correcting (ECC) protection, the DDR memory controller detects and corrects all single-bit errors, detects all double-bit errors within the 32-bit or 16-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 programmed via the DDR_SDRAM_INTERVAL[BSTOPRE] bit (see **Table 12-27** on page 12-49).

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

The data interface is source synchronous, so the source of the data provides a clocking signal to synchronize data reception. These bidirectional data strobes (MDQS[0–4]) are inputs to the controller during reads and outputs during writes. The DDR SDRAM specification requires the data strobe signals to be centered within the data tenure during writes and to be offset by the controller to the center of the data tenure during reads. These delays are implemented by the DDR SDRAM memory controller for both reads and writes. The address and command interface is also source synchronous, although 1/8 cycle adjustments are provided for adjusting the clock alignment. 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.

Figure 12-2 shows an example DDR SDRAM configuration with four logical banks.

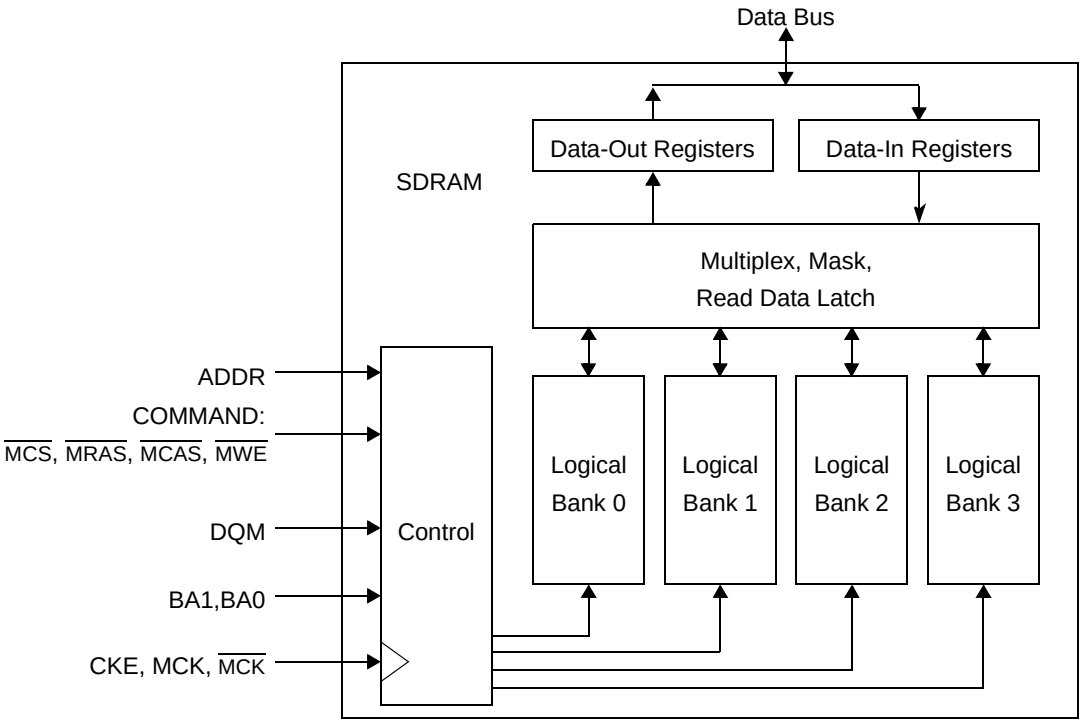


Figure 12-2. Typical Dual Data Rate SDRAM Internal Organization

Figure 12-3 shows some typical signal connections.

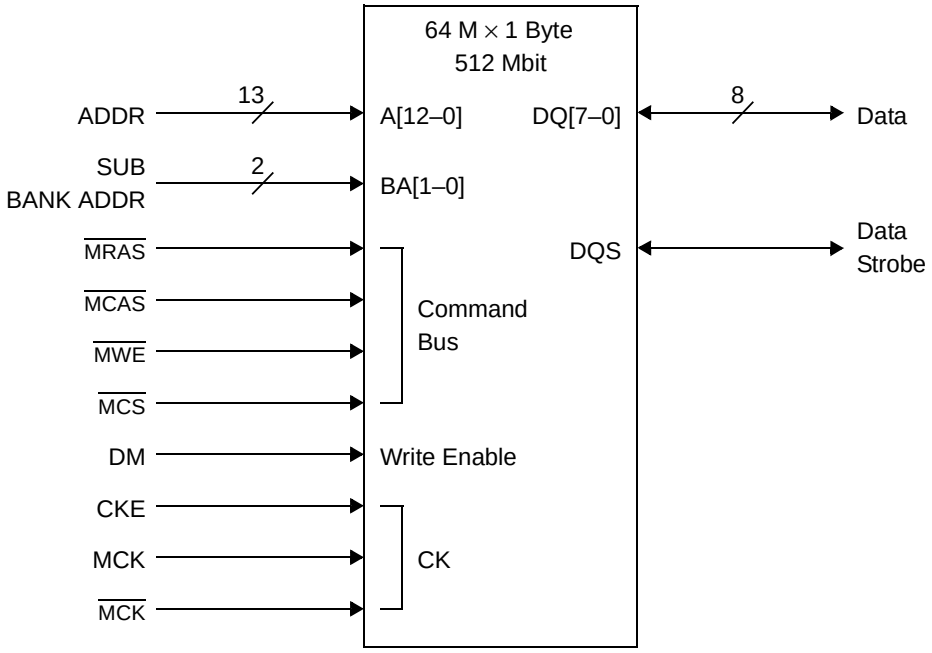


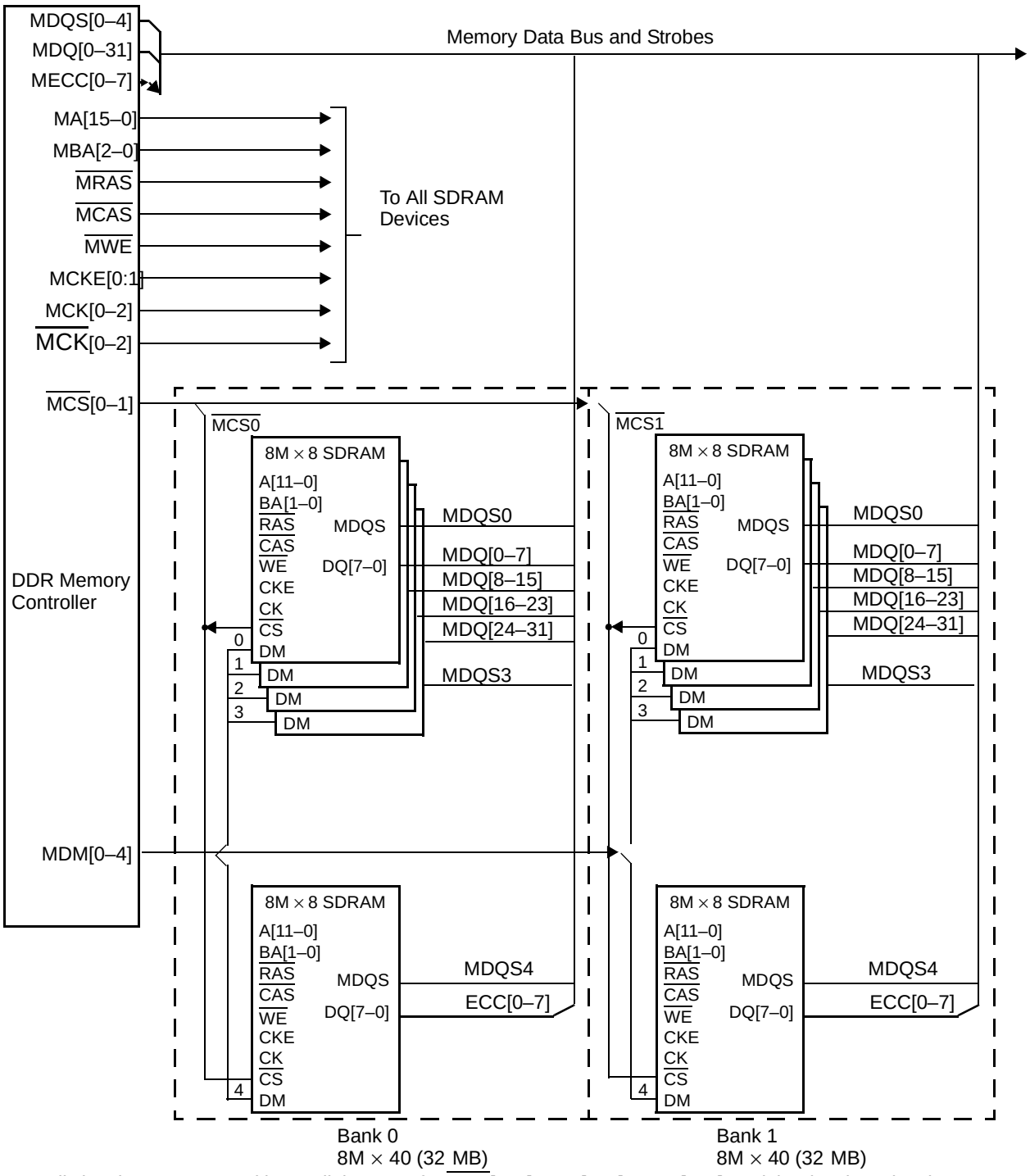
Figure 12-3. Typical DDR SDRAM Interface Signals

Figure 12-4 shows an example DDR SDRAM configuration with two physical banks, each containing five 8 M × 8 DDR modules for a total of 64 MB of system memory. One of the modules is used for the memory ECC checking function. The AC timing specifications, desired memory operating frequency, capacitive loads, and board routing loads can assist the system designer in defining signal buffering requirements. The DDR memory controller drives 16 address pins, but in this example the DDR SDRAM devices use only 13 bits.

12.1.1 DDR SDRAM Interface Operation

The DDR memory controller supports many different DDR SDRAM configurations. Sixteen multiplexed address signals and three logical bank select signals support device densities from 64 Mb to 4 Gb. The DDR SDRAM physical banks can be built from directly-attached memory devices. The data path to individual physical banks is 32 or 16 bits wide (40 or 24 bits with ECC). The DDR memory controller supports physical bank sizes from 16 MB to 1 GB. The physical banks can be constructed using × 8 or ×16 DDR1 or DDR2 memory devices. Five data qualifier (DQM including DQM for ECC) 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]).



1. All signals are connected in parallel, except for MCS[0–1],MCK[0–2], MDM[0–4], and the data bus signals.
2. Each MCS[0–1] signal corresponds with a separate physical bank of memory.
3. MCK[0–2] can be apportioned among all memory devices. Complementary bus is not shown.

Figure 12-4. Example 64 MB DDR SDRAM Configuration With ECC

When ECC is enabled, all memory accesses are performed on 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. **Table 12-1** and **Table 12-2** show how DDR SDRAM memories are used with x8 or x16 devices for 32 or 16 bit memory interfaces.

Table 12-1. Byte Lane to Data Relationship for a 32-Bit Memory Interface

Data Byte Lane	Data Bus Mask	Data Bus Strobe	Data Bus 32-Bit Mode
0 (MSB)	MDM0	MDQS0	MDQ[0–7]
1	MDM1	MDQS1	MDQ[8–15]
2	MDM2	MDQS2	MDQ[16–23]
3 (LSB)	MDM3	MDQS3	MDQ[24–31]

Table 12-2. Byte Lane to Data Relationship16-Bit Memory Interface

Data Byte Lane	Data Bus Mask	Data Bus Strobe	Data Bus 16-Bit Mode
0 (MSB)	MDM0	MDQS0	MDQ[0–7]
1 (LSB)	MDM1	MDQS1	MDQ[8–15]

12.1.2 DDR SDRAM Organization

Although the DDR memory controller multiplexes row and column address bits onto 16 memory address signals and 3 logical bank select signals, individual physical banks can contain memory devices with fewer than 31 address bits. Each physical bank can be individually configured to provide from 12 to 16 row address bits plus 2 or 3 logical bank-select bits and from 8–11 column address bits. **Table 12-3** lists the DDR SDRAM device configurations supported by the DDR memory controller.

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

Table 12-3. DDR SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column x Sub-bank Bits	32-Bit Bank Size	Two Banks of Memory
64 Mb	8 Mb × 8	12 × 9 × 2	32 MB	64 MB
64 Mb	4 Mb × 16	12 × 8 × 2	16 MB	32 MB
128 Mb	16 Mb × 8	12 × 10 × 2	64 MB	128 MB
128 Mb	8 Mb × 16	12 × 9 × 2	32 MB	64 MB
256 Mb	32 Mb × 8	13 × 10 × 2	128 MB	256 MB
256 Mb	16 Mb × 16	13 × 9 × 2	64 MB	128 MB
512 Mb	64 Mb × 8	13 × 11 × 2	256 MB	512 MB
512 Mb	32 Mb × 16	13 × 10 × 2	128 MB	256 MB
1 Gb	128 Mb × 8	14 × 11 × 2	512 MB	1 GB
1 Gb	64 Mb × 16	14 × 10 × 2	256 MB	512 MB
2 Gb	256 Mb × 8	15 × 11 × 2	1 GB	2 GB
2 Gb	128 Mb × 16	15 × 10 × 2	512 MB	1 GB
4 Gb	512 Mb × 8	16 × 11 × 2	2 GB	4 GB
4 Gb	256 Mb × 16	16 × 10 × 2	512 MB	2 GB

Table 12-4. Supported DDR2 Device Configurations

SDRAM Device	Device Configuration	Row × Column × Sub-bank Bits	32-Bit Bank Size	Two Banks of Memory
256 Mb	32 Mb × 8	13 × 10 × 2	128 MB	256 MB
256 Mb	16 Mb × 16	13 × 9 × 2	64 MB	128 MB
512 Mb	64 Mb × 8	14 × 10 × 2	256 MB	512 MB
512 Mb	32 Mb × 16	13 × 10 × 2	128 MB	256 MB
1 Gb	128 Mb × 8	14 × 10 × 3	512 MB	1 GB
1 Gb	64 Mb × 16	13 × 10 × 3	256 MB	512 MB
2 Gb	256 Mb × 8	15 × 10 × 3	1 GB	2 GB
2 Gb	128Mb × 16	14 × 10 × 3	512 MB	1 GB
4 Gb	512 Mb × 8	16 × 10 × 3	2 GB	4 GB
4 Gb	256Mb × 16	15 × 10 × 3	1 GB	2 GB

If a transaction request is issued to the DDR memory controller and the address does not lie within any of the programmed address ranges for an enabled chip select, a memory select error is flagged (see **Section 12.5**, *Error Management*, on page 12-24).

If the starting and ending address of a disabled bank overlaps with the address space of an enabled bank, system memory in the overlapping address range may be corrupted. The starting and ending addresses of unused memory banks should be mapped to unused memory space.

Using a memory-polling algorithm at power-on reset, system firmware configures the memory-boundary registers to map the size of each bank in memory. The memory controller uses its bank map to assert the appropriate $\overline{\text{MCSx}}$ signal for memory accesses according to the bank starting and ending addresses. The memory banks do not have to be mapped to a contiguous address space.

12.1.3 DDR SDRAM Address Multiplexing

Table 12-5 and **Table 12-6** show the address bit encodings for each DDR SDRAM configuration. The address at the memory controller signals MA[15–0] use MA15 as the MSB and MA0 as the LSB. Also, MA10 is the auto-precharge bit in DDR1/DDR2 modes for reads and writes, so the column address can never use MA10.

Table 12-5. DDR1 Address Multiplexing for 32-bit Data Bus

Row x Col		MSB		Address from Core Initiator																								LSB
		30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1–0
16 x 11 x 2	$\overline{MRAS}$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	$\overline{MBA}$																	1	0												
	$\overline{MCAS}$																			11	9	8	7	6	5	4	3	2	1	0	
16 x 10 x 2	$\overline{MRAS}$		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	$\overline{MBA}$																		1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
15 x 11 x 2	$\overline{MRAS}$		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	$\overline{MBA}$																	1	0												
	$\overline{MCAS}$																			11	9	8	7	6	5	4	3	2	1	0	
15 x 10 x 2	$\overline{MRAS}$			14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	$\overline{MBA}$																		1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
14 x 11 x 2	$\overline{MRAS}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	$\overline{MBA}$																	1	0												
	$\overline{MCAS}$																			11	9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{MRAS}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	$\overline{MBA}$																		1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
13 x 11 x 2	$\overline{MRAS}$				12	11	10	9	8	7	6	5	4	3	2	1	0														
	$\overline{MBA}$																	1	0												
	$\overline{MCAS}$																			11	9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{MRAS}$					12	11	10	9	8	7	6	5	4	3	2	1	0													
	$\overline{MBA}$																		1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
13 x 9 x 2	$\overline{MRAS}$						12	11	10	9	8	7	6	5	4	3	2	1	0												
	$\overline{MBA}$																			1	0										
	$\overline{MCAS}$																					8	7	6	5	4	3	2	1	0	

Table 12-5. DDR1 Address Multiplexing for 32-bit Data Bus (Continued)

Row x Col		MSB		Address from Core Initiator																								LSB
		30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1–0
12 x 10 x 2	$\overline{MRAS}$						11	10	9	8	7	6	5	4	3	2	1	0													
	$\overline{MBA}$																		1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
12 x 9 x 2	$\overline{MRAS}$							11	10	9	8	7	6	5	4	3	2	1	0												
	$\overline{MBA}$																			1	0										
	$\overline{MCAS}$																					8	7	6	5	4	3	2	1	0	
12 x 8 x 2	$\overline{MRAS}$								11	10	9	8	7	6	5	4	3	2	1	0											
	$\overline{MBA}$																				1	0									
	$\overline{MCAS}$																						7	6	5	4	3	2	1	0	

Table 12-6. DDR2 Address Multiplexing for 32-bit Data Bus

Row x Col		MSB		Address from Core Initiator																								LSB
		30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1–0
16 x 10 x 3	$\overline{MRAS}$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	$\overline{MBA}$																	2	1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
15 x 10 x 3	$\overline{MRAS}$		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	$\overline{MBA}$																	2	1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 3	$\overline{MRAS}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	$\overline{MBA}$																	2	1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{MRAS}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	$\overline{MBA}$																		1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{MRAS}$				12	11	10	9	8	7	6	5	4	3	2	1	0														
	$\overline{MBA}$																	2	1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	

Table 12-6. DDR2 Address Multiplexing for 32-bit Data Bus (Continued)

Row x Col		MSB		Address from Core Initiator																										LSB
		30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1–0
13 x 10 x 2	$\overline{MRAS}$					12	11	10	9	8	7	6	5	4	3	2	1	0													
	$\overline{MBA}$																		1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
13 x 9 x 2	$\overline{MRAS}$						12	11	10	9	8	7	6	5	4	3	2	1	0												
	$\overline{MBA}$																			1	0										
	$\overline{MCAS}$																					8	7	6	5	4	3	2	1	0	

12.2 JEDEC Standard DDR SDRAM Interface Commands

This section describes the commands and timings for DDR or DDR2 modes. The DDR memory controller performs all read or write accesses to DDR SDRAM using JEDEC standard DDR SDRAM interface commands. The SDRAM device samples command and address inputs on rising edges of the memory clock; data is sampled on both the rising and falling edges of DQS. Data read from the DDR SDRAM is also sampled on both edges of DQS.

Following are the DDR SDRAM interface commands (summarized in **Table 12-7**) 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 4.
- *Write*. Latches column address and transfers data from the data pins to the selected sense amplifier as determined by the column address. During each succeeding clock edge, additional data is transferred to the sense amplifiers from the data pins without additional write commands. The amount of data transferred is determined by the data masks and the burst size, which is set to four by the DDR memory controller.

- *Refresh* (similar to $\overline{\text{MCAS}}$ before $\overline{\text{MRAS}}$). Causes a row to be read in all logical banks (JEDEC SDRAM) as determined by the refresh row address counter. This refresh row address counter is internal to the SDRAM. After it is read, the row is automatically rewritten in the memory array. All logical banks must be in a precharged state before a refresh. The memory controller also supports posted refreshes in which several refreshes execute at once, and the refresh interval can be extended.
- *Mode register set (for configuration)*. Allows DDR SDRAM options to be set. These options are: $\overline{\text{MCAS}}$ latency, additive latency (for DDR2), write recovery (for DDR2), burst type, and burst length. $\overline{\text{MCAS}}$ latency 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}, and so on). Burst type is always sequential. Although some SDRAMs provide burst lengths of 1, 2, 4, 8, and page size, this memory controller supports only a burst length of 4. The DDR memory controller executes the mode register set command during system initialization. Parameters such as mode register data, $\overline{\text{MCAS}}$ latency, burst length, and burst type, are set by software in DDR_SDRAM_MODE[SDMODE] and transferred to the SDRAM array by the DDR memory controller after DDR_SDRAM_CFG[MEMEN] is set. If DDR_SDRAM_CFG[BI] is set to bypass the automatic initialization, software can configure the mode registers via the DDR_SDRAM_MD_CNTL register.
- S*elf refresh*. For use when the device is in standby for very long periods of time. Automatically generates internal refresh cycles to keep the data in all memory banks refreshed. Before execution of this command, the DDR controller places all logical bank in a precharged state.

Table 12-7. DDR SDRAM Command Table

Operation	CKE Previous	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
Auto refresh	H	H	L	L	L	H	X	X	X
Self refresh	H	L	L	L	L	H	X	X	X

12.3 DDR SDRAM Clocking and Interface Timing

The DDR memory controller supports four-beat bursts to SDRAM. For single-beat reads, the DDR memory controller performs a four-beat burst read but ignores the last three beats. Single-beat writes are performed by masking the last three beats of the four-beat burst using the data mask MDM[0–3]. If ECC is disabled, writes smaller than words are performed by appropriately activating the data mask. If ECC is enabled, the controller performs a read-modify write.

Note: If a second read or write is pending, reads shorter than four beats are not terminated early even if some data is irrelevant.

To accommodate available memory technologies across a wide spectrum of operating frequencies, the DDR memory controller allows you to set the timing intervals listed in **Table 12-8** with a granularity of one memory clock cycle, except for the CASLAT field, which can be programmed with a 1/2 clock granularity.

The value of these parameters (in whole clock cycles) must be set by application software at system initialization before the DDR controller is enabled and must be kept in the DDR memory controller configuration registers. Any update of the timing parameters should be done while the controller is disabled.

Table 12-8. DDR SDRAM Interface Timing Intervals

Timing Intervals	Definition	Register/Page
ACTTOACT	The number of clock cycles from a bank-activate command to another bank-activate command within a physical bank. This interval is listed in the AC specifications of the SDRAM as t_{RRD}.	DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1) **page 12-38**
ACTTOPRE	**Activate-to-Precharge Interval** The number of clock cycles from an activate command until a precharge command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RAS}.	
ACTTORW	**Activate-to-Read/Write Interval for SDRAM** The number of clock cycles from an activate command until a read or write command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RCD}.	
BSTOPRE	**Open Page Interval** The number of clock cycles to maintain a page open after an access. The page open duration counter is reloaded with BSTOPRE each time the page is accessed (including page hits). When the counter expires, the open page is closed with a SDRAM precharge bank command as soon as possible.	DDR SDRAM Interval Configuration Register **page 12-49**
CASLAT	**MCAS Latency from READ Command** Used in conjunction with additive latency to obtain the READ latency. The number of clock cycles between the registration of a READ command by the SDRAM and the availability of the first piece of output data. If a READ command is registered at clock edge n, and the read latency is m clocks, the data is available nominally coincident with clock edge $n + m$.	DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1) **page 12-38**

Table 12-8. DDR SDRAM Interface Timing Intervals (Continued)

Timing Intervals	Definition	Register/Page
PRETOACT	**Precharge-to-Activate Interval** The number of clock cycles from a precharge command until an activate or a refresh command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RP}.	DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1) **page 12-38**
REFINT	**Refresh Interval** Represents the number of memory bus clock cycles between refresh cycles. One row is refreshed in each SDRAM bank during each refresh cycle. Depending on DDR_SDRAM_CFG_2[NUM_PR], some number of rows are refreshed in each SDRAM bank during each refresh cycle. The value of REFINT depends on the specific SDRAMs used and the frequency of the interface. This interval is listed in the AC specifications of the SDRAM as t_{REFI}.	DDR SDRAM Interval Configuration Register **page 12-49**
REFREC	**Refresh Recovery Time** The number of clock cycles from the refresh command until an activate command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RFC}.	DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1) **page 12-38**
WR_DATA_DELAY	**Write Data Delay** Provides different options for the timing between a write command and the write data strobe. This allows write data to be sent later than the nominal time to meet the SDRAM timing requirement between the registration of a write command and the reception of a data strobe associated with the write command. The specification dictates that the data strobe may not be received earlier than 75% of a cycle, or later than 125% of a cycle, from the registration of a write command. This parameter is not defined in the SDRAM specification. It is implementation-specific, defined for the DDR memory controller in TIMING_CFG_2.	DDR SDRAM Timing Configuration Register 2 (TIMING_CFG_2) **page 12-40**
WRREC	**Write Recovery** The number of clock cycles from the last beat of a write until a precharge command is allowed. This interval, write recovery time, is listed in the AC specifications of the SDRAM as t_{WR}.	DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1) **page 12-38**
WR_DATA_DELAY	**Last Write Pair to Read Command**. Controls the number of clock cycles from the last write data pair to the subsequent read command to the same bank. This interval, write recovery time, is listed in the AC specifications of the SDRAM as t_{WTR}.	

Software should initialize the parameters in the DDR controller registers with the appropriate values (by boot code, for example) before the controller is enabled. Altering the register values while the controller is enabled can produce unpredictable controller behavior, can cause data loss, and can lock the controller or device.

System software is responsible 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 attempting any accesses to SDRAM.

Figure 12-5 through **Figure 12-7** show DDR SDRAM timing for various types of accesses, including a single-beat read, a single-beat write, and a burst-write. Note that all 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 that DDR_SDRAM_CLK_CNTL[CLKADJ] = 0100 (set to 1/2 DRAM cycle), the additive latency is 0 DRAM cycles, and the write latency is 1 DRAM cycle (for DDR1).

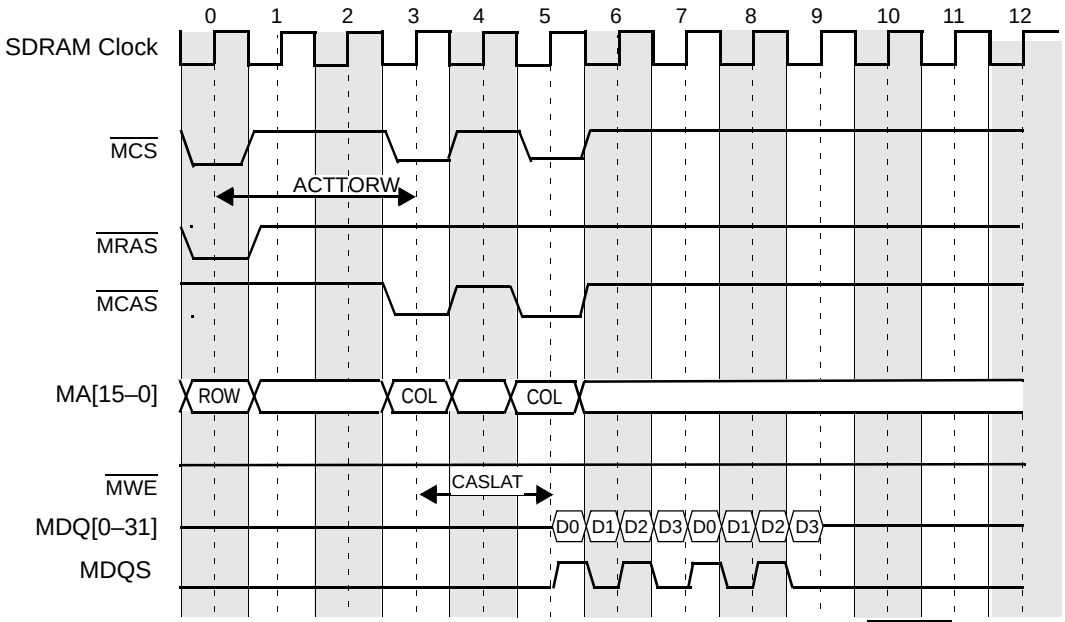


Figure 12-5. DDR SDRAM Burst Read Timing: ACTTORW = 3, $\overline{\text{MCAS}}$ Latency = 2

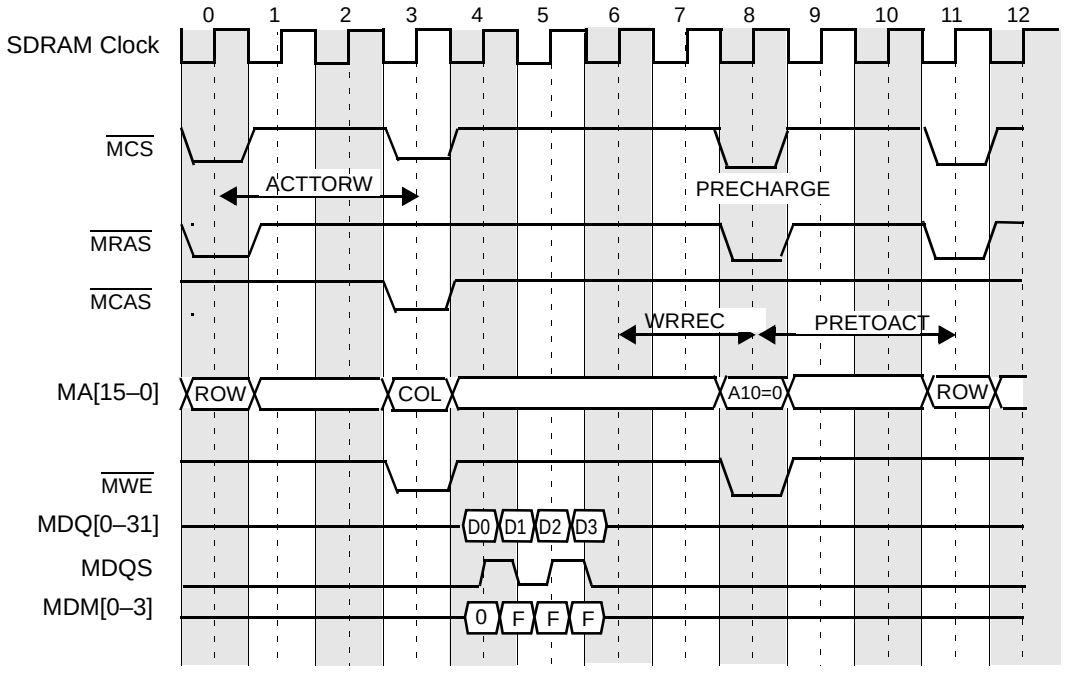


Figure 12-6. DDR SDRAM Single-Beat (Word) Write Timing: ACTTORW = 3

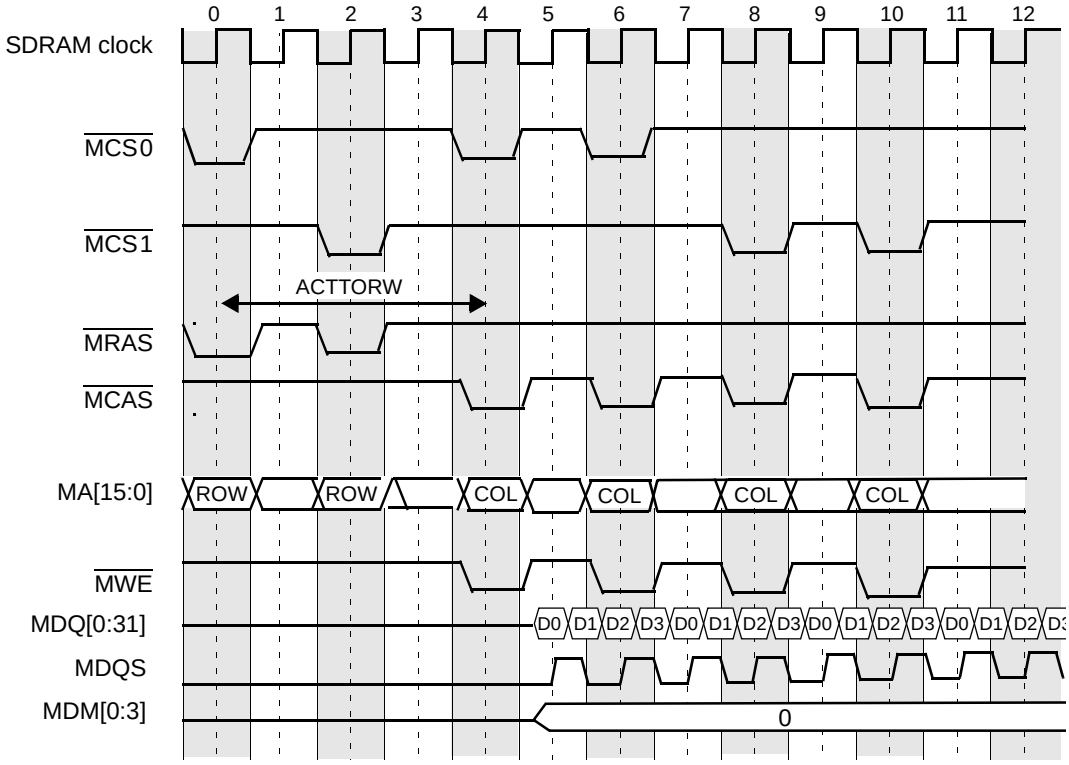


Figure 12-7. DDR SDRAM 4-Beat Burst Write Timing: ACTTORW = 4

12.3.1 Clock Distribution

- If a system is composed of many devices, consider using zero-delay PLL clock buffers that are compliant with the JEDEC-JESD82 standard. These buffers are designed for DDR applications.
- A 40-bit × 64 Mbyte DDR bank has up to 5 DDR memory chips (10 chips in a two-chip-select system). Distribute MCK/$\overline{\text{MCK}}$ equally between chips to ensure similar load/delay.
- PCB traces for DDR clock signals should be short, all on the same layer, and of equal length and loading.
- DDR SDRAM manufacturers provide details on PCB layout and termination issues.

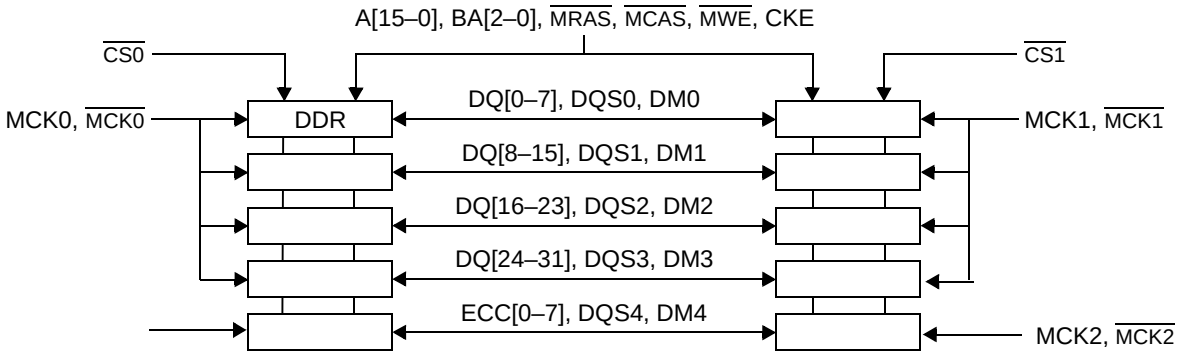


Figure 12-8. DDR SDRAM Clock Distribution Example for x8 DDR SDRAMs

12.3.2 DDR SDRAM Mode-Set Command Timing

The DDR controller transfers the mode register set commands to the SDRAM array and it uses the setting of TIMING_CFG_0[MRS_CYC] for the Mode Register Set cycle time. **Figure 12-9** shows the timing of the mode-set command. The first transfer corresponds to the ESDMODE code; the second corresponds to SDMODE. The Mode Register Set cycle time is set to 2 DRAM cycles.

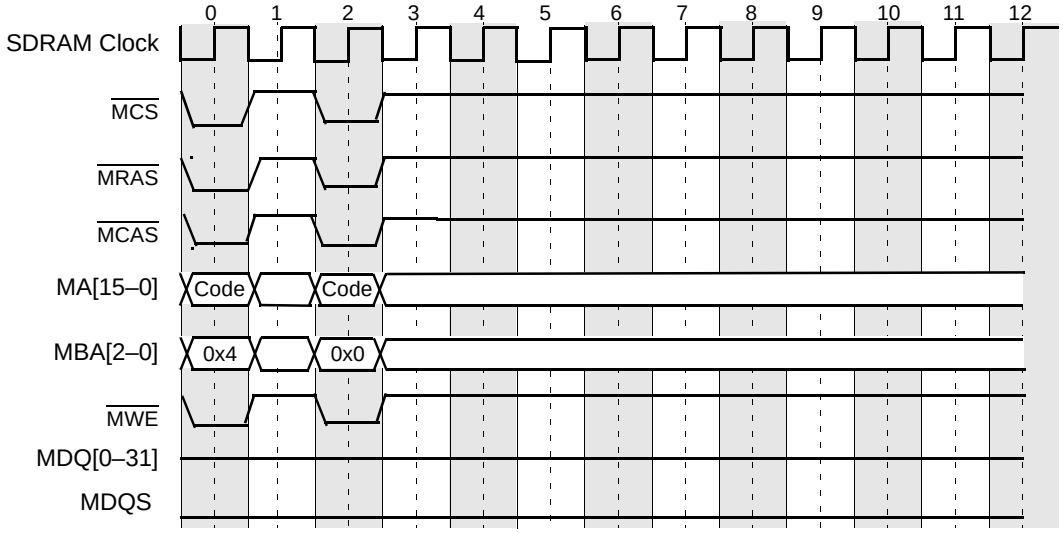


Figure 12-9. DDR SDRAM Mode-Set Command Timing

12.3.3 DDR SDRAM Write Timing Adjustments

The DDR memory controller facilitates system design flexibility by providing a write timing adjustment parameter, WRITE DATA DELAY, configured in TIMING_CFG_2[WRITE_DATA_DELAY] for data and DQS. The DDR SDRAM specification requires that DQS be received no sooner than 75 percent and no later than 125 percent of an SDRAM clock period from the capturing clock edge of the command/address at the SDRAM. The TIMING_CFG_2[WRITE_DATA_DELAY] parameter can be used to meet this timing requirement for a variety of system configurations, ranging from a system with one SDRAM device to a fully populated system with ten memory devices. The TIMING_CFG_2[WRITE_DATA_DELAY] field 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 12-10** shows the use of the write data delay parameter.

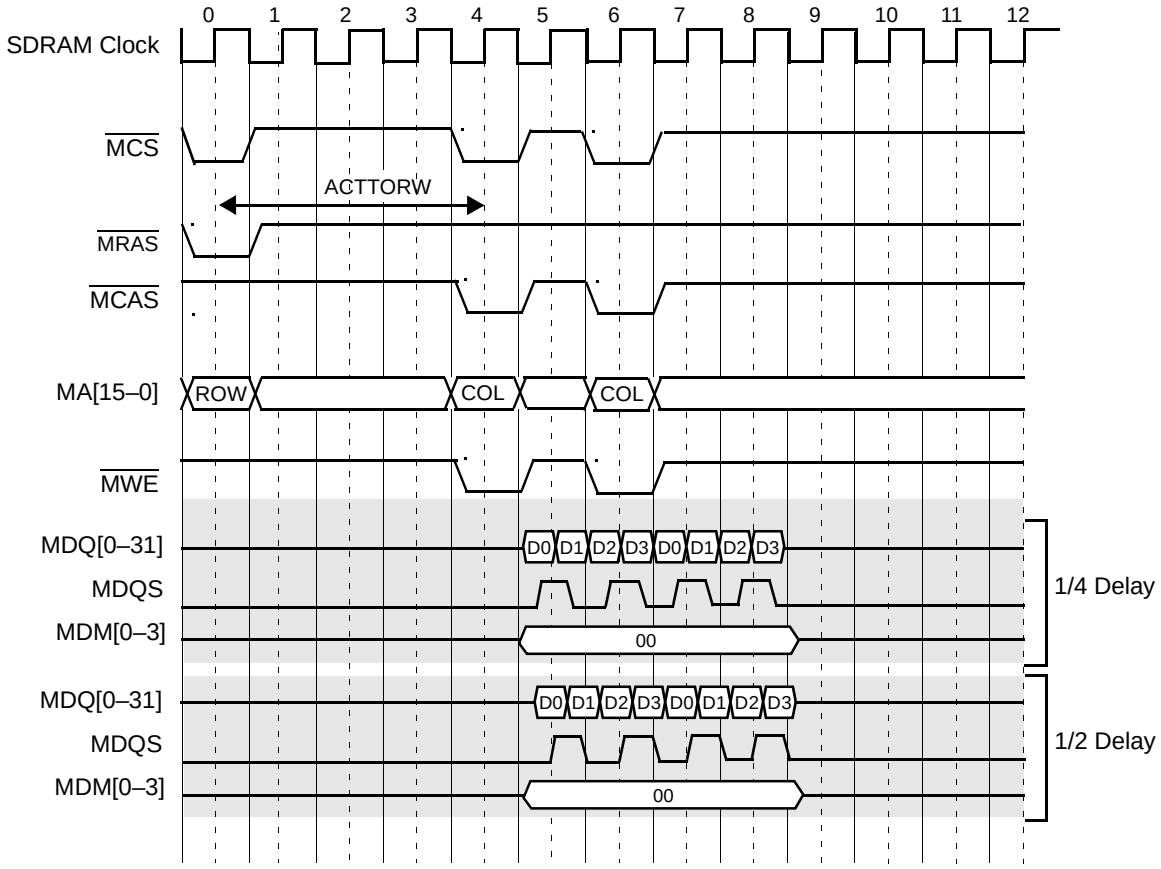


Figure 12-10. Write Timing Adjustments Example for Write Latency = 1

12.3.4 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 or a soft-stop state. The REFINT value, which represents the number of memory bus clock cycles between refresh cycles, must allow any outstanding transactions to complete before a refresh request is sent to the memory after the REFINT value is reached. If a memory transaction is in progress when the refresh interval is reached, the refresh cycle waits for the transaction to complete. In the worst case, the refresh cycle must wait the number of bus clock cycles required by the longest programmed access. To ensure that the latency caused by a memory transaction does not violate the device refresh period, it is recommended that the programmed value of REFINT be less than that required by the SDRAM. When a refresh cycle is required, the DDR memory controller does the following:

1. Completes all current memory requests.
2. Closes all open pages with a PRECHARGE ALL command to each DDR SDRAM bank with an open page (as indicated by the row open table).

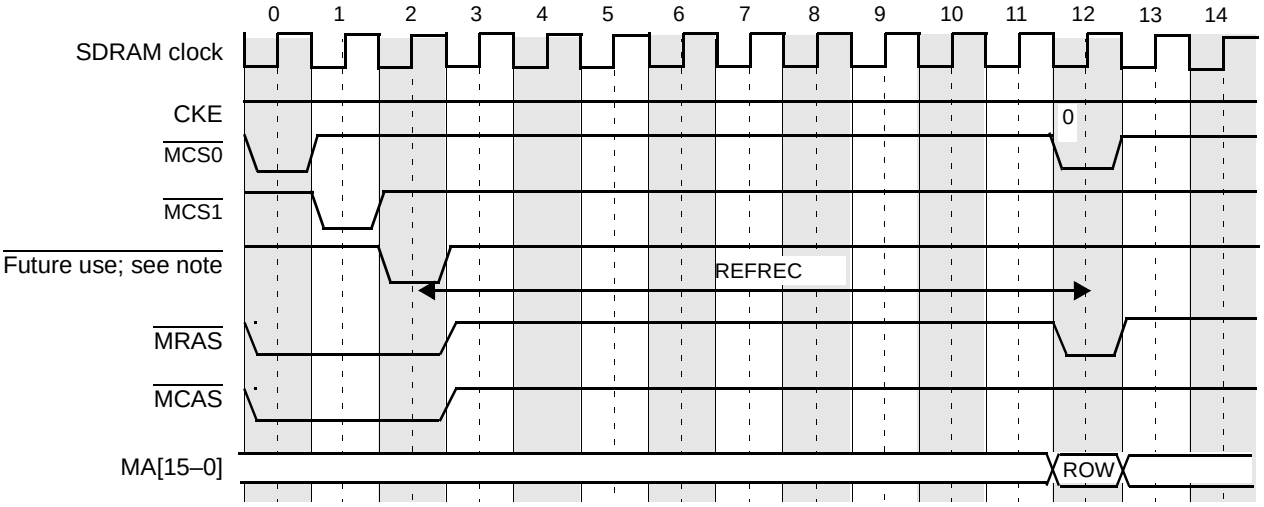
3. Issues one or more auto-refresh commands to each DDR SDRAM bank (as identified by its chip select) to refresh one row in each logical bank of the selected physical bank.

The auto refresh commands are staggered across the possible banks to reduce instantaneous power requirements. Three sets of auto refresh commands are issued on consecutive cycles. The initial PRECHARGE ALL commands are also staggered in three groups. When the system enters self refresh mode, only one refresh command is issued simultaneously to all physical banks. For the entire refresh sequence, no cycle optimization occurs for the usual case where fewer banks are installed. After the refresh sequence completes, any pending memory request is initiated after an inactive period specified by TIMING_CFG_1[REFREC]. In addition, posted refreshes allow the refresh interval to be set to a larger value.

Note: The MSC8144 will initiate three cycles of Precharge ALL commands and three cycles of Refresh commands although there are only two banks (chip select) available,

12.3.4.1 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 12-11** (TIMING_CFG_1[REFREC] = 10 in this example). System software is responsible for optimal configuration of TIMING_CFG_1 [REFREC] at reset. Configuration must be complete before DDR SDRAM accesses are attempted.



Note: The DDR controller is designed to support up to six banks. MSC8144E uses only two banks The third refresh command is used for devices with more than two banks.

Figure 12-11. DDR SDRAM Bank Staggered Auto Refresh Timing

12.3.4.2 DDR SDRAM Refresh and Power-Saving Modes

In full-on mode, the DDR memory controller supplies the normal auto refresh to SDRAM. In sleep mode, the DDR memory controller can be configured to take advantage of self-refreshing SDRAMs or to provide no refresh support. Self-refresh support is enabled by the DDR_SDRAM_CFG[SREN] bit.

Note: In absence of refresh support, system software must preserve DDRS DRAM data (for example - by copying the memory content to a non-volatile memory - disk, flash etc.) before entering power saving mode.

The dynamic power-saving mode uses the CKE pin to power down the memory device dynamically when there is no system memory activity. The CKE pin is deasserted when both conditions are met

- no memory refreshes are scheduled.
- no memory accesses are scheduled.

The CKE pin is reasserted when a new access or refresh is scheduled or the dynamic power mode is disabled. This mode is controlled with DDR_SDRAM_CFG[DYN_PWR].

Dynamic power management mode offers tight control of the memory system power consumption by trading power for performance through the use of CKE. Powering up the DDR SDRAM when a new memory reference is scheduled causes an access latency penalty, depending upon whether active or precharge power-down is used, along with the settings of TIMING_CFG_0[ACT_PD_EXIT] and TIMING_CFG_0[PRE_PD_EXIT]. A penalty of one cycle is shown in **Figure 12-12**.

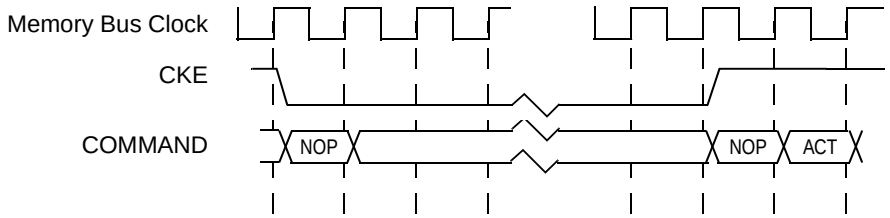


Figure 12-12. DDR SDRAM Power-Down Mode

The entry and exit timing for self-refreshing SDRAMs in Sleep mode is shown in **Figure 12-13** and **Figure 12-14**.

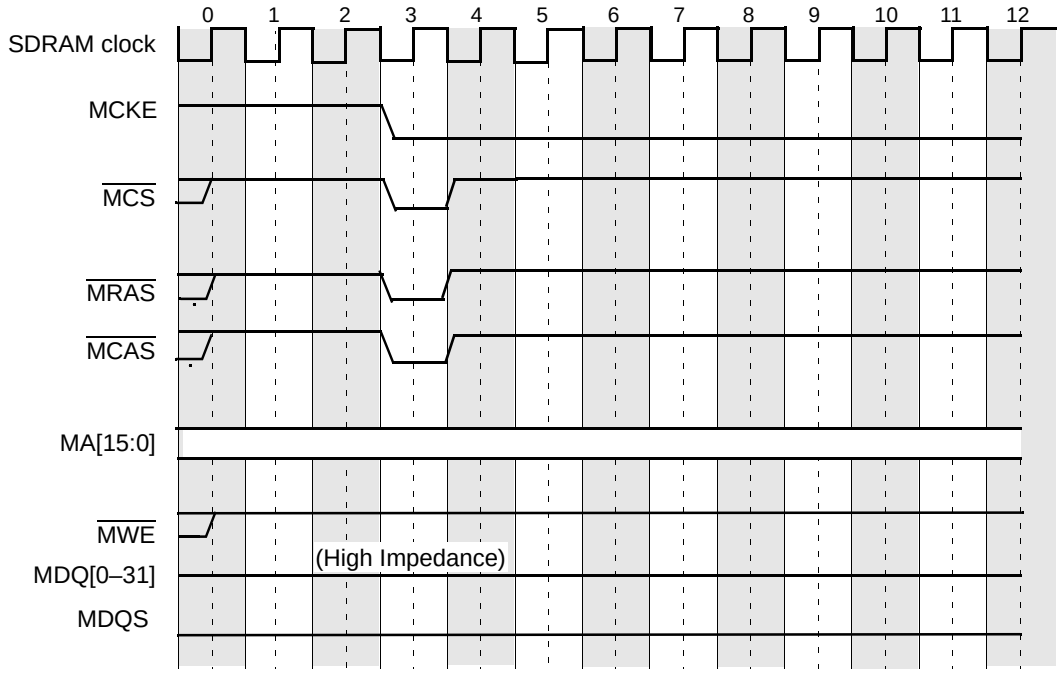


Figure 12-13. DDR SDRAM Self-Refresh Entry Timing

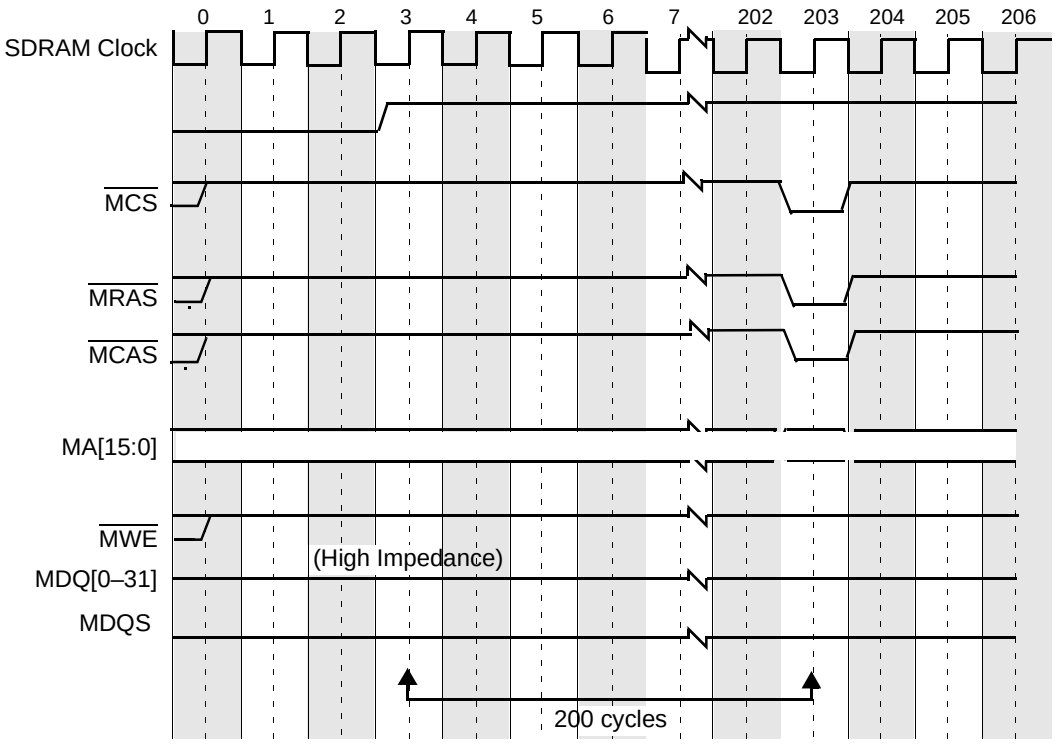


Figure 12-14. DDR SDRAM Self-Refresh Exit Timing

12.3.4.3 DDR Memory Controller Clock Stop Mode

To reduce the power dissipation, it is possible to shut down the clock to the DDR memory controller. To shut down the memory controller the DDR_PWR[STOP] bit should be asserted. Once this occurs, the DDR memory controller will end all the pending transactions to the memory, new transactions will not be accepted, and will enter the clock stop mode. During the clock stop period the DDR memory controller will not initiate refresh commands toward the memory. To exit the stop mode, perform a chip reset.

Note: Any transaction performed to the DDR memory controller by one of the cores or any other peripheral that can access the DDR memory while in stop mode will not be executed by the controller and the MSC8144E might get stuck. To prevent this stuck condition it is possible to disable the DDR window within the CLASS by deasserting REGISTER[bit] prior to requesting the stop mode.

12.3.5 DDR Data Beat Ordering

Transfers to and from memory are always performed in four-beat bursts (four beats = 16 or 8 bytes for 32 bits or 16 bits interface, accordingly)‘. For transfer sizes other than four beats, the data transfers are still in four-beat bursts. If ECC is enabled and either the access is not word-aligned or the size is not a multiple of a word, a full read-modify-write is performed for a write to SDRAM. If ECC is disabled or the access is word-aligned with a size that is a multiple of a word, the data masks MDM[0–5] 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 words from writing to SDRAM. For example, if a write transaction with a size of one word (4 bytes) is desired, then the second, third, and fourth beats of data are not written to SDRAM.

Table 12-9 lists the data beat sequencing to and from the DDR SDRAM and the data queues for each of the possible transfer sizes with each of the possible starting double-word offsets. All underlined double-word offsets are valid for the transaction.

Table 12-9. Memory Controller–Data Beat Ordering - for 32 bit interface

Transfer Size	Starting Word Offset	Word Sequence[1] to/from DRAM and Queues
1 word	0	**0** - 1 - 2 - 3
	1	**1** - 2 - 3 - 0
	2	**2** - 3 - 0 - 1
	3	**3** - 0 - 1 - 2
2 words	0	**0 - 1** - 2 - 3
	1	**1 - 2** - 3 - 0
	2	**2 - 3** - 0 - 1
3 words	0	**0 - 1 - 2** - 3
	1	**1 - 2 - 3** - 0
4 words	0	**0 - 1 - 2 - 3**
4 words	1	**1 - 2 - 3** - 0 - **0** - 1 - 2 - 3
	2	**2 - 3** - 0 - 1 - **0 - 1** - 2 - 3
	3	**3** - 0 - 1 - 2 - **0 - 1 - 2** - 3

1. All underlined word offsets are valid for the transaction.

Note: For MSC8144E the DDR controller will not perform wrapped accesses. Accesses that their start address is not aligned to 16 or 8 byte (according to the actual mode) and crosses the alignment boundary will be split into two accesses toward the memory. For example a 16 byte access to word offset 1 will result in two bursts one for words 1,2,3 and the second to word 0 of the next group - In any case the memory interface will be occupied for 8 data beats.

Note: Similar behavior will occur for 16bit memory interface - every “word” mentioned above should be read a “half word - two bytes”.

12.3.6 Page Mode and Logical Bank Retention

The DDR memory controller supports an open/closed page mode that allows an open page for each logical bank of DRAM. In closed page mode for DDR SDRAMs, the DDR memory controller uses the auto-precharge feature, which allows the controller to indicate the DDR SDRAM that it must automatically close the page after the read or write access. This is performed by using MA10 of the address during the COMMAND phase of the access to enable auto-precharge. Auto-precharge is non-persistent in that it is either enabled or disabled for each individual READ or WRITE command. However, it can be separately enabled or disabled for each chip select. In open page mode, the DDR memory controller retains the currently active SDRAM page by not issuing a precharge command. The page remains opens until one of the following conditions occurs:

- Refresh interval is met.
- The user-programmable DDR_SDRAM_INTERVAL[BSTOPRE] value is exceeded.
- There is a logical bank row collision with another transaction that must be issued.

Page mode can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, using page mode can save two to three clock cycles for subsequent burst accesses that hit in an active page. Also, better performance can be obtained by using more banks, especially in systems which use many different channels. Page mode is disabled by clearing DDR_SDRAM_INTERVAL[BSTOPRE] or setting CSx_CONFIG[AP_x_EN].

12.4 Error Checking and Correction

The DDR memory controller supports error checking and correcting (ECC) for the data path between the core initiator and system memory. The DDR memory controller detects all double-bit errors in a word, detects all multi-bit errors within a nibble, and it corrects all single-bit errors in a word. Other errors may be detected, but are not guaranteed to be corrected or detected. Multiple-bit errors are always reported when error reporting is enabled. When a single-bit error occurs, the single-bit error counter register is incremented, its value compared to the single-bit error trigger register. An error is reported when these values are equal. The

single-bit error registers can be programmed so that minor memory faults are corrected and ignored, but a catastrophic memory failure generates an interrupt.

For writes smaller than the bus width (32 or16 bits), the DDR memory controller performs a 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 data. The data and ECC code is then written to memory. If a multi-bit error is detected on the read, the transaction completes the read-modify-write to keep the DDR memory controller from hanging. However, the corrupt data is masked on the write, so the original contents in SDRAM remain unchanged. The syndrome encoding for the ECC code are shown in **Table 12-10** and **Table 12-11**.

Table 12-10. 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	•	•						•	16			•	•				•
1	•		•					•	17			•		•			•
2	•			•				•	18	•		•		•			
3	•				•			•	19		•	•		•			
4	•	•				•			20			•	•		•		
5	•		•			•			21			•		•	•		
6	•			•		•			22	•		•		•	•		•
7	•				•	•			23		•	•		•	•		•
8		•	•					•	24		•				•	•	
9		•		•				•	25			•			•	•	
10		•			•			•	26				•		•	•	
11	•	•			•				26	•					•	•	
12		•	•			•			28		•				•		•
13		•		•		•			29			•			•		•
14		•			•	•			30				•		•		•
15	•	•			•	•		•	31	•					•		•

Table 12-11. DDR SDRAM ECC Syndrome Encoding (Check Bits)

Check Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•							
1		•						
2			•					
3				•				
4					•			

Table 12-11. DDR SDRAM ECC Syndrome Encoding (Check Bits) (Continued)

Check Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
5						•		
6							•	
7								•

12.5 Error Management

The DDR memory controller detects single-bit, multi-bit, memory select, and training errors. The following discussion assumes all the relevant error detection, correction, and reporting functions are enabled as described in **Section 12.7**, *Memory Controller Programming Model*, on page 12-30.

Single-bit errors are counted and reported based on the ERR_SBE register value (see **Table 12-45** on page 12-63). When a single-bit error is detected, the DDR memory controller does the following:

1. Corrects the data.
2. Increments the single-bit error counter ERR_SBE[SBEC].
3. Generates a critical interrupt if the counter value ERR_SBE[SBEC] equals the programmable threshold ERR_SBE[SBET].
4. Completes the transaction normally.

If a multi-bit error is detected for a read, the DDR memory controller logs the error and generates the critical interrupt (if enabled, as described in **Table 12-41** on page 12-59). The DDR memory controller also detects a memory select error, which causes the DDR memory controller to log the error and generate a critical interrupt (if enabled, as described in **Table 12-40** on page 12-58). This error is detected if the address from the memory request does not fall into any of the enabled, programmed chip-select address ranges. For all memory select errors, the DDR memory controller does not issue any transactions onto the pins after the first read has returned data strobes. If the DDR memory controller is not using sample points, then a dummy transaction is issued to DDR SDRAM with the first enabled chip select. **Table 12-12** describes the errors.

The final error the memory controller detects is the automatic calibration error. This error is set if the memory controller detects an error during its training sequence.

Table 12-12. Memory Controller Errors

Category	Error	Descriptions	Action	Detect Register
Notification	Single-bit ECC threshold	The number of ECC errors has reached the threshold specified in the ERR_SBE.	The error is reported via critical interrupt if enabled.	The error control register logs only read versus write, not full type
Access Error	Multi-bit ECC error	A multi-bit ECC error is detected during a read, or read-modify-write memory operation.		
	Memory select error	Read, or write, address does not fall within the address range of any of the memory banks.		
Training	Calibration error	One of the calibration processes executed during the initialization failed		

12.6 Set-Up and Initialization

System software can configure the DDR memory controller. The DDR memory controller uses its bank map to assert the appropriate $\overline{\text{MCS}}$[0–1] signal for memory accesses according to the provided bank depths. System software also configures the DDR memory controller at system start-up to multiplex the row and column address bits for each bank (see **Table 12-17** on page 12-33). Address multiplexing occurs according to these configuration bits. At system power-up, initialization software (boot code, for example) must set up the programmable parameters in the memory interface configuration registers listed in **Table 12-13**.

Note:

1. Before initiating the DDR controller registers the DDR_GCR[DDR_VSEL] should be programmed according to the DDR device connected to the MSC8144. For DDR1 devices the DDR_GCR[DDR_VSEL] = 0 (reset value), for DDR2 devices the DDR_GCR[DDR_VSEL] = 1 Programming the correct value is essential for the correct

operation of the DDR interface, since it defines to the I/O cells the selected operating voltage.

2. The DDR WINDOW END ADDRESS REGISTER[bit] in the CLASS should be programmed to point to the last available DDR address. Using this register will prevent accesses to unavailable memory space beyond the DDR memory space.

Table 12-13. Memory Interface Configuration Register Initialization Parameters

Register	Parameter Bits	Page
Chip-Select Memory Bounds Register (CSx_BNDS)	Starting Address for Chip Select x (SA*x*) Ending Address for Chip Select x (EA*x*)	**Table 12-16** on page 12-32
Chip-Select Configuration Register (CSx_CONFIG)	Chip Select x Enable (CS_x_EN) Chip Select x Auto-Precharge Enable (AP_x_EN) ODT for Reads (ODT_RD_CFG) ODT for Writes (ODT_WR_CFG) Number of Bank Bits (BA_BITS_CS_x) Number of Row Bits (ROW_BITS_CS_x) Number of Column Bits (COL_BITS_CS_x)	**Table 12-17** on page 12-33
DDR SDRAM Extended Refresh Recovery Register (TIMING_CFG_3)	Extended Refresh Recovery Time (REFR)	**Table 12-18** on page 12-35
Timing Configuration 0 Register (TIMING_CFG_0)	Read-to-Write Turn-Around (RWT) Write-to-Read Turn-Around (WRT) Read-to-Read Turn-Around (RRT) Write-to-Write Turn-Around (WWT) Active Power-Down Exit Timing (ACT_PD_EXIT) Precharge Power-Down Exit Timing (PRE_PD_EXIT) ODT Power-Down Exit Timing (PDT_PD_EXIT) Mode Register Set Cycle Time (MRS_CYC)	**Table 12-19** on page 12-35
Timing Configuration 1 Register (TIMING_CFG_1)	Precharge-to-Activate Interval (PRETOACT) Activate-to-Precharge Interval (ACTTOPRE) Activate to Read/Write Interval for SDRAM (ACTTORW) $\overline{\text{MCAS}}$ Latency from Read Command (CASLAT) Refresh Recovery Time (REFREC) Last Data to Precharge Minimum Interval (WRREC) Activate-to-Activate Interval (ACTTOACT) Last Write Data Pair to Read Interval (WR_DATA_DELAY)	**Table 12-20** on page 12-38
Timing Configuration 2 Register (TIMING_CFG_2)	Additive Latency (ADD_LAT) $\overline{\text{MCAS}}$-to-Preamble Override (CPO) Write Latency (WR_LAT) Read-to-Precharge (RD_TO_PRE) Write Data Delay (WR_DATA_DELAY) Minimum CKE Pulse Width (CKE_PLS) Window for Four Activates (FOUR_ACT)	**Table 12-21** on page 12-40
DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG)	Self Refresh Enable (SREN) ECC Enable (ECC_EN) SDRAM Type (SDRAM_Type) Dynamic Power Management Mode (DYN_PWR) 32-Bit Bus Enable (32_BE) Non-Current Auto Precharge (NCAP) 2T Timing Enable (2T_EN) Half-Strength Drive Enable (HSE) Bypass Initialization (BI)	**Table 12-22** on page 12-43

Table 12-13. Memory Interface Configuration Register Initialization Parameters (Continued)

Register	Parameter Bits	Page
DDR SDRAM Control Configuration Register 2 (DDR_SDRAM_CFG_2)	Force Self Refresh (FRC_SR) DLL Reset Disable (DLL_RST_DIS) DQS Configuration (DQS_CFG) ODT Configuration (ODT_CFG) Number of Posted Refreshes (NUM_PR) DRAM Data Initialization (D_INIT)	**Table 12-23** on page 12-45
DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE)	Extended SDRAM Mode (ESDMODE) SDRAM Mode (SDMODE)	**Table 12-24** on page 12-46
DDR SDRAM Mode Configuration 2 Register (DDR_SDRAM_MODE_2)	Extended SDRAM Mode 2 (ESDMODE2) Extended SDRAM Mode 3 (ESDMODE3)	**Table 12-25** on page 12-47
DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL)	Refresh Interval (REFINT) Precharge Interval (BSTOPRE)	**Table 12-27** on page 12-49
DDR SDRMA Data Initialization Register	Initialization Value (IVAL)	**Table 12-28** on page 12-50
DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL)	Clock Adjust (CLK_ADJUST)	**Table 12-29** on page 12-51
DDR Initialization Address Register (DDR_INIT_ADDRESS)	Initialization Address (IADDR)	**Table 12-30** on page 12-52

12.6.1 Programming Differences Between Memory Types

Depending upon the memory type, certain fields must be programmed differently. **Table 12-14** illustrates the differences in certain fields for different memory types. This table does not list all fields that must be programmed.

Note: Before initiating the DDR controller registers the DDR_GCR[DDR_VSEL] should be programmed according to the DDR device connected to the MSC8144. For DDR1 devices the DDR_GCR[DDR_VSEL] = 0 (reset value), for DDR2 devices the DDR_GCR[DDR_VSEL] = 1 Programming the correct value is essential for the correct operation of the DDR interface, since it defines to the I/O cells the selected operating voltage.

Table 12-14. Programming Differences Between Memory Types

Parameter	Description	Differences		Page
AP_x_EN	Chip Select *x* Auto Precharge Enable	DDR1 DDR2	Can be used to place chip select *x* into auto precharge mode.	**Table 12-17** on page 12-33

Table 12-14. Programming Differences Between Memory Types (Continued)

Parameter	Description	Differences		Page
ODT_RD_CFG	ODT Read Configuration	DDR1	Always clear to 000.	**Table 12-17** on page 12-33
		DDR2	Can be enabled to assert ODT if desired. This bit can be configured differently depending upon system topology. However, systems with only one chip select typically do not use ODT when issuing reads to the memory.	
ODT_WR_CFG	ODT Write Configuration	DDR1	Always clear to 000.	**Table 12-17** on page 12-33
		DDR2	Can be enabled to assert ODT if desired. This bit can be configured differently depending upon system topology. However, ODT typically is set to assert the chip select that is the target of the write (value would be 001).	
PDT_PD_EXIT	ODT Power-Down Exit	DDR1	Cleared to 0000.	**Table 12-19** on page 12-35
		DDR2	Set according to the DDR2 specifications for the memory used. The JEDEC parameter this applies to is t_{axpd}.	
PRETOACT	Precharge-to-Activate Interval	DDR1	Configure according to the specifications for the memory used (t_{rp})	**Table 12-20** on page 12-38
		DDR2		
ACTTOPRE	Activate-to-Precharge Interval	DDR1	Configure according to the specifications for the memory used (t_{ras})	**Table 12-20** on page 12-38
		DDR2		
ACTTORW	Activate to Read/Write Timing	DDR1	Configure according to the specifications for the memory used (t_{rcd})	**Table 12-20** on page 12-38
		DDR2		
CASLAT	CAS Latency	DDR1	Configure to the desired $\overline{\text{CAS}}$ latency	**Table 12-20** on page 12-38
		DDR2		
REFREC	Refresh Recovery	DDR1	Configure along with the Extended Refresh Recovery, to the specifications for the memory used (t_{rfc})	**Table 12-20** on page 12-38
		DDR2		
WRREC	Write Recovery	DDR1	Configure according to the specifications for the memory used (t_{wr})	**Table 12-20** on page 12-38
		DDR2		
ACTTOACT	Activate A to Activate B Interval	DDR1	Configure according to the specifications for the memory used (t_{rrd})	**Table 12-20** on page 12-38
		DDR2		
WRTORD	Write to Read Interval	DDR1	Configure according to the specifications for the memory used (t_{wrd})	**Table 12-20** on page 12-38
		DDR2		
ADD_LAT	Additive Latency	DDR1	Configure to 000	**Table 12-21** on page 12-40
		DDR2	Configure to the desired additive latency. This must be set to a value less than TIMING_CFG_1[ACTTORW]	
WR_LAT	Write Latency	DDR1	Configure to 001.	**Table 12-21** on page 12-40
		DDR2	Configure to CAS latency – 1 cycle. For example, if the CAS latency if 5 cycles, then configure this field to 100 (4 cycles).	

Table 12-14. Programming Differences Between Memory Types (Continued)

Parameter	Description	Differences		Page
RD_TO_PRE	Read-to-Precharge Interval	DDR1	Configure to 010.	**Table 12-21** on page 12-40
		DDR2	Configure according to the specifications for the memory used (ADD_LAT + t_{rtp}).	
CKE_PLS	Minimum CKE Pulse Width	DDR1	Can be set to 001.	**Table 12-21** on page 12-40
		DDR2	Configure according to the specifications for the memory used (t_{cke}).	
FOUR_ACT	Window for Four Activates	DDR1	Configure to 0001.	**Table 12-21** on page 12-40
		DDR2	Configure according to the specifications for the memory used (t_{faw}). Applicable for 8 logical banks.	
2T_EN	2T Timing Enable	DDR1	In heavily loaded systems, this bit can be set to 1 to gain extra timing margin on the interface at the cost of address/command bandwidth.	**Table 12-22** on page 12-43
		DDR2		
DLL_RST_DIS	DLL Reset DIsable	DDR1	Typically 0, unless you want to bypass the DLL reset when exiting self refresh.	**Table 12-23** on page 12-45
		DDR2		
DQS_CFG	DQS Configuration	DDR1	Configure to 00.	**Table 12-23** on page 12-45
		DDR2	Can be either 00 or 01, depending upon whether differential strobes are used.	
ODT_CFG	ODT Configuration	DDR1	Configure to 00.	**Table 12-23** on page 12-45
		DDR2	Can be set for termination at the I/Os according to system topology. Typically, if ODT is enabled, the internal I/Os are set up for termination only during reads to DRAM.	
BSTOPRE	Burst To Precharge Interval	DDR1	Can be set to any value, depending upon the application. To enable auto precharge, clear this field to all 0s.	**Table 12-27** on page 12-49
		DDR2		

12.6.2 DDR SDRAM Initialization Sequence

After all parameters are configured, system software must set CSx_CONFIG[MEMEN] to enable the memory interface. 200 µs must elapse after DRAM clocks are stable (DDR_SDRAM_CLK_CNTL[CLK_ADJUST] is set and any chip select is enabled) before MEMEN can be set. Therefore, a delay loop in the initialization code may be necessary if software is enabling the memory controller. If DDR_SDRAM_CFG[BI] bit is not set to bypass initialization, the DDR memory controller conducts an automatic initialization sequence to the memory, which follows the memory specifications. If the bypass initialization mode is used, software can initialize the memory through the DDR_SDRAM_MD_CNTL register.

Table 12-15 illustrates the steps the DDR controller will follow when using this automatic hardware initialization. If the bypass initialization mode is used, then software can initialize the memory through the DDR_SDRAM_MD_CNTL register.

Table 12-15. DRAM Initialization Requirements

DDR1	DDR2
CKE low at POR	CKE low at POR
NOP w/CKE high	NOP w/CKE high
	Wait 400 ns (200 cycles)
PRE_ALL	PRE_ALL
	EMRS(2)
	EMRS(3)
EMRS (en DLL)	EMRS (en DLL)
MRS (DLL reset)	MRS (DLL reset)
PRE_ALL	PRE_ALL
AUTO REFRESH	AUTO REFRESH
AUTO REFRESH	AUTO REFRESH
MRS (w/o DLL reset)	MRS (w/o DLL reset)
Wait 200 clocks	Wait 200 clocks
	Set OCD Disabled

12.7 Memory Controller Programming Model

In the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignore 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 12-32**.
- Chip-Select Configuration Register (CSx_CONFIG), **page 12-33**.
- DDR SDRAM Extended Refresh Recovery Register (TIMING_CFG_3), **page 12-34**.
- DDR SDRAM Timing Configuration 0 Register (TIMING_CFG_0), **page 12-35**.
- DDR SDRAM Timing Configuration 1 Register (TIMING_CFG_1), **page 12-38**.
- DDR SDRAM Timing Configuration 2 Register (TIMING_CFG_2), **page 12-40**.
- DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG), **page 12-42**.
- DDR SDRAM Control Configuration 2 Register (DDR_SDRAM_CFG_2), **page 12-44**.
- DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE), **page 12-46**.
- DDR SDRAM Mode Configuration 2 Register (DDR_SDRAM_MODE_2), **page 12-47**.

- DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL), **page 12-47**.
- DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL), **page 12-49**.
- DDR SDRAM Data Initialization Register (DDR_DATA_INIT), **page 12-50**.
- DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL), **page 12-50**.
- DDR Initialization Address Register (DDR_INIT_ADDRESS), **page 12-51**.
- DDR Initialization Enable Register (DDR_INIT_EN), **page 12-52**.
- DDR IP Block Revision 1 Register (DDR_IP_REV1), **page 12-53**.
- DDR IP Block Revision 2 Register (DDR_IP_REV2), **page 12-53**.
- Memory Data Path Error Injection Mask High Register (DDR_ERR_INJECT_HI), **page 12-54**.
- Memory Data Path Error Injection Mask Low Register (DDR_ERR_INJECT_LO), **page 12-55**.
- Memory Data Path Error Injection Mask ECC Register (DDR_ERR_INJECT), **page 12-55**
- Memory Data Path Read Capture High Register (CAPTURE_DATA_HI), **page 12-56**.
- Memory Data Path Read Capture Low Register (CAPTURE_DATA_LO), **page 12-57**.
- Memory Data Path Read Capture ECC Register (CAPTURE_ECC), **page 12-57**.
- Memory Error Detect Register (ERR_DETECT), **page 12-58**.
- Memory Error Disable Register (ERR_DISABLE), **page 12-59**.
- Memory Error Interrupt Enable Register (ERR_INT_EN), **page 12-60**.
- Memory Error Attributes Capture Register (CAPTURE_ATTRIBUTES), **page 12-61**.
- Memory Error Address Capture Register (CAPTURE_ADDRESS), **page 12-62**.
- Single-Bit ECC Memory Error Management Register (ERR_SBE), **page 12-62**.
- DDR Status Register (DDR_STOP_STATUS), **page 12-63**.
- DDR Power Control Register (DDR_PWR), **page 12-64**.
- MDIC Output Enable Control Register (MDIC_OE_CONT), **page 12-65**.
- DDR SDRAM Termination, OCD, and ODT Control Register (TERM_OCD_ODT_CONT), **page 12-66**.
- Clock Ratio Control Register (CLK_RATIO_CONT), **page 12-67**.

Note: The DDR controller registers use a base address of: 0xFFF20000.

12.7.1 Chip-Select Bounds (CSx_BNDS)

CSx_BNDS Chip-Select Bounds Register

CS0_BNDS Offset 0x0000

CS1_BNDS 0x0008

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

CSx_BNDS defines the starting and ending address of the memory space that corresponds to the individual chip selects.

Note: The size specified in CSx_BNDS should equal the size of physical DRAM. Also, note that EA*x* must be greater than or equal to SA*x*. The 8 msb of the address is used to define the SAx and EAx. Due to a specific implementation, the address values written to SAx and EAx should be shifted left by one bit. The address applied to the DDR Controller is also shifted left by one bit.

For example, if the DDR SDRAM start address is 0x4000_0000 the value written to SAx should be 0x080. If the DDR SDRAM end address is 0x43FF_FFFF the value written to EAx should be 0x087.

Table 12-16. CSx_BNDS Field Descriptions

Bit	Reset	Description
— 31–25	0	Reserved. Cleared to zero for future compatibility.
SAx 24–16	0	**Starting Address** Specifies the starting address for chip-select (bank) *x*. This value is compared against the 8 MSBs of the 32-bit address. See the previous note
— 15–9	0	Reserved. Cleared to zero for future compatibility.
EA*x* 8–0	*0*	**Ending Address** Specifies the ending address for chip select (bank) *x*. This value is compared against the 8 MSBs of the 32-bit address. See the previous note.

12.7.2 Chip-Select x Configuration Register (CSx_CONFIG)

CSx_CONFIG Chip-Select x Configuration Register

CS0_CONFIG Offset 0x0080

CS1_CONFIG 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							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

CSx_CONFIG registers enable the DDR chip selects and set the number of row and column bits used for each chip select. These registers should be loaded with the correct number of row and column bits for each SDRAM. Because the ROW_BITS_CS_x and COL_BITS_CS_x fields establish address multiplexing, it is essential to set these values correctly.

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 this specific chip select.	0 Chip select *x* is auto-precharged only if global auto-precharge mode is enabled (SICFG[BSTOPRE] = 0). 1 Chip select *x* always issues an auto-precharge for read and write transactions.
ODT_RD_CFG 22–20	0	**On-Die Termination (ODT) for Reads** Specifies when ODT is to be asserted for read accesses. Note that $\overline{\text{CAS}}$ latency plus additive latency must be at least 3 cycles for ODT_RD_CFG to be enabled. ODT should be used only with DDR2 memories.	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.
— 19	0	Reserved. Write to zero for future compatibility.	

Table 12-17. CSx_CONFIG Field Descriptions (Continued)

Bit	Reset	Description	Settings
ODT_WR_CFG 18–16	0	**ODT for Writes** Specifies when ODT is to be asserted for write accesses. Note that write latency plus additive latency must be at least 3 cycles for ODT_WR_CFG to be enabled. ODT should be used only with DDR2 memories.	000 Never assert ODT for writes. 001 Assert ODT only during writes to CSx. 010 Assert ODT only during writes to other chip selects. 011 Reserved. 100 Assert ODT for all writes. 101–111 Reserved.
BA_BITS_CS_x 15–14	0	**Number of Bank Bits** Specifies the number of logical bank bits for SDRAM on chip select *x*. see **Table 12-5** and **Table 12-6** for details	00 2 logical bank bits. 01 3 logical bank bits. 10–11 Reserved
— 13–11	0	Reserved. Write to zero for future compatibility.	
ROW_BITS_CS_x 10–8	0	**Number of Row Bits** Specifies the number of row bits for SDRAM on chip select *x.* see **Table 12-5** and **Table 12-6** for details	000 12 row bits 001 13 row bits 010 14 row bits 011 15 row bits 100 16 row bits 101–111 Reserved
— 7–3	0	Reserved. Write to zero for future compatibility.	
COL_BITS_CS_x 2–0	0	**Number of Column Bits** Specifies the number of column bits for SDRAM on chip select *x.* see **Table 12-5** and **Table 12-6** for details.	000 8 column bits. 001 9 column bits. 010 10 column bits. 011 11 column bits. 100–111 Reserved.

12.7.3 DDR SDRAM Extended Refresh Recovery Register (TIMING_CFG_3)

TIMING_CFG_3 DDR SDRAM Extended Refresh Recovery Register Offset 0x0100

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

TIMING_CFG_3 sets the extended refresh recovery time, which is combined with TIMING_CFG_1[REFREC] to determine the full refresh recovery time.

Table 12-18. TIMING_CFG_3 Bit Descriptions

Bits	Reset	Description	Setting	
— 31–19	0	Reserved. Write to zero for future compatibility.		
REFR 18–16	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 concatenated with TIMING_CFG_1[REFREC] to obtain a 7-bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final 7-bit value of the refresh recovery. t_{RFC} = {TIMING_CFG_3 || REFREC} + 8 min. value = 8 clocks TIMING_CFG_3 = 0x0, REFREC = 0x0 max. value = 135 clocks TIMING_CFG_3 = 0x7, REFREC = 0xF = 112+15+8	000 001 010 011 100 101 110 111	0 clock cycles. 16 clock cycles. 32 clock cycles. 48 clock cycles. 64 clock cycles. 80 clock cycles. 96 clock cycles. 112 clock cycles.
— 15–0	0	Reserved. Write to zero for future compatibility.		

12.7.4 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								R	R/W			R	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**
	—				PDT_PD_EXIT				—				MRS_CYC			
Type	R				R/W				R				R/W			
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1

TIMING_CFG_0 sets the number of clock cycles between various SDRAM control commands.

Table 12-19. 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. (BL = 4 for all accesses)	00 0 clock cycles. 01 1 clock cycle. 10 2 clock cycles. 11 3 clock cycles.

Table 12-19. TIMING_CFG_0 Field Descriptions (Continued)

Bit	Reset	Description	Settings
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 $\overline{\text{CAS}}$ latency rounded down to the next integer, WL is the programmed write latency, and BL is the burst length. (BL = 4 for all accesses)	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 - 3 clocks for the turnaround, selecting a value other than 00 adds extra cycles to this predefined value according to the selection	00 0 clock cycles. 01 1 clock cycle. 10 2 clock cycles. 11 3 clock cycles.
WWT 25–24	0	**Write-to-Write Turn-Around** Specifies how many extra cycles to add between writes to different chip selects. By default, 2 cycles are required between write commands to different chip selects. If 00 is selected the DDR Controller will use a predefined value - 2 clocks for the turnaround, selecting a value other than 00 adds extra cycles to this predefined value according to the selection	00 0 clock cycles. 01 1 clock cycle. 10 2 clock cycles. 11 3 clock cycles.
— 23	0	Reserved. Write to zero for future compatibility.	
ACT_PD_EXIT 22–20	0b001	**Active Power-Down Exit Timing (t_{XARD} and t_{XARDS})** Specifies how many clock cycles to wait between exit from active power-down and issuing a command. The default is one clock cycle.	000 Reserved 001 1 clock cycle. 010 2 clock cycles. 011 3 clock cycles. 100 4 clock cycles. 101 5 clock cycles. 110 6 clock cycles. 111 7 clock cycles.
— 19	0	Reserved. Write to zero for future compatibility.	
PRE_PD_EXIT 18–16	0b001	**Precharge Power-Down Exit Timing (t_{xp})** Specifies how many clock cycles to wait after exiting precharge power-down before issuing any command. The default is one clock cycle.	000 Reserved. 001 1 clock. 010 2 clock cycles. 011 3 clock cycles. 100 4 clock cycles. 101 5 clock cycles. 110 6 clock cycles. 111 7 clock cycles.
— 15–12	0	Reserved. Write to zero for future compatibility.	

Table 12-19. TIMING_CFG_0 Field Descriptions (Continued)

Bit	Reset	Description	Settings	
PDT_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.	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111	0 clock 1 clock 2 clock cycles. 3 clock cycles. 4 clock cycles. 5 clock cycles. 6 clock cycles. 7 clock cycles. 8 clock cycles. 9 clock cycles. 10 clock cycles. 11 clock cycles. 12 clock cycles. 13 clock cycles. 14 clock cycles. 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 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111	Reserved 1 clock 2 clock cycles. 3 clock cycles. 4 clock cycles. 5 clock cycles. 6 clock cycles. 7 clock cycles. 8 clock cycles. 9 clock cycles. 10 clock cycles. 11 clock cycles. 12 clock cycles. 13 clock cycles. 14 clock cycles. 15 clock cycles.

12.7.5 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	R/W			R/W				R	R/W			R/W			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	REFREC				—	WRREC			—	ACTTOACT			—	WRTORD		
Type	R/W				R	R/W			R	R/W			R	R/W		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TIMING_CFG_1 sets the number of clock cycles between various SDRAM control commands.

Table 12-20. TIMING_CFG_1 Field Descriptions

Bits	Reset	Description	Settings	
— 31	0	Reserved. Write to zero for future compatibility.		
PRETOACT 30–28	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.	000 001 010 011 100 101 110 111	Reserved. 1 clock 2 clock cycles. 3 clock cycles. 4 clock cycles. 5 clock cycles. 6 clock cycles. 7 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 must be programmed for proper operation of the DDR Controller.	0000 0001 0010 0011 0100 0101 0110 0111 … 1111	16 clock cycles. 17 clock cycles. 18 clock cycles. 19 clock cycles. 4 clock cycles. 5 clock cycles. 6 clock cycles. 7 clock cycles. 15 clock cycles.
— 23	0	Reserved. Write to zero for future compatibility.		
ACTTORW 22–20	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.	000 001 010 011 100 101 110 111	Reserved. 1 clock cycle. 2 clock cycles. 3 clock cycles. 4 clock cycles. 5 clock cycles. 6 clock cycles. 7 clock cycles.

Table 12-20. TIMING_CFG_1 Field Descriptions (Continued)

Bits	Reset	Description	Settings	
CASLAT 19–16	0	**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. CASLAT is used in conjunction with additive latency to obtain the read latency. 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 value must be programmed at initialization as described in the discussion of the DDR SDRAM Control Configuration Register (SCFG) on **page 12-46**. This field must be programmed for proper operation of the DDR Controller.	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111	Reserved. 1 clock cycle. 1.5 clock cycles. 2 clock cycles. 2.5 clock cycles. 3 clock cycles. 3.5 clock cycles. 4 clock cycles. 4.5 clock cycles. 5 clock cycles. 5.5 clock cycles. 6 clock cycles. 6.5 clock cycles. 7 clock cycles. 7.5 clock cycles. 8 clock cycles.
REFREC 15–12	0	**Refresh Recovery Time (t_{RFC})** Specifies the minimum number of clock cycles between a refresh command and an activate command. This 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 nanoseconds. This field is concatenated with TIMING_CFG_3[REFR] to obtain a 7-bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final 7-bit value of the refresh recovery. t_{RFC} = {TIMING_CFG_3 || REFREC} + 8 min. value = 8 clocks TIMING_CFG_3 = 0x0, REFREC = 0x0 max. value = 135 clocks TIMING_CFG_3 = 0x7, REFREC = 0xF = 112+15+8	0000 0001 0010 0011 ... 1111	8 clock cycles. 9 clock cycles. 10 clock cycles. 11 clock cycles. 23 clock cycles.
— 11	0	Reserved. Write to zero for future compatibility.		
WRREC 10–8	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. This field must be programmed for proper operation of the DDR Controller.	000 001 010 011 100 101 110 111	0 clock cycle. 1 clock cycle. 2 clock cycles. 3 clock cycles. 4 clock cycles. 5 clock cycles. 6 clock cycles. 7 clock cycles.
— 7	0	Reserved. Write to zero for future compatibility.		

Table 12-20. TIMING_CFG_1 Field Descriptions (Continued)

Bits	Reset	Description	Settings
ACTTOACT 6–4	0	**Activate-to-Activate Interval (t_{RRD})** Specifies the minimum number of clock cycles between an activate command and another activate command for a different logical bank in the same physical bank (chip select). The default is one clock cycle. This interval is calculated from the AC specifications of the SDRAM. This field must be programmed for proper operation of the DDR Controller.	000 Reserved. 001 1 clock cycle. 010 2 clock cycles. 011 3 clock cycles. 100 4 clock cycles. 101 5 clock cycles. 110 6 clock cycles. 111 7 clock cycles.
— 3	0	Reserved. Write to zero for future compatibility.	
WRTORD 2–0	0	**Last Write Data Pair to Read Command Interval (t_{WTR})** Specifies the minimum number of clock cycles between the last write data pair and the subsequent read command to the same physical bank. This field must be programmed for proper operation of the DDR Controller.	000 Reserved. 001 1 clock cycle. 010 2 clock cycles. 011 3 clock cycles. 100 4 clock cycles. 101 5 clock cycles. 110 6 clock cycles. 111 7 clock cycles.

12.7.6 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
	—	ALL			CPO					—	WR_LAT			—		
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**
	RD_TO_PRE			WRITE_DATA_DELAY			—	CKE_PLS			FOUR_ACT					
Type	R/W						R	R/W								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TIMING_CFG_2 sets the clock delay to data for writes and should be defined according to the system timing.

Table 12-21. TIMING_CFG_2 Bit Descriptions

Bit	Reset	Description	Settings
0	—	Reserved. Write to zero for future compatibility.	
ALL 30–28	0	**Additive Latency** The additive latency must be set to a value less than TIMING_CFG_1[ACTTORW]. This timing parameters applies to DDR2 only	000 0 clock cycles. 001 1 clock cycle. 010 2 clock cycles. 011 3 clock cycles. 100 4 clock cycles. 101 5 clock cycles. 110–111 Reserved.

Table 12-21. TIMING_CFG_2 Bit Descriptions (Continued)

Bit	Reset	Description	Settings
CPO 27–23	0	**$\overline{\text{MCAS}}$-to-Preamble Override** Defines the number of DRAM cycles between a read command and the time the corresponding DQS preamble is valid for the memory controller. For these decodings, read latency (RL) is equal to the $\overline{\text{CAS}}$ latency plus the additive latency. For CPO decodings other than 00000 and 11111, read latency is rounded up to the next integer value.	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 10110–11110 Reserved 11111 Automatic Calibration (recommended)
— 22	0	Reserved. Write to zero for future compatibility.	
WR_LAT 21–19	0	**Write Latency** Note that the total write latency for DDR2 is equal to WR_LAT + ADD_LAT. The Write Latency for DDR1 is 1. This field must be programmed for proper operation of the DDR Controller.	000 Reserved. 001 1 clock cycle. 010 2 clock cycles. 011 3 clock cycles. ... 111 7 clock cycles.
— 18–16	0	Reserved. Write to zero for future compatibility.	
RD_TO_PRE 15–13	0	**Read to Precharge (t_{RTP})** For DDR2, with a non-zero ADD_LAT value, takes a minimum of ADD_LAT + t_{RTP} cycles between read and precharge. For DDR1 must be set to 010. This field must be programmed for proper operation of the DDR Controller.	000 Reserved. 001 1 clock cycle. 010 2 clock cycles. 011 3 clock cycles. 100 4 clock cycles. 101–111 Reserved

Table 12-21. TIMING_CFG_2 Bit Descriptions (Continued)

Bit	Reset	Description	Settings
WRITE_ DATA_ DELAY 12–10	0	**Write Data Delay** Controls the amount of delay applied to the data and data strobes for writes. These bits allow 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 percent of a cycle or later than 125 percent of a cycle from the registration of a write command. This parameter is not defined in the SDRAM specification and should be defined according to the system timing.	000 0 clock delay 001 1/4 clock delay 010 1/2 clock delay 011 3/4 clock delay 100 1 clock delay 101 5/4 clock delay 110 3/2 clock delay 111 Reserved
— 9	0	Reserved. Write to zero for future compatibility.	
CKE_PLS 8–6	0	**Minimum CKE Pulse Width (t_{CKE})** Can be set to 001 for DDR1. This field must be programmed for proper operation of the DDR Controller.	000 Reserved. 001 1 clock cycle. 010 2 clock cycles. 011 3 clock cycles. 100 4 clock cycles. 101–111 Reserved
FOUR_ACT 5–0	0	**Window for Four Activates (t_{FAW}).** This is applied to DDR2 with eight logical banks only. Must be set to 0001 for DDR1	000000 Reserved. 000001 1 clock cycle. 000010 2 clock cycles. 000011 3 clock cycles. 000100 4 clock cycles. ... 010011 19 clock cycles. 010100 20 clock cycles. 010101–11111 Reserved.

12.7.7 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
	MEMEN	SREN	ECC_ EN	—		SDRAM_Type			—		DYN_ PWR_ MGMT	—	16BE	—	NCAP	—
Type	R/W			R		R/W			R		R/W	R	R/W	R	R/W	R
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	2T_EN	—											HSE	—	MHALT	BI
Type	R/W	R											R/W	R	R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_SDRAM_CFG enables the interface logic and specifies certain operating features such as self refreshing, error checking and correcting and dynamic power management. The default value is 0b010, designating DDR1 SDRAM.

Table 12-22. DDR_SDRAM_CFG Field Descriptions

Bits	Reset	Description	Settings
MEMEN 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 or soft-stop.	0 SDRAM self refresh is disabled during sleep or soft-stop. 1 SDRAM self refresh is enabled during sleep or soft-stop
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.
— 28–27	0	Reserved. Write to zero for future compatibility.	
SDRAM_Type 26–24	010	**Type of SDRAM Device** Specifies the type of device. The default value is 0b010 to designate DDR1 SDRAM.	000–001 Reserved. 010 DDR1 SDRAM. 011 DDR2 SDRAM. 100 - 111 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.
— 20	0	Reserved. Write to zero for future compatibility.	
16BE 19	0	**16-Bit Bus Enable**	0 32-bit bus is used. 1 16-bit bus is used.
— 18	0	Reserved. Write to zero for future compatibility.	
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.
— 16	0	Reserved. Write to zero for future compatibility.	
2T_EN 15	0	**2T Timing Enable** Enables/disabled 2T timing. When this bit is cleared, the DRAM command/address are held for only one cycle on the DRAM bus. When this bit is set, the DRAM command/address are held for two full clock cycles. on the DRAM bus for every DRAM transaction. However, the chip select is held only for the second cycle.	0 1T timing is used. 1 2T timing is enabled.
— 14–4	0	Reserved. Write to zero for future compatibility.	

Table 12-22. DDR_SDRAM_CFG Field Descriptions (Continued)

Bits	Reset	Description	Settings
HSE 3	0	**Half-Strength Drive Enable** Specifies whether the I/O drivers are configured to full strength or half strength. This bit applies only when automatic driver compensation is disabled and the software override for the driver strength is not used in the Global Utilities Register.	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.	
MHALT 1	0	**DDR Memory Controller Halt** When this bit is set, the memory controller does not accept any new transactions until the bit is cleared. This bit can be used when initialization is bypassed and the MODE REGISTER SET and EXTENDED MODE REGISTER SET commands are forced through software. This bit should be used carefully. Using this MHALT option can create congestion on the system interconnection and can cause hangs of the cores and other initiator.	0 DDR controller accepts new transactions. 1 DDR controller finishes any remaining transactions and then halts until software clears this bit.
BI 0	0	**Bypass Initialization** Specifies the conditions for initialization. When this bit is set, software is responsible for initializing memory through the SMCFG2 register. If software is initializing memory, the MHALT bit can be set to prevent the DDR controller from issuing transactions during the initialization sequence. For details on avoiding ECC errors in this mode, see the discussion of the DDR SDRAM Initialization Address Register on **page 12-51**.	0 DDR controller cycles through the initialization routine based on the value of SDRAM_Type. 1 Initialization routine is bypassed.

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

DDR_SDRAM_CFG_2 provides control configuration for the DDR controller in addition to that provided by DDR_SDRAM_CFG.

Table 12-23. DDR_SDRAM_CFG_2 Field Descriptions

Bit	Reset	Description	
FRC_SR 31	0	**Force Self Refresh**	0 Normal operating mode. 1 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** Defines the DQS mode - single ended or differential. Should be cleared for DDR1.	00 Only true DQS signals are used. 01 Differential DQS signals are used for DDR2 support. 10 Reserved. 11 Reserved.
— 25–23	0	Reserved. Write to zero for future compatibility.	
ODT_CFG 22–21	0	**ODT Configuration** Defines how ODT is driven to the on-chip I/O. See **Table 12-49** 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.	
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 SICFG[REFINT], must be programmed so that the maximum t_{ras} specification cannot be violated. For example, some DDR1 SDRAMs cannot use more than three posted refreshes because the required refresh interval can exceed the maximum constraint for t_{ras}.	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.
— 11–5	0	Reserved. Write to zero for future compatibility.	

Table 12-23. DDR_SDRAM_CFG_2 Field Descriptions (Continued)

Bit	Reset	Description	
D_INIT 4	0	**DRAM Data Initialization** Software sets this bit, and hardware clears it. When this bit is set, the value in the D_INIT register is used to initialize memory. If software sets this bit before the memory controller is enabled, the controller automatically initialize DRAM after it is enabled. During initialization known data is written to the entire memory space. This bit remains asserted until the initialization is complete. Hardware automatically clears this bit when initialization completes.	0 No data initialization, and no data initialization is scheduled. 1 The memory controller initializes memory when it is enabled.
— 3–0	0	Reserved. Write to zero for future compatibility.	

12.7.9 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 sets the values loaded into the DDR mode registers.

Table 12-24. DDR_SDRAM_MODE Bit Descriptions

Bit	Refresh	Description
ESDMODE 31–16	0	**Extended SDRAM Mode** Specifies the initial value loaded into the DDR SDRAM extended mode register. The range of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus during DDR SDRAM initialization, MA0 presents the LSB of ESDMODE, which corresponds to DDR_SDRAM_MODE bit 16. The MSB of the SDRAM extended mode register value must be stored at DDR_SDRAM_MODE bit 31.
SDMODE 15–0	0	**SDRAM Mode** Specifies the initial value loaded into the DDR SDRAM mode register. The range of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus during DDR SDRAM initialization, MA0 presents the LSB of SDMODE, which, in the little-endian convention used, corresponds to MODCFG bit 0. The MSB of the SDRAM mode register value must be stored at MODCFG 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), the memory controller ignores the corresponding bits of this field.

12.7.10 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 sets the values loaded into the DDR extended mode 2 and 3 registers (for DDR2).

Table 12-25. DDR_SDRAM_MODE_2 Bit Descriptions

Bit	Reset	Description
ESDMODE2 31–16	0	**Extended SDRAM Mode 2** Specifies the initial value loaded into the DDR SDRAM Extended 2 Mode Register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus during the DDR SDRAM initialization sequence, MA0 presents the LSB bit of ESDMODE2, which corresponds to DDR_SDRAM_MODE_2 bit 16. The MSB of the SDRAM extended mode 2 register value must be stored at DDR_SDRAM_MODE_2 bit 31.
ESDMODE3 15–0	0	**Extended SDRAM Mode 3** Specifies the initial value loaded into the DDR SDRAM Extended 3 Mode Register. The range of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus during DDR SDRAM initialization, MA0 presents the LSB of ESDMODE3, which corresponds to DDR_SDRAM_MODE_2 bit 0. The MSB of the SDRAM extended mode 3 register value must be stored at DDR_SDRAM_MODE_2 bit 15.

12.7.11 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
	MDEN	—		CSSEL	—	MDSEL			SETR	SETPRE	CKECTL		—			
Type	R/W	R		R/W	R	R/W							R			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	MDV															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_SDRAM_MD_CNTL allows software to force mode/extended mode register set commands to DRAM.

Note: Each command initiated through the DDR_SDRAM_MD_CNTL register should be initiated separately in the order required by the DDR SDRAM. If more than one command are initiated simultaneously the execution order is not guarantied and can cause malfunction of the DDR SDRAM.

Table 12-26. DDR_SDRAM_MD_CNTL Bit Descriptions

Bit	Reset	Description	Settings
MDEN 31	0	**Mode Enable** Indicates that valid data is ready to write to DRAM as a MODE REGISTER SET, EXTENDED MODE REGISTER SET, EXTENDED MODE REGISTER SET 2, or EXTENDED MODE REGISTER SET 3 command. Software sets this value, and hardware clears it after the command is issued. Note that MD_EN, SET_REF, and SET_PRE are mutually exclusive; they cannot be set at the same time	0 No MODE REGISTER SET, EXTENDED MODE REGISTER SET, EXTENDED MODE REGISTER SET 2, or EXTENDED MODE REGISTER SET 3 command to be issued. 1 Valid data in the register is ready to issue as a MODE REGISTER SET, EXTENDED MODE REGISTER SET, EXTENDED MODE REGISTER SET 2, or EXTENDED MODE REGISTER SET 3 command.
— 30–29	0	Reserved. Write to zero for future compatibility.	
CSSEL 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.	0 Chip select 0 is active. 1 Chip select 1 is active.
— 27	0	Reserved. Write to zero for future compatibility.	
MDSEL 26–24	0	**Mode Register Select.** Specifies the value to present to the memory bank address pins 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
SETR 23	0	**Set Refresh** Forces a refresh to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. Software sets this bit and hardware clears it when the command is issued. Note that 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.
SETPRE 22	0	**Set Precharge All** Forces a precharge all to be issued to the chip select specified by DDR_SDRAM_MD_CNTL[CSSEL]. Software sets this bit, and hardware clears it when the command is issued. Note that 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 A precharge all command is ready to issue.

Table 12-26. DDR_SDRAM_MD_CNTL Bit Descriptions (Continued)

Bit	Reset	Description	Settings
CKECTL 21–20	0	**Clock Enable Control** Software uses this bit to clear or set globally all CKE signals issued to DRAM. When software forces the value driven on CKE, that value continues to be forced until software clears the CKECTL bit. 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.
— 19–16	0	Reserved. Write to zero for future compatibility.	
MDV 15–0	0	**Mode Register Value** Specifies the value to present to the memory address pins of the DDR controller during the MODE REGISTER SET, EXTENDED MODE REGISTER SET, EXTENDED MODE REGISTER SET 2, or EXTENDED MODE REGISTER SET 3 command.	

12.7.12 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 sets the number of DRAM clock cycles between bank refreshes issued to the DDR SDRAMs. In addition, it specifies the number of DRAM cycles that a page is maintained after it is accessed.

Table 12-27. DDR_SDRAM_INTERVAL Bit Descriptions

Bit	Refresh	Description
REFINT 31–16	0	**Refresh Interval** Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR SDRAM CFG2 {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 12-27. DDR_SDRAM_INTERVAL Bit Descriptions (Continued)

Bit	Refresh	Description
— 15–14	0	Reserved. Write to zero for future compatibility.
BSTOPRE 13–0	0	**Precharge Interval** Specifies the duration (in memory bus clock cycles) that a page is retained after a DDR SDRAM access. The page open counter is loaded with BSTOPRE each time the page is accessed (including page hits). When the counter expires, the open page is closed with an SDRAM precharge bank command as soon as possible. If BSTOPRE has a value of zero, the DDR memory controller uses auto-precharge read and write commands rather than operating in page mode. This is called global auto-precharge mode.

12.7.13 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
	IVAL															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	IVAL															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_DATA_INIT provides the value to initialize memory if DDR_SDRAM_CFG_2[D_INIT] is set. When ECC is enabled, initializing all the available memory space can ensure that reads from unwritten addresses will not return ECC errors and interrupts.

Table 12-28. DDR_DATA_INIT Bit Descriptions

Bits	Reset	Description
IVAL 31–0	0	**Initialization Value** Specifies the initialization value for the DRAM if DDR_SDRAM_CFG_2[D_INIT] is set.

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

DDR_SDRAM_CLK_CNTL provides a source synchronous option, along with a 1/8 cycle clock adjustment.

Table 12-29. DDR_SDRAM_CLK_CNTL Bit Descriptions

Bit	Reset	Description	Settings
— 31–27	0	Reserved. Write to zero for future compatibility.	
CLK_ADJUST 26–23	0	**Clock Adjust** Specifies when the clock is launched in relationship to the address/command.	0000 Clock launched and aligned with address/command. 0001 Clock launched 1/8 applied cycle after address/command. 0010 Clock launched 1/4 applied cycle after address/command. 0011 Clock launched 3/8 applied cycle after address/command. 0100 Clock launched 1/2 applied cycle after address/command. 0101 Clock launched 5/8 applied cycle after address/command. 0110 Clock launched 3/4 applied cycle after address/command. 0111 Clock launched 7/8 applied cycle after address/command. 1000 Clock launched 1 applied cycle after address/command. 1001–1111Reserved.
— 22–0	0	Reserved. Write to zero for future compatibility.	

12.7.15 DDR SDRAM Initialization Address Register (DDR_INIT_ADDRESS)

DDR_INIT_ADDRESS DDR SDRAM Initialization Address Register Offset 0x0148

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

DDR_INIT_ADDRESS provides the address for the data strobe to data skew adjustment and automatic CAS to preamble calibration after power-on reset.

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 $\overline{\text{Reset}}$ is asserted after the DRAM enters Self-Refresh mode, memory is not initialized. Therefore this address should be written to avoid possible ECC errors when this address is accessed later.

Table 12-30. DDR_INIT_ADDRESS Bit Descriptions

Bit	Reset	Description	Settings
IADDR 31–0	0	**Initialization Address** Provides the address used for the data strobe to data skew adjustment and automatic CAS to preamble calibration at power-on reset. If used during initialization sequence, this address will be written during the initialization sequence.	

Note: Due to the specific implementation, the memory address programmed in this register should be shifted left one bit. For example, if the initialization address should be 0x4000_0010 the value written to SAx should be 0x8000_0020.

12.7.16 DDR SDRAM Initialization Address Enable Register (DDR_INIT_EN)

DDR_INIT_EN DDR SDRAM Initialization Address Enable Register Offset 0x014C

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

DDR_INIT_EN contains the enable bit to allow the usage of the address specified in DDR_INIT_ADDRESS register for data strobe to data skew adjustment and automatic CAS to preamble calibration after power-on reset.

Table 12-31. DDR_INIT_EN Bit Descriptions

Bit	Reset	Description	Settings
UIA 31	0	**Use Initialization Address** Enables the address used during initialization. When this bit is cleared, the address is the first valid address in the first enabled chip select.	0 Use the default address for the training sequence as calculated by the memory controller. 1 Use the initialization address programmed in DDR_INIT_ADDRESS.
— 30–0	0	Reserved. Write to zero for future compatibility.	

12.7.17 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
	IPID															
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**
	IPMJ								IPMN							
Type	R															
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

DDR_IP_REV1 provides read-only fields with the IP block ID, along with major and minor revision information.

Table 12-32. DDR_IP_REV1 Bit Descriptions

Bit	Reset	Description
IPID 31–16	0x0002	**IP Block ID** For the DDR controller.
IPMJ 15–8	0x02	**Major Revision**
IPMN 7–0	0x00	**Minor Revision**

12.7.18 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
	—								IPINIT							
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

DDR_IP_REV2 provides read-only fields with the IP block integration and configuration options.

Table 12-33. DDR_IP_REV2 Bit Descriptions

Bit	Reset	Description
— 31–24	0	Reserved. Write to zero for future compatibility.
IPINT 23–16	0	**IP Block Integration Options**

Table 12-33. DDR_IP_REV2 Bit Descriptions (Continued)

Bit	Reset	Description
— 15–8	0	Reserved. Write to zero for future compatibility.
IPCFG 7–0	0	**IP Block Configuration Options**

12.7.19 DDR SDRAM Memory Data Path Error Injection Mask High Register (DDR_ERR_INJECT_HI)

DDR_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
	EMSB								—							
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**
	ELSB								—							
Type	R/W								R							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_ERR_INJECT_HI is used to inject ECC errors for 32- and 16-bit SDRAM data bus interfaces.

Table 12-34. DDR_ERR_INJECT_HI Bit Descriptions

Bit	Reset	Description
EMSB 31–24	0	**Error Injection Mask for Bits 0–7** Tests ECC by forcing errors on the high bytes of the data path. Setting a bit causes the corresponding data path bit to be inverted during memory bus writes.
— 23–16	0	Reserved. Write to zero for future compatibility.
ELSB 15–8	0	**Error Injection Mask for Bits 8–15** Tests ECC by forcing errors on the high bytes of the data path. Setting a bit causes the corresponding data path bit to be inverted during memory bus writes.
— 7–0	0	Reserved. Write to zero for future compatibility.

12.7.20 DDR SDRAM Memory Data Path Error Injection Mask Low Register (DDR_ERR_INJECT_LO)

DDR_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
	EIMSB								—							
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**
	EILSB								—							
Type	R/W								R							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_ERR_INJECT_LO is used to inject ECC errors only for 32-bit SDRAM data bus interfaces. It has no effect in 16-bit mode.

Table 12-35. DDR_ERR_INJECT_LO Bit Descriptions

Bit	Reset	Description
EIMSB 31–24	0	**Error Injection Mask for Bits 16–23** Tests ECC by forcing errors on the low bytes of the data path. Setting a bit causes the corresponding data path bit to be inverted during memory bus writes.
— 23–16	0	Reserved. Write to zero for future compatibility.
EILSB 15–8	0	**Error Injection Mask for Bits 24–31** Tests ECC by forcing errors on the low bytes of the data path. Setting a bit causes the corresponding data path bit to be inverted during memory bus writes.
— 7–0	0	Reserved. Write to zero for future compatibility.

12.7.21 DDR SDRAM Memory Data Path Error Injection Mask ECC Register (DDR_ERR_INJECT)

DDR_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
	—															
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**
	—							EIEN	EEIM							
Type	R							R/W								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_ERR_INJECT sets the ECC mask, enables errors to be written to ECC memory, and allows the ECC byte to mirror the most significant data byte.

Table 12-36. DDR_ERR_INJECT Bit Descriptions

Bit	Reset	Description	Settings
— 31–9	0	Reserved. Write to zero for future compatibility.	
EIEN 8	0	**Error Injection Enable** Enables/disables injection of errors.	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.	

12.7.22 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
	CDHMSB								—							
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**
	CDHLSB								—							
Type	R/W								R							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAPTURE_DATA_HI stores the high bytes of the read data path during error capture. This register is used to read captured errors for 32-bit and 16-bit SDRAM data bus interfaces.

Table 12-37. CAPTURE_DATA_HI Bit Descriptions

Bit	Reset	Description
CDHMSB 31–24	0	**Error Capture High Data Path** Captures bits 0–7 of the data path when errors are detected.
— 23–16	0	Reserved. Write to zero for future compatibility.
CDHISB 15–8	0	**Error Capture High Data Path** Captures bits 8–15 of the data path when errors are detected.
— 7–0	0	Reserved. Write to zero for future compatibility.

12.7.23 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
	CDLMSB								—							
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**
	CDLLSB								—							
Type	R/W								R							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAPTURE_DATA_LO stores the low bytes of the read data path during error capture. This register is used to read captured error data only for 32-bit SDRAM data bus interfaces. Its values are undefined in 16-bit mode.

Table 12-38. CAPTURE_DATA_LO Bit Descriptions

Bit	Reset	Description
CDLMSB 31–24	0	**Error Capture Low Data Path** Captures bits 16–23 of the data path when errors are detected.
— 23–16	0	Reserved. Write to zero for future compatibility.
CDLISB 15–8	0	**Error Capture Low Data Path** Captures bits 24–31 of the data path when errors are detected.
— 7–0	0	Reserved. Write to zero for future compatibility.

12.7.24 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															
Reset	0	0	0	0	0	0	0	0	0	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								R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAPTURE_ECC stores the ECC syndrome bits on the data bus when an error is detected.

Table 12-39. CAPTURE_ECC Bit Descriptions

Bit	Reset	Description
— 31–8	0	Reserved. Write to zero for future compatibility.
ECE 7–0	0	**Error Capture ECC** Captures the ECC bits on the data path when errors are detected. (0:7)

12.7.25 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														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—								ACE	—			MBE	SBE	—	MSE
Type	R								W1C	R			W1C		R	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: W1C - Write 0b1 to clear the bit, writing 0b0 as no effect.

ERR_DETECT stores the detection bits for multiple memory errors, single- and multiple-bit ECC errors, calibration error, and memory select errors. Each bit is cleared when software writes a value of 1 to it. System software can determine the type of memory error by examining the contents of this register. If an error is disabled with ERR_DISABLE, the corresponding error is never detected or captured in ERR_DETECT.

Table 12-40. 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.
— 30–8	0	Reserved. Write to zero for future compatibility.	
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.
— 6–4	0	Reserved. Write to zero for future compatibility.	
MBE 3	0	**Multiple-Bit Error** Indicates whether a multiple-bit error was detected. This bit is cleared by software writing a 1 to it.	0 Multiple-bit error not detected. 1 Multiple-bit error detected.

Table 12-40. ERR_DETECT Bit Descriptions (Continued)

Bit	Reset	Description	Settings
SBE 2	0	**Single-Bit ECC Error** Indicates whether the number of single-bit ECC errors detected is greater or equal to the threshold set in ERR_SBE[SBET]. This bit is cleared by software writing a 1 to it.	0 The number of errors is not less than the threshold. 1 The number of errors is greater or equal to 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.

12.7.26 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															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—								ACED	—			MBED	SBED	—	MSED
Type	R								R/W	R			R/W		R	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERR_DISABLE selectively disables the DDR controller error detection circuitry. Disabled errors are not detected or reported.

Table 12-41. ERR_DISABLE Bit Descriptions

Bit	Reset	Description	Settings
— 31–8	0	Reserved. Write to zero for future compatibility.	
ACED 7	0	**Automatic Calibration Error Disable** Enables/disables automatic calibration errors detection	0 Calibration errors enabled. 1 Calibration errors disabled.
— 6–4	0	Reserved. Write to zero for future compatibility.	
MBED 3	0	**Multiple-Bit ECC Error Disable** Enables/disables multiple-bit ECC errors detection. When this bit is cleared, multiple-bit errors are detected if CCFG[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.

Table 12-41. ERR_DISABLE Bit Descriptions (Continued)

Bit	Reset	Description	Settings
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.

12.7.27 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															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—								ACEE	—			MBEE	SBEE	—	MSEE
Type	R								R/W	R			R/W		R	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERR_INT_EN enables ECC interrupts 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 12-42. ERR_INT_EN Bit Descriptions

Bit	Reset	Description	Settings
— 31–8	0	Reserved. Write to zero for future compatibility.	
ACEE 7	0	**Automatic Calibration Error Interrupt Enable** Specifies whether automatic calibration errors generate interrupts.	0 Calibration errors cannot generate interrupts. 1 Calibration errors generate interrupts.
— 6–4	0	Reserved. Write to zero for future compatibility.	
MBEE 3	0	**Multiple-Bit ECC Error Interrupt Enable** Specifies whether multiple-bit ECC errors generate interrupts.	0 Multiple-bit ECC errors cannot generate interrupts. 1 Multiple-bit ECC errors generate interrupts.
SBEE 2	0	**Single-Bit ECC Error Interrupt Enable** Specifies whether single-bit ECC errors generate interrupts.	0 Single-bit ECC errors cannot generate interrupts. 1 Single-bit ECC errors generate interrupts.
— 1	0	Reserved. Write to zero for future compatibility.	
MSEE 0	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.

12.7.28 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		—											
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**
	—		TTYP		—											VLD
Type	R		R/W		R											R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAPTURE_ATTRIBUTES sets attributes for errors including type, size, source, and so on.

Table 12-43. CAPTURE_ATTRIBUTES Bit Descriptions

Bit	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
BNUM 29–28	0	**Data Beat Number** Captures the data beat number for the detected error. This bit is relevant only for ECC errors.	
— 27–14	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.	

12.7.29 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 holds the 31 LSBs of a transaction address when a DDR ECC error is detected.

Note: Due to a specific implementation, the address values captured in CAPTURE_ADDRESS register will be shifted left by one bit.

For example, if the erroneous address is 0x4000_0000 the value in the register will be 0x8000_0000. If the erroneous address is 0x41F2_E5A1 the value in the register will be 0x83E5_CB42.

Table 12-44. CAPTURE_ADDRESS Bit Descriptions

Bit	Reset	Description	Settings
CADDR 31–0	0	**Captured Address** Captures the 31 LSBs of the transaction address when an error is detected. The read value should be right shifted by the core to obtain the correct address value.	

12.7.30 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 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 12-45. ERR_SBE Bit Descriptions

Bit	Reset	Description
— 31–24	0	Reserved. Write to zero for future compatibility.
SBET 23–16	0	**Single-Bit Error Threshold** Establishes the number of single-bit errors that must be detected before an error condition is reported.
— 15–8	0	Reserved. Write to zero for future compatibility.
SBEC 7–0	0	**Single-Bit Error Counter** Indicates the number of single-bit errors detected and corrected since the last error report. If single-bit error reporting is enabled, an error is reported when this value equals SBET. SBEC is automatically cleared when the threshold value is reached.

12.7.31 DDR SDRAM DDR Status (DDR_STOP_STATUS)

DDR_STOP_STATUS DDR SDRAM DDR Status Offset 0x1000

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**
	NIS				PIS				—						IMEM	SACK
Type	R															
Reset	1	1	0	0	1	1	0	0	0	0	0	0	0	0	1	0

DDR_STOP_STATUS reports memory controller readiness to enter the clock stop status as well as the values obtained by the automatic driver calibration sequence. This is a read-only register.

Table 12-46. DDR Status Field Descriptions DDR_STOP_STATUS Bit Descriptions

Bit	Reset	Description	Settings
— 31–16	0	Reserved. Write to zero for future compatibility.	
NIS 15–12	1100	**Driver N-Impedance Status** Results of driver calibration. Reset value 0xC represents the nominal impedance	
PID 11–8	1100	**Driver P-Impedance Status** Results of driver calibration. Reset value 0xC represents the nominal impedance	
— 7–2	0	Reserved. Write to zero for future compatibility.	

Table 12-46. DDR Status Field Descriptions (Continued)DDR_STOP_STATUS Bit

Bit	Reset	Description	Settings
IMEM 1	1	**Memory Controller Idle** Indicates whether the memory controller is idle. Reset value represents that the controller is in IDLE state after reset	0 DDR controller is busy. 1 DDR controller is idle. - default after reset
SACK 0	0	**DDR Controller Stop Status** Indicates whether the memory controller accepts a stop request. When this bit is cleared, there is either no stop request or the memory controller does not accept the request. When this bit is set, the DDR controller accepts the stop request and is ready for the clock stop.	0 DDR controller not ready to stop. 1 DDR controller ready to stop.

12.7.32 DDDR SDRAM Power Control Register (DDR_PWR)

DDR_PWR DDR SDRAM Power Control Register Offset 0x1004

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**
	—														DQSL	STOP
Type	R														R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_PWR enables the memory controller to reduce power by entering a clock stop mode.

Table 12-47. DDR_PWR Bit Descriptions

Bit	Reset	Description	Settings
— 31–2	0	Reserved. Write to zero for future compatibility.	
DQSL 1	0	**Drain Queues for Sleep** Indicates whether the input queue should be drained before the device enters self refresh mode. This bit affects only the software method for entering self refresh.	0 Queues should not be drained. 1 Queues should be drained.
STOP 0	0	**Stop Request to Memory Controller** Specifies whether a stop request is sent to the memory controller.	0 No stop request. 1 Stop request to memory controller.

12.7.33 DDR SDRAM MDIC Output Enable Control Register (MDIC_OE_CONT)

MDIC_OE_CONT DDR SDRAM MDIC Output Enable Control Register Offset 0x1008

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**
	—														1OE	0OE
Type	R														R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MDIC_OE_CONT controls the use of the MDIC pins. There a dedicated control bit for each MDIC pin.

Table 12-48. MDIC_OE_CONT Bit Descriptions

Bits	Reset	Description	Setting
— 31–2	0	Reserved. Write to zero for future compatibility.	
MDIC1_OE 1	0	**Output Enable for MDIC_1** Forces the output enable to be set only if TERM_OCD_ODT_CONT[DCOV] is also set.	0 - if TERM_OCD_ODT_CONT[DCOV] is set, force MDIC1_OE to disable 1- if TERM_OCD_ODT_CONT[DCOV] is set, force MDIC1_OE to enable
MDIC0_OE 0	0	**Output Enable for MDIC_0** Forces the output enable to be set only if TERM_OCD_ODT_CONT[DCOV] is also set.	0 - if TERM_OCD_ODT_CONT[DCOV] is set, force MDIC0_OE to disable 1 - if TERM_OCD_ODT_CONT[DCOV] is set, force MDIC0_OE to enable

12.7.34 DDR SDRAM Termination, OCD, and ODT Control Register (TERM_OCD_ODT_CONT)

TERM_OCD_ODT_CONT DDR SDRAM Termination, OCD, and ODT Control Register — Offset 0x100C

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**
	—	DBD	DCOV	DCEN	TV		TOEN	ODT	PCI				NCI			
Type	R	R/W														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TERM_OCD_ODT_CONT controls the strength of the I/O drivers and the termination value for the incoming signals.

Table 12-49. TERM_OCD_ODT_CONT Bit Descriptions

Bit	Description	Setting
— 31–15	Reserved. Write to zero for future compatibility.	
DBD 14	**Disable Bit De-Skew** Memory controller will not use the bit de-skew functionality if this pin is tied high. The end user must decide if bit de-skew is required for the product	0 Bit Deskew during initialization enabled. 1 Bit Deskew during initialization disabled.
DCOV 13	**Driver Comp Override** Enable for the software override of the driver impedance.	0 Software override for driver impedance disabled. 1 Software override for driver impedance enabled.
DCEN 12	**Driver Comp Enable** Enable bit for the driver impedance hardware calibration.	0 Hardware calibration of driver impedance disabled. 1 Hardware calibration of driver impedance enabled.
TV 11–10	**Term Value** Value on the TERMSEL pins during a software override.	
TOEN 9	**Termsel Override Enable** Software override enable for the TERMSEL (on-chip ODT) pins on the I/O signals.	0 Software override for on chip termination disabled. 1 Software override for on chip termination enabled.
ODT 8	**ODT** Indicates the on-chip ODT termination value. A value of 0 indicates 75 ohm termination, and a value of 1 indicates 150 ohm termination.	0 75 ohm termination 1 150 ohm termination
PCI 7–4	**P Channel Impedance** Value for the p-impedance if jdcp_drvr_comp_override is asserted.	
NCI 3–0	**N Channel Impedance** Value for the n-impedance if jdcp_drvr_comp_override is asserted.	

12.7.35 DDR SDRAM Clock Ratio Control Register (CLK_RATIO_CONT)

CLK_RATIO_CONT DDR SDRAM Clock Ratio Control Register Offset 0x1010

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**
	—														INIT	—
Type	R														R/W	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CLK_RATIO_CONT controls the system clock-to-memory controller clock decoupling unit. For proper operation, the relation between the two clock cycles is defined here. Take care to configure both control bits properly to prevent undefined results.

Table 12-50. CLK_RATIO_CONT Bit Descriptions

Bit	Reset	Description	Settings
31–2	0	Reserved. Write to zero for future compatibility.	
INIT 1	0	Specifies whether the CLASS128 clock frequency is higher than the frequency of the memory controller clock.	0 Memory controller clock frequency is higher than CLASS128 frequency. 1 CLASS128 clock frequency is higher than memory controller clock frequency.
0	0	Reserved. Write to zero for future compatibility.	

Interrupt Handling 13

The MSC8144E interrupt system is optimized for a multi-processing environment and performs the following functions:

- Routes each of the interrupt sources to each of the extended SC3400 cores thus allowing:
  - Flexible resource allocations as well as for a symmetrical or a non-symmetrical application architecture.
  - Provides a core-to-core signaling mechanism by virtual interrupt generation.
- Allows for the enabling/disabling of each interrupt source per core.

The MSC8144E supports both internal and external interrupt sources as well as allowing for the generation of an interrupt to external devices.

There are three device level interrupt handlers in the MSC8144E:

1. *Global interrupt controller*. Allows for the generation of virtual interrupt requests (VIRQ) as well as virtual non-maskable interrupts (VNMI) towards the cores as well as generates interrupts to external devices.
2. *General configuration block*. Concentrates and routes rare and debug interrupts to the SC3400 cores.
3. *Embedded programmable interrupt controller (EPIC)*. Concentrates all the interrupt directed at the associated core and dispatches the highest priority interrupt to the SC3400 core. Although there are various interrupts in the system designated as non-interruptible interrupts (NMIs), you must program them to be non-interruptible in the EPIC. The is typically done by the MSC8144E boot program.

Note: See the *MSC8144 SC3400 DSP Core Subsystem Reference Manual* for details about the EPIC. The manual is available with a signed non-disclosure agreement. Contact your local Freescale dealer or sales representative for more information.

Note: The QUICC Engine module also includes an interrupt controller that handles interrupts within the module for the dual-RISC processor system. For details about that interrupt controller and how to program it, refer to **Chapter 17**, *QUICC Engine Module*.

13.1 Global Interrupt Controller (GIC)

The GIC generates 18 VIRQs and 4 VNMIs 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.

13.1.1 Virtual Interrupt Generation

A virtual interrupt is generated via a write access to the Virtual Interrupt Generation Register (VIGR) by one of the SC3400 cores or an external host CPU. **Table 13-1** describes the destination of the supported VIRQs.

Table 13-1. VIRQ Description

VIRQ Num	Destination
VIRQ_0	Connected to Virtual Interrupt 0 at SC3400
VIRQ_1	Connected to Virtual Interrupt 1 at SC3400
VIRQ_2	Connected to Virtual Interrupt 2 at SC3400
VIRQ_3	Connected to Virtual Interrupt 3 at SC3400
VIRQ_4	Connected to Virtual Interrupt 4 at SC3400
VIRQ_5	Connected to Virtual Interrupt 5 at SC3400
VIRQ_6	Connected to Virtual Interrupt 6 at SC3400
VIRQ_7	Connected to Virtual Interrupt 7 at SC3400
VIRQ_8	Connected to Virtual Interrupt 8 at SC3400
VIRQ_9	Connected to Virtual Interrupt 9 at SC3400
VIRQ_10	Connected to Virtual Interrupt 10 at SC3400
VIRQ_11	Connected to Virtual Interrupt 11 at SC3400
VIRQ_12	Connected to Virtual Interrupt 12 at SC3400
VIRQ_13	Connected to Virtual Interrupt 13 at SC3400
VIRQ_14	Connected to Virtual Interrupt 14 at SC3400
VIRQ_15	Connected to Virtual Interrupt 15 at SC3400
VIRQ_16	Connected to $\overline{\text{INT_OUT}}$ (see **13.2.2**, *External Interrupts*)
VIRQ_17	Used for the generation of $\overline{\text{NMI_OUT}}$ (see **13.2.2**, *External Interrupts*)
VIRQ_18	Connected to the QUICC Engine module 0 input interrupt
VIRQ_19	Connected to the QUICC Engine module 1 input interrupt

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.

13.1.2 Virtual $\overline{\text{NMI}}$ Generation

The global interrupt controller includes a Virtual NMI system that generates four NMI signals. An NMI is generated by a write access of once of the SC3400 cores or by external host CPU. The virtual NMI system does not have a status register.

13.2 General Configuration Block

The general configuration block performs services for rare and debug interrupts generated throughout the MSC8144E before they reach the SC3400 EPICs. These services include:

- Generating ORed interrupt signals towards the SC3400 cores (see **Section 13.2.1**).
- Providing an interrupt enable bit for each interrupt source for each SC3400 core (see **Section 13.5.2**, *General Interrupt Configuration*, on page 13-15).
- Providing a status bit for each interrupt source. These bits are shared for all the SC3400 cores (see **Section 13.5.2**, *General Interrupt Configuration*, on page 13-15).

13.2.1 Interrupt Groups

The general configuration block generates 4 interrupts based on the groups of ORed interrupts described in **Table 13-2**. The general configuration block also routes and records the status of the VNMIs generated by the GIC.

Table 13-2. General Configuration Block Interrupt Sources

TDM	Debug	General	Watch Dog Timer
TDM 0 Rx error	CLASS 0 overrun	M2_0 ECC error	Watch Dog Timer 0
TDM 0 Tx error	CLASS 0 watchpoint	M2_1 ECC error	Watch Dog Timer 1
TDM 1 Rx error	CLASS 1 overrun	M2_2 ECC error	Watch Dog Timer 2
TDM 1 Tx error	CLASS 1 watchpoint	M2_3 ECC error	Watch Dog Timer 3
TDM 2 Rx error	CLASS 1 error		Watch Dog Timer 4
TDM 2 Tx error	CLASS 2 overrun		
TDM 3 Rx error	CLASS 2 watchpoint	Parity error from TDM[0–3]	
TDM 3 Tx error	L2 ICache initiator CLASS overrun	Parity error from TDM[4–7]	
TDM 4 Rx error	L2 ICache initiator CLASS watchpoint	QUICC Engine module DRAM ECC error	
TDM 4 Tx error	L2 ICache target CLASS overrun	QUICC Engine module IMEM ECC error	
TDM 5 Rx error	L2 ICache target CLASS watchpoint		
TDM 5 Tx error	Performance Monitor all	DMA error	
TDM 6 Rx error		DDR interrupt	
TDM 6 Tx error		PCI all	
TDM 7 Rx error		OCeaN to MBus	
TDM 7 Tx error			

13.2.2 External Interrupts

The MSC8144E allows a number of external interrupt inputs to be multiplexed with the GPIO signals to enable external devices to interrupt the cores (see **Chapter 23**, *GPIO*). There are also dedicated external interrupt pins.

Note: All external $\overline{\text{IRQ}}$ signals are multiplexed options of the associated GPIO ports.

Table 13-3 summarizes all the external interrupt inputs in the MSC8144E.

Table 13-3. MSC8144E External Interrupt Pins

Name	GPIO	Direction
$\overline{\text{NMI}}$	N/A	In
$\overline{\text{NMI_OUT}}$	N/A	Out
$\overline{\text{INT_OUT}}$	N/A	Out
$\overline{\text{IRQ0}}$	GPIO16	In
$\overline{\text{IRQ1}}$	GPIO21	In
$\overline{\text{IRQ2}}$	GPIO30	In
$\overline{\text{IRQ3}}$	GPIO31	In
$\overline{\text{IRQ4}}$	GPIO22	In
$\overline{\text{IRQ5}}$	GPIO23	In
$\overline{\text{IRQ6}}$	GPIO24	In
$\overline{\text{IRQ7}}$	GPIO29	In
$\overline{\text{IRQ8}}$	GPIO14	In
$\overline{\text{IRQ9}}$	GPIO15	In
$\overline{\text{IRQ10}}$	GPIO4	In
$\overline{\text{IRQ11}}$	GPIO5	In
$\overline{\text{IRQ12}}$	GPIO6	In
$\overline{\text{IRQ13}}$	GPIO7	In
$\overline{\text{IRQ14}}$	GPIO8	In
$\overline{\text{IRQ15}}$	GPIO25	In

$\overline{\text{NMI_OUT}}$ is asserted whenever one of the following conditions is fulfilled:

- $\overline{\text{NMI}}$ is asserted.
- VIRQ_17 is asserted.
- The system watchdog timer interrupt (SWT4) is asserted.

These conditions are not maskable by software. $\overline{\text{INT_OUT}}$ is asserted when VIRQ_16 is asserted.

13.2.3 Interrupt Handling

The MSC8144E interrupts sources can be grouped in to four basic types:

1. Interrupts that represent a single interrupt source and are routed directly to the cores (for example, the TDM0 Rx first threshold interrupt).
2. Interrupts that represent multiple interrupt sources and are routed directly to the cores (for example, all I^2C interrupts).
3. Interrupts that represent a single interrupt source and are routed to the cores via the general configuration block (for example, virtual non maskable interrupt 0).
4. Interrupts that represent multiple interrupt sources and are routed to the cores via the General Configuration Block (for example, parity error from TDM[0–3] interrupt).

Figure 13-1 outlines the flow for handling the various types of interrupts.

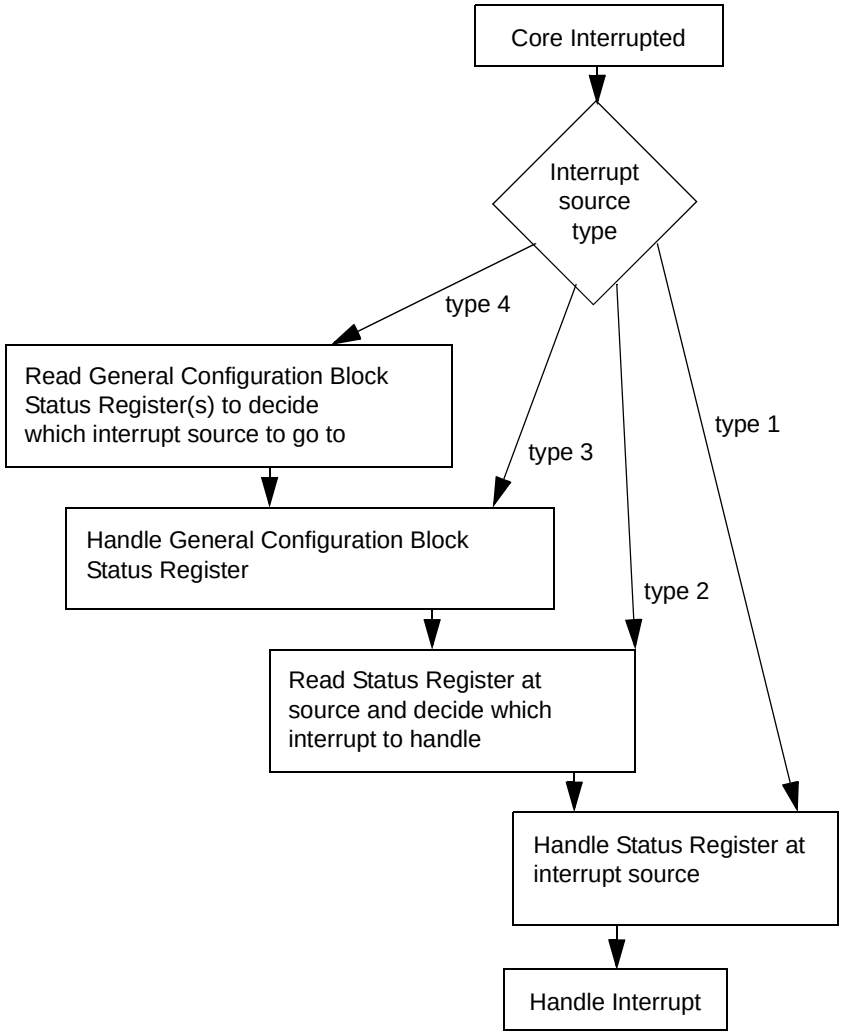


Figure 13-1. Interrupt Handling Flow

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 MSC8144 SC3400 DSP core subsystem reserves the first 34 interrupt sources for internal use, leaving 222 available interrupt sources. The MSC8144E does not implement all of these possible sources.

Table 13-4 describes the interrupt capabilities (level/edge), default priority, and index for each interrupt source. Interrupts sources routed via the general configuration block are marked as having no index and their routing is set by the EPIC (see the *MSC8144 SC3400 DSP Core Subsystem Reference Manual* for details).

Table 13-4. MSC8144E Interrupt Table

Interrupt Description	IRQ index	Level	Edge
TDM			
TDM 0 Rx first threshold	48	+	+
TDM 0 Rx second threshold	49	+	+
TDM 0 Tx first threshold	50	+	+
TDM 0 Tx second threshold	51	+	+
TDM 1 Rx first threshold	52	+	+
TDM 1 Rx second threshold	53	+	+
TDM 1 Tx first threshold	54	+	+
TDM 1 Tx second threshold	55	+	+
TDM 2 Rx first threshold	56	+	+
TDM 2 Rx second threshold	57	+	+
TDM 2 Tx first threshold	58	+	+
TDM 2 Tx second threshold	59	+	+
TDM 3 Rx first threshold	60	+	+
TDM 3 Rx second threshold	61	+	+
TDM 3 Tx first threshold	62	+	+
TDM 3 Tx second threshold	63	+	+
TDM 4 Rx first threshold	64	+	+
TDM 4 Rx second threshold	65	+	+
TDM 4 Tx first threshold	66	+	+
TDM 4 Tx second threshold	67	+	+
TDM 5 Rx first threshold	68	+	+
TDM 5 Rx second threshold	69	+	+
TDM 5 Tx first threshold	70	+	+
TDM 5 Tx second threshold	71	+	+
TDM 6 Rx first threshold	72	+	+
TDM 6 Rx second threshold	73	+	+

Table 13-4. MSC8144E Interrupt Table (Continued)

Interrupt Description	IRQ index	Level	Edge
TDM 6 Tx first threshold	74	+	+
TDM 6 Tx second threshold	75	+	+
TDM 7 Rx first threshold	76	+	+
TDM 7 Rx second threshold	77	+	+
TDM 7 Tx first threshold	78	+	+
TDM 7 Tx second threshold	79	+	+
Serial RapidIO			
Serial RapidIO message in 0	84	+	—
Serial RapidIO message in 1	85	+	—
Serial RapidIO message out 0	86	+	—
Serial RapidIO message out 1	87	+	—
Serial RapidIO doorbell in 1	88	+	—
Serial RapidIO doorbell out 0	89	+	—
Serial RapidIO general error	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 1 SMII system error	108	+	—
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 13-4. MSC8144E 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	+	—
Ethernet 2 SMII system error	126	+	—
ATM			
ATM global buffer pool busy	127	+	—
ATM global 0	128	+	—
ATM global 1	129	+	—
ATM global 2	130	+	—
ATM global 3	131	+	—
ATM global red line	132	+	—
ATM interrupt queue 0 overflow	133	+	—
ATM interrupt queue 1 overflow	134	+	—
ATM interrupt queue 2 overflow	135	+	—
ATM interrupt queue 3 overflow	136	+	—
ATM transmit rate underrun	137	+	—
QUICC Engine Subsystem			
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 13-4. MSC8144E 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	—	+

Table 13-4. MSC8144E Interrupt Table (Continued)

Interrupt Description	IRQ index	Level	Edge
Virtual Interrupt 10	187	—	+
Virtual Interrupt 11	188	—	+
Virtual Interrupt 12	189	—	+
Virtual Interrupt 13	190	—	+
Virtual Interrupt 14	191	—	+
Virtual Interrupt 15	192	—	+
Virtual Non Maskable Interrupt 0	193	+	—
Virtual Non Maskable Interrupt 1	194	+	—
Virtual Non Maskable Interrupt 2	195	+	—
Virtual Non Maskable Interrupt 3	196	+	—
I^2C			
I^2C all	213	+	—
External $\overline{IRQ}$s			
$\overline{IRQ0}$	226	+	+
$\overline{IRQ1}$	227	+	+
$\overline{IRQ2}$	228	+	+
$\overline{IRQ3}$	229	+	+
$\overline{IRQ4}$	230	+	+
$\overline{IRQ5}$	231	+	+
$\overline{IRQ6}$	232	+	+
$\overline{IRQ7}$	233	+	+
$\overline{IRQ8}$	234	+	+
$\overline{IRQ9}$	235	+	+
$\overline{IRQ10}$	236	+	+
$\overline{IRQ11}$	237	+	+
$\overline{IRQ12}$	238	+	+
$\overline{IRQ13}$	239	+	+
$\overline{IRQ14}$	240	+	+
$\overline{IRQ15}$	241	+	+
$\overline{NMI}$	242	+	+
General Configuration Block			
ORed TDM interrupts	243	+	—
ORed Debug Interrupts	244	+	—
ORed General Interrupts	245	+	—
ORed Watch Dog Timer Interrupts	246	+	—

Table 13-4. MSC8144E Interrupt Table (Continued)

Interrupt Description	IRQ index	Level	Edge
OCN DMA			
Channel 0 Interrupt	248	+	—
Channel 1 Interrupt	249	+	—
Channel 2 Interrupt	250	+	—
Channel 3 Interrupt	251	+	—
General Hardware Interrupt	252	+	—

13.4 Restrictions

Some interrupts can cause the core to deadlock. These interrupts can occur when the core issues a single read towards a peripheral and the read triggers an interrupt. The deadlock occurs because the interrupt signal reaches the core before the actual data. As a result, the core jumps to the interrupt handler and upon returning from handling the interrupt, reissues the read request.

Table 13-5 summarizes the problematic scenarios and resulting restrictions on the user.

Table 13-5. Restrictions Listed by Interrupt Source

EPIC Index	Event	Problematic scenario	Restriction
226	IRQ0	External source generates a IRQ/NMI as a response to a Core read access from the DDR/PCI	The peripheral must not generate IRQ/NMI as a response to a Core read access from the DDR/PCI
227	IRQ1		
228	IRQ2		
229	IRQ3		
230	IRQ4		
231	IRQ5		
232	IRQ6		
233	IRQ7		
234	IRQ8		
235	IRQ9		
236	IRQ10		
237	IRQ11		
238	IRQ12		
239	IRQ13		
240	IRQ14		
241	IRQ15		
242	NMI		

Table 13-5. Restrictions Listed by Interrupt Source

EPIC Index	Event	Problematic scenario	Restriction
244	CLASS1 error	DSP core is reading an illegal address during the debug phase.	Use a debug instruction at the end of the ISR.
	CLASS0 watch-point	Setting a hardware watchpoint due to a single read access of the DSP core during debug phase	
	CLASS1 watch-point		
	L2IC_CLASS_M watch-point		
	L2IC_CLASS_S watch-point		
245	Parity error from TDM[0-7]	DSP core is reading an address in TDM PRAM with soft error.	Read the failing address and correct the soft error.
	QUICC Engine DRAM/IMEM ECC error	DSP core is reading an address in the QUICC Engine subsystem DRAM/IMEM with soft error.	ISR must reset the device.
	DMA error	DSP core is reading an address in DMA PRAM (during debug mode) with soft error.	Read the failing address (channel number) and correct the soft error.
	DDR single / double ECC error	DSP core is reading an address in DDR with soft error	Read the address from the DDR controller and correct the soft error. For single ECC errors, the threshold can be set to more than one error.
	DDR access to an address that does not hit any DDR configuration space.	DSP core is reading an address in DDR that does not hit any DDR configuration space (during debug phase).	Read the problematic address from the DDR controller. Use a debug instruction at the end of the ISR.
	PCI error	DSP core is reading an address in PCI with no response or with parity error.	Use a debug instruction at the end of the ISR.
	Performance monitor	DSP core is reading an address in L2 ICache and it is a hit, PM counter is set to report a single L2 ICache hit access.	Set the threshold of the counter of the PM that counts hit events in the L2 ICache to 2 or more.

13.5 Programming Model

The MSC8144E interrupt program model includes configuration of the global interrupt controller and the general configuration block interrupt registers.

Note: See the *MSC8144 SC3400 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)
- Virtual NMI Generation Register (VNMIGR)

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 19.

Table 13-6. 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
	—	—	—	—	—	—	—	—	—	—	—	—	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.

13.5.1.3 Virtual NMI Generation Register (VNMIGR)

VNMIGR Virtual NMI Generation Register ‘Offset 0x10

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
	—	—	—	—	—	—	VNMINUM		—	—	—	—	—	—	—	—
Type	W															

VNMIGR generates a virtual non-maskable interrupt to the SC3400 cores. The VNMI generated corresponds with the number written to VNMINUM. A read from VNMIGR returns all zeros.

Table 13-7. VNMIGR Bit Descriptions

Name	Description
— 31–10	Reserved. Write to zero for future compatibility.
VNMINUM 9–8	**Virtual Non Maskable Interrupt Number** Asserts the corresponding VNMI.
— 7–0	Reserved. Write to zero for future compatibility.

13.5.2 General Interrupt Configuration

The general configuration block resides in a 128-byte address space (see **Chapter 9**, *Memory Map*) and has the following interrupt configuration registers:

- General Interrupt Register 1 (GIR1)
- General Interrupt Enable Registers 1 for Cores 0–3 (GIER1_[0–3])
- General Interrupt Register 2 (GIR2)
- General Interrupt Enable Registers 2 for Cores 0–3 (GIER2_[0–3])
- General Interrupt Register 3 (GIR3)
- General Interrupt Enable Registers 3 for Cores 0–3 (GIER3_[0–3])

Note: The general interrupt configuration registers use a base address of: 0xFFF78000.

13.5.2.1 General Interrupt Register 1 (GIR1)

GIR1 General Interrupt Register 1 Offset 0x40

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**
	—				VNMI_3	VNMI_2	VNMI_1	VNMI_0
Type	R/W				Sticky W1C			
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—				M2_3_ECC	M2_2_ECC	M2_1_ECC	M2_0_ECC
Type	R/W				Sticky W1C			
Reset	0	0	0	0	0	0	0	0

GIR1 includes interrupt status of ECC events of M2. Those bits are sticky and cleared by writing 1. The GIR1 is 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 13-8. GIR1 Bit Descriptions

Name	Description	Settings
— 31–12	Reserved. Write to zero for future compatibility.	
VNMI_3 11	**Virtual NMI 3** Asserted when VNMI_3 is activated	0 Interrupt not asserted 1 Interrupt asserted

Table 13-8. GIR1 Bit Descriptions (Continued)

Name	Description	Settings
VNMI_2 10	**Virtual NMI 2** Asserted when VNMI_2 is activated	0 Interrupt not asserted 1 Interrupt asserted
VNMI_1 9	**Virtual NMI 1** Asserted when VNMI_1 is activated	0 Interrupt not asserted 1 Interrupt asserted
VNMI_0 8	**Virtual NMI 0** Asserted when VNMI_0 is activated	0 Interrupt not asserted 1 Interrupt asserted
— 7–4	Reserved. Write to zero for future compatibility.	
M2_3_ECC 3	**M2 Block 3 ECC Error Interrupt** Asserted when ECC error is reported by M2_3	0 Interrupt not asserted 1 Interrupt asserted
M2_2_ECC 2	**M2 Block 2 ECC Error Interrupt** Asserted when ECC error is reported by M2_2	0 Interrupt not asserted 1 Interrupt asserted
M2_1_ECC 1	**M2 Block 1 ECC Error Interrupt** Asserted when ECC error is reported by M2_1	0 Interrupt not asserted 1 Interrupt asserted
M2_0_ECC 0	**M2 Block 0 ECC Error Interrupt** Asserted when ECC error is reported by M2_0	0 Interrupt not asserted 1 Interrupt asserted

13.5.2.2 General Interrupt Enable Register 1 for Cores 0–3 (GIER1_[0–3])

GIER1_0 ‘General Interrupt Enable Register 1 for Cores 0–3 Offset 0x44
GIER1_1 Offset 0x48
GIER1_2 Offset 0x4C
GIER1_3 Offset 0x50

Bit	30	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
	—				M2_3_ECC_EN	M2_2_ECC_EN	M2_1_ECC_EN	M2_0_ECC_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIER1_[0–3] includes interrupt enable bits of ECC events of M2 for cores 0–3. 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 13-9. GIER1_n Bit Descriptions

Name	Description	Settings
— 31–4	Reserved. Write to zero for future compatibility.	
M2_3_ECC_EN 3	**M2 Block 3 ECC Error Enable**	0 Interrupt disabled 1 Interrupt enabled
M2_2_ECC_EN 2	**M2 Block 2 ECC Error Enable**	0 Interrupt disabled 1 Interrupt enabled
M2_1_ECC_EN 1	**M2 Block 1 ECC Error Enable**	0 Interrupt disabled 1 Interrupt enabled
M2_0_ECC_EN 0	**M2 Block 0 ECC Error Enable**	0 Interrupt disabled 1 Interrupt enabled

13.5.2.3 General Interrupt Register 2 (GIR2)

GIR2 General Interrupt Register 2 Offset 0x54

Bit	31	30	29	28	27	26	25	24
	—	—	SWT4	SWT3	SWT2	SWT1	SWT0	OCN_ERR
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	PCI_ERR	DDR_ERR	DMA_ERR	—	CE_IECC	CE_DECC	TDM_P1ECC	TDM_P0ECC
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	TDM7_TERR	TDM7_RERR	TDM6_TERR	TDM6_RERR	TDM5_TERR	TDM5_RERR	TDM4_TERR	TDM4_RERR
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	TDM3_TERR	TDM3_RERR	TDM2_TERR	TDM2_RERR	TDM1_TERR	TDM1_RERR	TDM0_TERR	TDM0_RERR
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIR2 includes interrupt status of some events within MSC8144E that are rare. Those bits are not sticky but only sample the events. The GIR2 register is reset on a hard reset event. All bits are cleared on reset.

Table 13-10. GIR2 Bit Descriptions

Name	Description	Settings
— 31–30	Reserved. Write to zero for future compatibility.	

Table 13-10. GIR2 Bit Descriptions

Name	Description	Settings
SWT4 29	**Software Watchdog Timer 4 Interrupt** Reflects SWT 4 interrupt	0 Interrupt not asserted 1 Interrupt asserted
SWT3 28	**Software Watchdog Timer 3 Interrupt** Reflects SWT 3 interrupt	0 Interrupt not asserted 1 Interrupt asserted
SWT2 27	**Software Watchdog Timer 2 Interrupt** Reflects SWT 2 interrupt	0 Interrupt not asserted 1 Interrupt asserted
SWT1 26	**Software Watchdog Timer 1 Interrupt** Reflects SWT 1 interrupt	0 Interrupt not asserted 1 Interrupt asserted
SWT0 25	**Software Watchdog Timer 0 Interrupt** Reflects SWT 0 interrupt	0 Interrupt not asserted 1 Interrupt asserted
OCN_ERR 24	**OCeaN-to-MBus Error Interrupt** Reflects OCeaN error interrupt	0 Interrupt not asserted 1 Interrupt asserted
PCI_ERR 23	**PCI Error Interrupt** Reflects PCI error interrupt	0 Interrupt not asserted 1 Interrupt asserted
DDR_ERR 22	**DDR Error Interrupt** Reflects DDR error interrupt	0 Interrupt not asserted 1 Interrupt asserted
DMA_ERR 21	**DMA Error Interrupt** Reflects DMA error interrupt	0 Interrupt not asserted 1 Interrupt asserted
— 20	Reserved. Write to zero for future compatibility.	
QE_IECC 19	**QUICC Engine module IMEM ECC Error Interrupt** Reflects ECC error interrupt of the QUICC Engine module IMEM	0 Interrupt not asserted 1 Interrupt asserted
QE_DECC 18	**QUICC Engine module DRAM ECC Error Interrupt** Reflects ECC error interrupt of the CE DRAM	0 Interrupt not asserted 1 Interrupt asserted
TDM_P1ECC 17	**TDM[4–7] Parity Error Interrupt** Reflects parity error interrupt of TDM4, TDM5, TDM6 or TDM7	0 Interrupt not asserted 1 Interrupt asserted
TDM_P0ECC 16	**TDM[0–3] Parity Error Interrupt** Reflects parity error interrupt of TDM0, TDM1, TDM2 or TDM3	0 Interrupt not asserted 1 Interrupt asserted
TDM7_TERR 15	**TDM7 Transmit Error Interrupt** Reflects TDM7 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM7_RERR 14	**TDM7 Receive Error Interrupt** Reflects TDM7 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM6_TERR 13	**TDM6 Transmit Error Interrupt** Reflects TDM6 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM6_RERR 12	**TDM6 Receive Error Interrupt** Reflects TDM6 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM5_TERR 11	**TDM5 Transmit Error Interrupt** Reflects TDM5 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM5_RERR 10	**TDM5 Receive Error Interrupt** Reflects TDM5 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM4_TERR 9	**TDM4 Transmit Error Interrupt** Reflects TDM4 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM4_RERR 8	**TDM4 Receive Error Interrupt** Reflects TDM4 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted

Table 13-10. GIR2 Bit Descriptions

Name	Description	Settings
TDM3_TERR 7	**TDM3 Transmit Error Interrupt** Reflects TDM3 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM3_RERR 6	**TDM3 Receive Error Interrupt** Reflects TDM3 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM2_TERR 5	**TDM2 Transmit Error Interrupt** Reflects TDM2 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM2_RERR 4	**TDM2 Receive Error Interrupt** Reflects TDM2 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM1_TERR 3	**TDM1 Transmit Error Interrupt** Reflects TDM1 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM1_RERR 2	**TDM1 Receive Error Interrupt** Reflects TDM1 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM0_TERR 1	**TDM0 Transmit Error Interrupt** Reflects TDM0 Transmit error interrupt	0 Interrupt not asserted 1 Interrupt asserted
TDM0_RERR 0	**TDM0 Receive Error Interrupt** Reflects TDM0 Receive error interrupt	0 Interrupt not asserted 1 Interrupt asserted

13.5.2.4 General Interrupt Enable Register 2 for Cores 0–3 (GIER2_[0–3])

GIER2_0 General Interrupt Enable Register 2 for Cores 0–3 Offset 0x58
GIER2_1 Offset 0x5C
GIER2_2 Offset 0x60
GIER2_3 Offset 0x64

Bit	31	30	29	28	27	26	25	24
	—	—	SWT4_EN	SWT3_EN	SWT2	SWT1_EN	SWT0_EN	OCN_ERR_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PCI_ERR_EN	DDR_ERR_EN	DMA_ERR_EN	—	CE_IECC_EN	CE_DECC_EN	TDM_P1ECC_EN	TDM_P0ECC_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TDM7_TERR_EN	TDM7_RERR_EN	TDM6_TERR_EN	TDM6_RERR_EN	TDM5_TERR_EN	TDM5_RERR_EN	TDM4_TERR_EN	TDM4_RERR_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TDM3_TERR_EN	TDM3_RERR_EN	TDM2_TERR_EN	TDM2_RERR_EN	TDM1_TERR_EN	TDM1_RERR_EN	TDM0_TERR_EN	TDM0_RERR_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIER2_[0–3] include interrupt enable bits for cores 0–3 for some events that rarely occur. The GIER2_[0–3] are 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 13-11. GIER2_x Bit Descriptions

Name	Description	Settings
— 31–30	Reserved. Write to zero for future compatibility.	
SWT4_EN 29	**SWT 4 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT3_EN 28	**SWT 3 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT2_EN 27	**SWT 2 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT1_EN 26	**SWT 1 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT0_EN 25	**SWT 0 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
OCN_ERR_EN 24	**OCeaN Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
PCI_ERR_EN 23	**PCI Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
DDR_ERR_EN 22	**DDR Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
DMA_ERR_EN 21	**DMA Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
— 20	Reserved. Write to zero for future compatibility.	
CE_IECC_EN 19	**ECC Error Interrupt of the CE IMEM Enable**	0 Interrupt disabled 1 Interrupt enabled
CE_DECC_EN 18	**ECC Error Interrupt of the CE DRAM Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM_P1ECC_EN 17	**Parity Error Interrupt of TDM[4–7] Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM_P0ECC_EN 16	**Parity Error Interrupt of TDM[0–3] Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM7_TERR_EN 15	**TDM7 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM7_RERR_EN 14	**TDM7 Receive Error Interrupt Enable/Disable**	0 Interrupt disabled 1 Interrupt enabled
TDM6_TERR_EN 13	**TDM6 Transmit Error Interrupt Enable/Disable**	0 Interrupt disabled 1 Interrupt enabled
TDM6_RERR_EN 12	**TDM6 Receive Error Interrupt Enable/Disable**	0 Interrupt disabled 1 Interrupt enabled
TDM5_TERR_EN 11	**TDM5 Transmit Error Interrupt Enable/Disable**	0 Interrupt disabled 1 Interrupt enabled

Table 13-11. GIER2_x Bit Descriptions

Name	Description	Settings
TDM5_RERR_EN 10	**TDM5 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM4_TERR_EN 9	**TDM4 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM4_RERR_EN 8	**TDM4 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM3_TERR_EN 7	**TDM3 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM3_RERR_EN 6	**TDM3 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM2_TERR_EN 5	**TDM2 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM2_RERR_EN 4	**TDM2 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM1_TERR_EN 3	**TDM1 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM1_RERR_EN 2	**TDM1 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM0_TERR_EN 1	**TDM0 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM0_RERR_EN 0	**TDM0 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled

13.5.2.5 General Interrupt Register 3 (GIR3)

GIR3 General Interrupt Register 3 Offset 0x68

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**
	—	—	—	—	PM	L2ICS_WP	L2ICS_OV	L2ICM_WP
Type	R/W							
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	L2ICM_OV	CLS2_WP	CLS2_OV	CLS1_ERR	CLS1_WP	CLS1_OV	CLS0_WP	CLS0_OV
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIR3 includes interrupt status of some debug/profiling events within MSC8144E. 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 13-12. GIR2 Bit Descriptions

Name	Description	Settings
— 31–12	Reserved. Write to zero for future compatibility.	
PM 11	**Performance Monitor Interrupt** Reflects the performance monitor interrupt	0 Interrupt not asserted 1 Interrupt asserted
L2ICS_WP 10	**L2 ICache Target CLASS Watchpoint Interrupt** Reflects L2 ICache target CLASS watch-point interrupt	0 Interrupt not asserted 1 Interrupt asserted
L2ICS_OV 9	**L2 ICache Target CLASS Overrun Interrupt** Reflects L2 ICache target Class overrun interrupt	0 Interrupt not asserted 1 Interrupt asserted
L2ICM_WP 8	**L2 ICache Initiator CLASS Watchpoint Interrupt** Reflects L2 ICache initiator Class watchpoint interrupt	0 Interrupt not asserted 1 Interrupt asserted
L2ICM_OV 7	**L2 ICache Initiator CLASS Overrun Interrupt** Reflects L2 ICache initiator Class overrun interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS2_WP 6	**CLASS2 Watchpoint Interrupt** Reflects CLASS2 watchpoint interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS2_OV 5	**CLASS2 Overrun Interrupt** Reflects CLASS2 overrun interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS1_ERR 4	**CLASS1 Error Interrupt** Reflects CLASS1 Error Interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS1_WP 3	**CLASS1 Watchpoint Interrupt** Reflects CLASS1 watchpoint interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS1_OV 2	**CLASS1 Overrun Interrupt** Reflects CLASS1 overrun interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS0_WP 1	**CLASS0 Watchpoint Interrupt** Reflects Class0 watchpoint interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS0_OV 0	**CLASS0 Overrun Interrupt** Reflects CLASS0 overrun interrupt	0 Interrupt not asserted 1 Interrupt asserted

13.5.2.6 General Interrupt Enable Register 3 for Cores 0–3 (GIER3_[0–3])

GIER3_0 General Interrupt Enable Register 3 for Cores 0–3 Offset 0x6C
GIER3_1 Offset 0x70
GIER3_2 Offset 0x74
GIER3_3 Offset 0x78

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
	—	—	—	—	PM_EN	L2ICS_WP_EN	L2ICS_OV_EN	L2ICM_WP_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	L2ICM_OV_EN	CLS2_WP_EN	CLS2_OV_EN	CLS1_ERR_EN	CLS1_WP_EN	CLS1_OV_EN	CLS0_WP_EN	CLS0_OV_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIER3_[0–3] include interrupt enable bits for cores 0–3 for debug/profiling events within MSC8144E. GIER3_[0–3] 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 13-13. GIER2_[0–3] Bit Descriptions

Name	Description	Settings
— 31–30	Reserved. Write to zero for future compatibility.	
PM_EN 11	**Performance Monitor Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
L2ICS_WP_EN 10	**L2 ICache Target CLASS Watchpoint Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
L2ICS_OV_EN 9	**L2 ICache Target CLASS Overrun Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
L2ICM_WP_EN 8	**L2 ICache Initiator CLASS Watchpoint Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
L2ICM_OV_EN 7	**L2 ICache Initiator CLASS Overrun Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS2_WP_EN 6	**CLASS2 Watchpoint Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS2_OV_EN 5	**CLASS2 Overrun Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled

Table 13-13. GIER2_[0–3] Bit Descriptions

Name	Description	Settings
CLS1_ERR_EN 4	**CLASS1 Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS1_WP_EN 3	**CLASS1 Watchpoint Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS1_OV_EN 2	**CLASS1 Overrun Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS0_WP_EN 1	**CLASS0 Watchpoint Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS0_OV_EN 0	**CLASS0 Overrun Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled

Direct Memory Access (DMA) Controller 14

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 external initiator through the RapidIO or PCI using buffer descriptors (BDs). All channels are capable of complex data movement and advanced transaction chaining. Operations such as descriptor fetches and block transfers are initiated by each of the sixteen channels. Full duplex operation allows the DMA controller to read data from one target and store it in its internal memory while concurrently writing another buffer to another target. This capability can be used extensively when data is read from the M3 memory and written into the M2 memory. The bidirectional DMA controller reads from one of the CLASS target ports while writing to the second one. The DMA controller supports smart arbitration algorithms such as round-robin and a timer-based mechanism using an earliest deadline first (EDF) algorithm. The DMA controller also supports a Debug mode and profiling for application development and testing. **Figure 14-1** shows the VCOP block diagram.

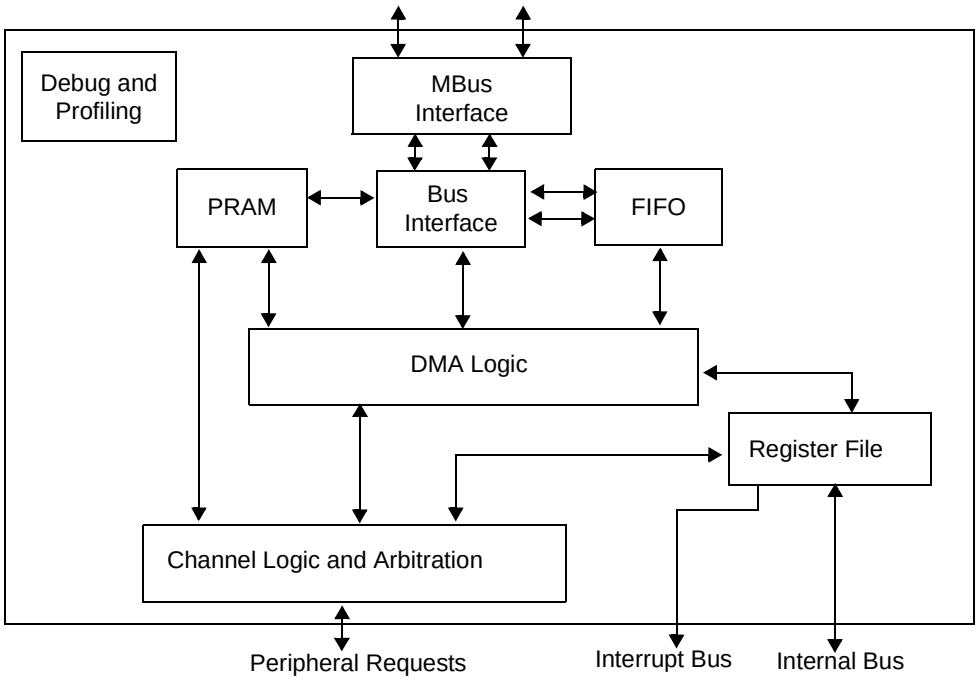


Figure 14-1. DMA Controller Block Diagram

14.1 Operating Modes

The DMA controller supports to modes of operation:

- *Functional mode*. Each of the data transactions can be executed on each of the MBus ports. The VCOP supports memory to memory data transfers.
- *Debug mode*. The VCOP enters the debug by external debug request. Once the VCOP enters the debug mode, it holds its requests to the MBus ports. The internal logic in the VCOP masks the channels.
  - MBus is in debug mode and each of its ports gracefully stops its transaction.
  - Channel logic in debug mode. The arbitration mechanism masks all channels requests. The last serviced channel gets is fully serviced.

14.2 Buffer Types

The DMA channel parameter RAM (PRAM) is accessible to the DMA controller and the port interface and includes a parity mechanism. Each channel has one dedicated PRAM line. The external BD is fetched into the PRAM the first time the channel wins during arbitration.

When a buffer is activated, the DMA controller generates a bus transaction with a maximum size as described in the buffer descriptor BTSZ field and decrements BD_SIZE accordingly. The address can increment or freeze. When BD_SIZE reaches zero, the channel takes one of the following actions:

- Shuts down (simple buffer)
- Reinitializes itself (cyclic buffer)
- Reinstalls its size (incremental buffer)
- Switches to another buffer (chained-buffers)
- Any combination of the preceding
- Updates the multi-dimension parameters (multi dimension-buffers)

The sections that follow provide examples of several types of buffers. The BD_ATTR fields listed for each example are only those that do not have zero values.

14.2.1 One-Dimensional Simple Buffer

A simple channel is a buffer that closes when BD_SIZE reaches zero. It is defined by clearing the BD_ATTR[CONT] field to zero (see **Table 14-29,** *BD_ATTR Field Descriptions*, on page 14-44). **Figure 14-2** shows an example simple buffer.

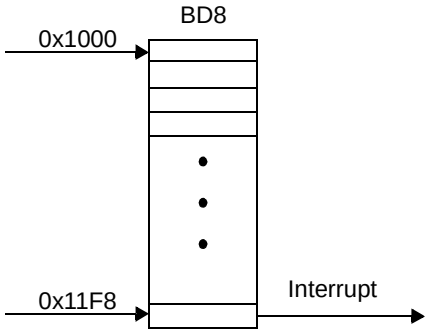


Figure 14-2. One-Dimensional Simple Buffer

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.

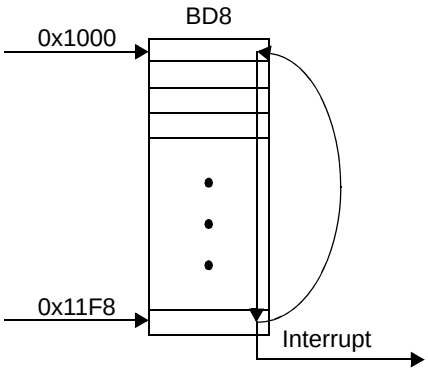


Figure 14-3. One-Dimensional Cyclic Buffer

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.

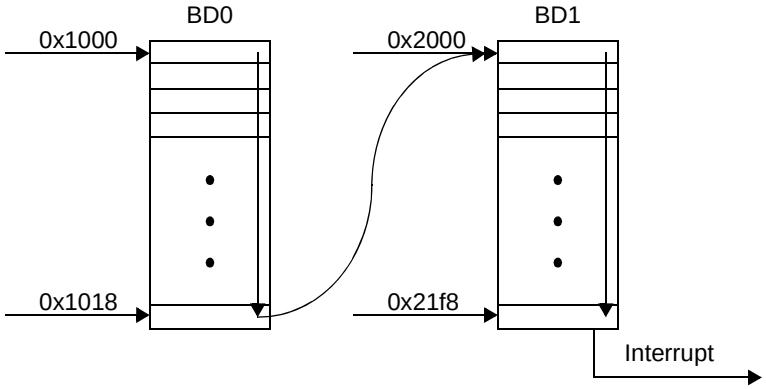


Figure 14-4. One-Dimensional Chained Buffer

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.

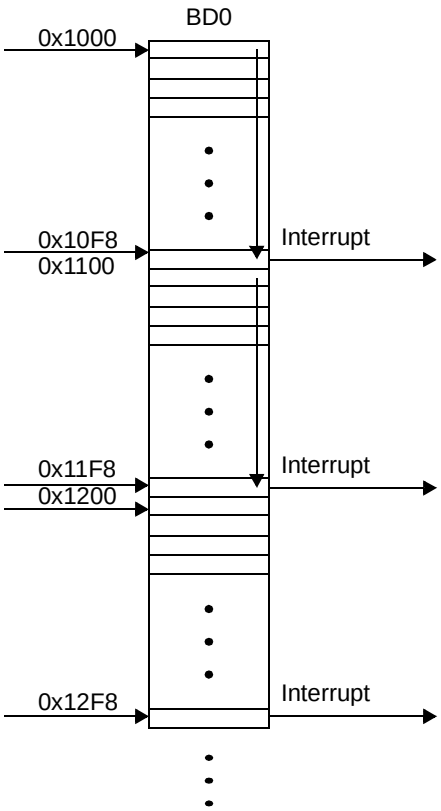


Figure 14-5. One-Dimensional Incremental Buffer

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**.

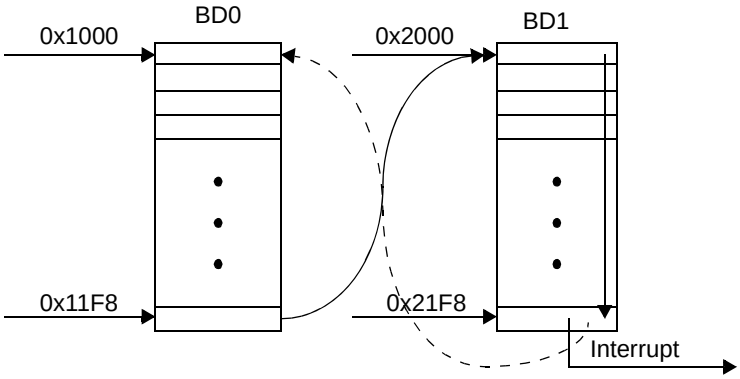


Figure 14-6. Dual Cyclic Buffers

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.

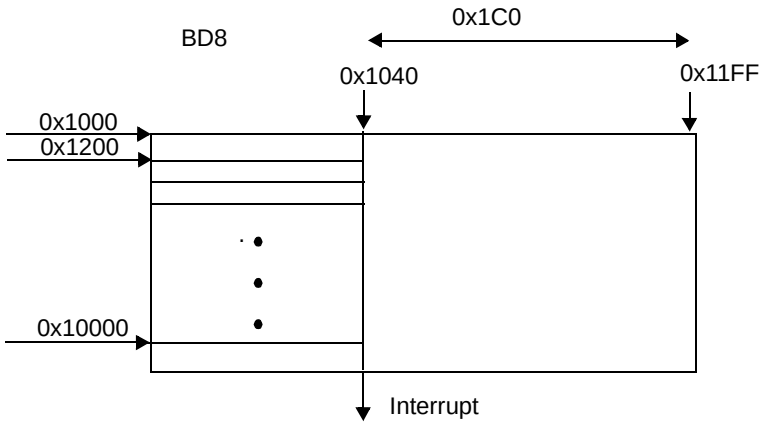


Figure 14-7. Two-Dimensional Simple Buffer

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 two's complement. The parameters of the fourth dimension must be cleared to zero. **Figure 14-8** shows a three-dimensional simple buffer.

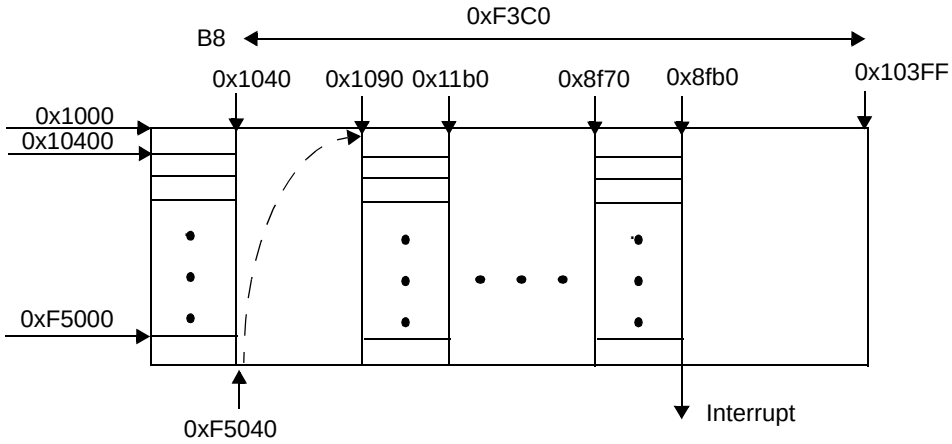


Figure 14-8. Three-Dimensional Simple Buffer

Table 14-7 shows the configuration of a simple buffer designated as channel BD8. A 0x40000 (0x100 × 10 × 40) three-dimensional block is read from address 0x1000. The basic buffer is 0x40 byte. The offset between each 0x40 byte transaction is 0xF3C0 (0x10400 – 0x1040). The second dimension parameter is 0x10. The offset between each two-dimensional buffers is –0xF3FB0 (0x1090 – 0xF5040). 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	–0xF3FB0	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 two's complement. **Figure 14-9** shows an example four-dimensional simple buffer.

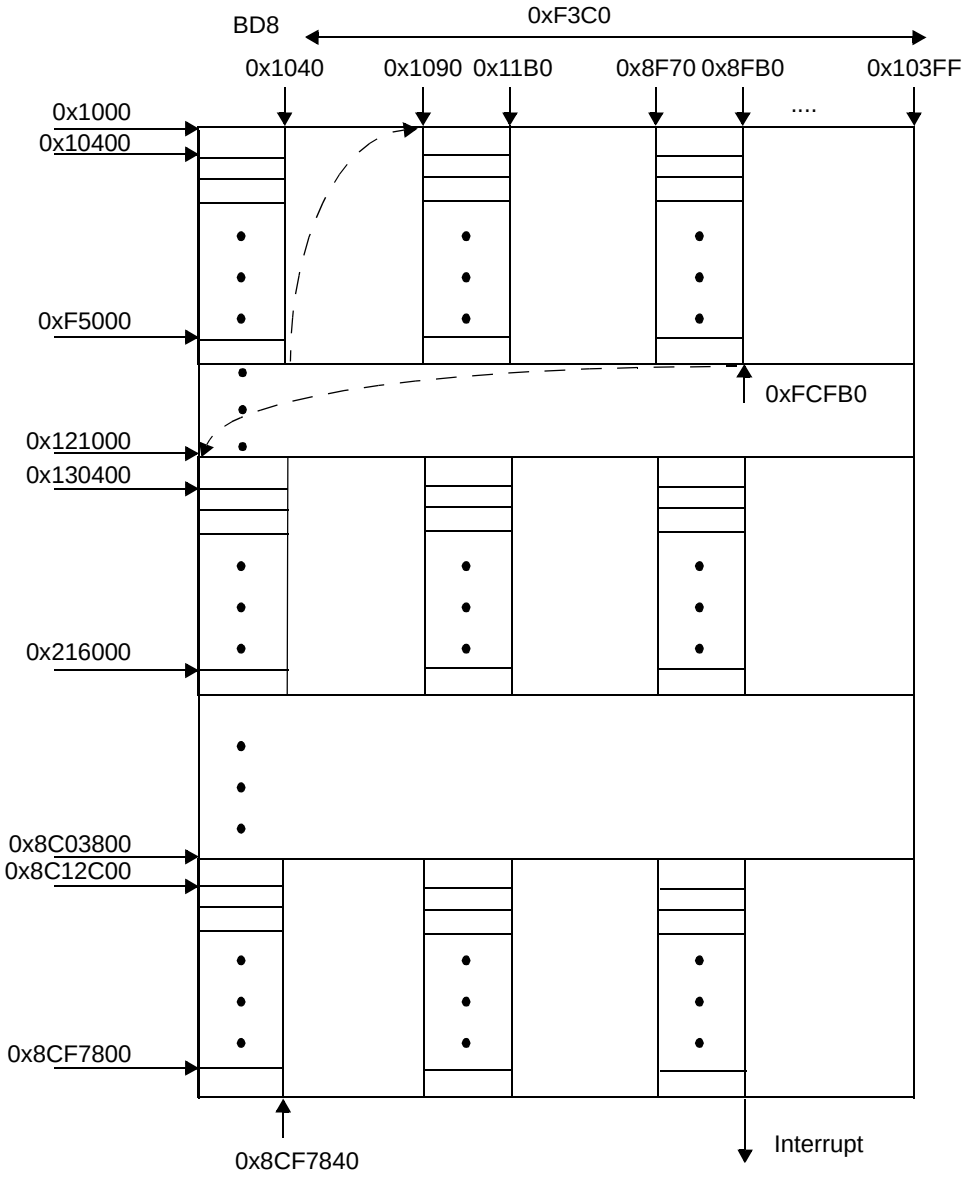


Figure 14-9. Four-Dimensional Simple Buffer

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.

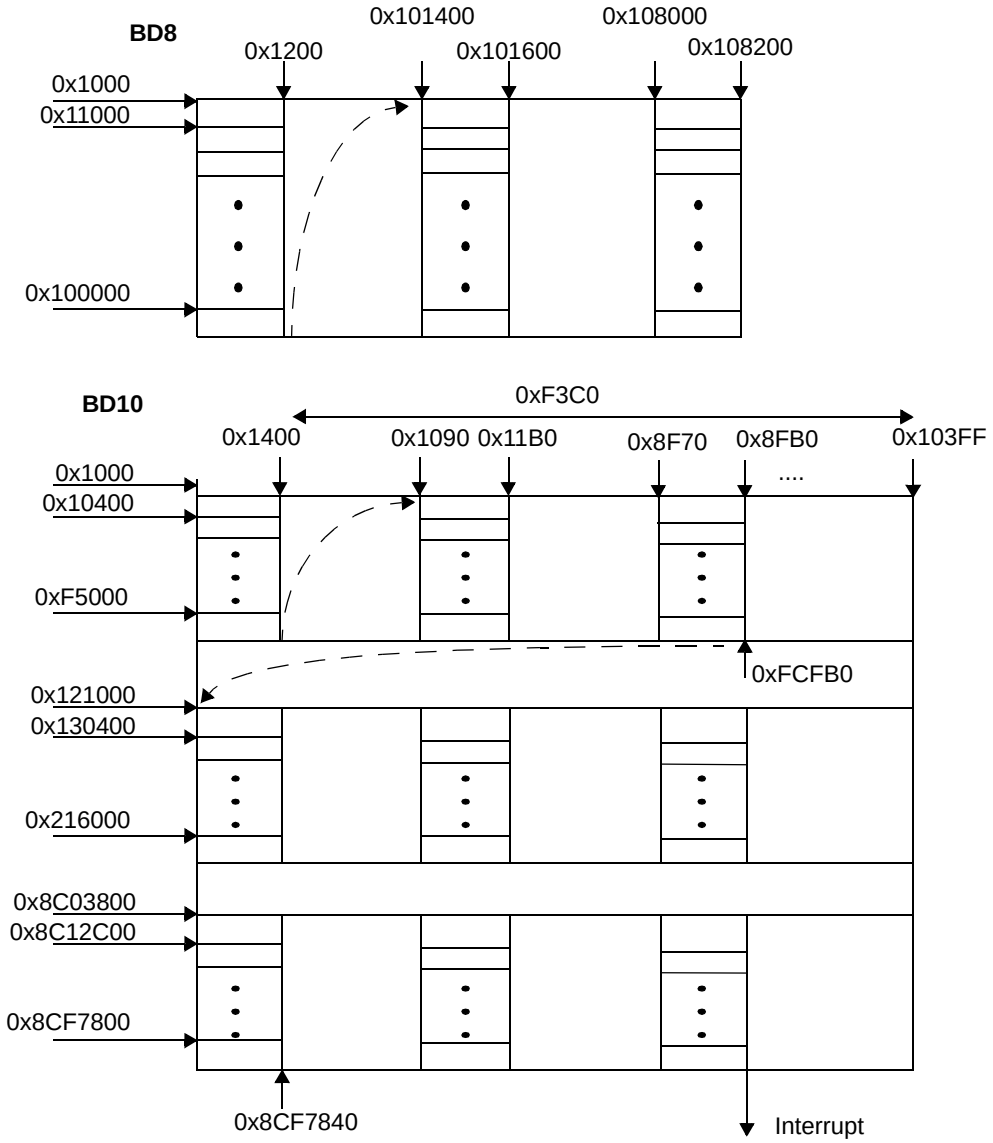


Figure 14-10. Multi-Dimensional Chained Buffer

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.

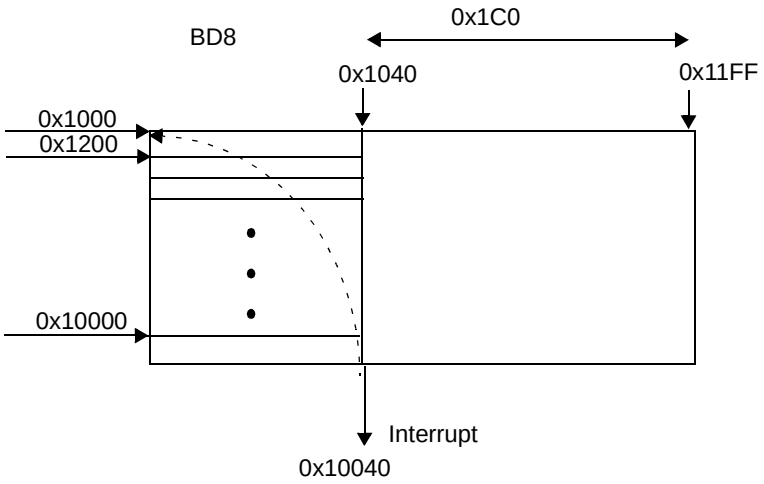


Figure 14-11. Two-Dimensional Cyclic Buffer

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.

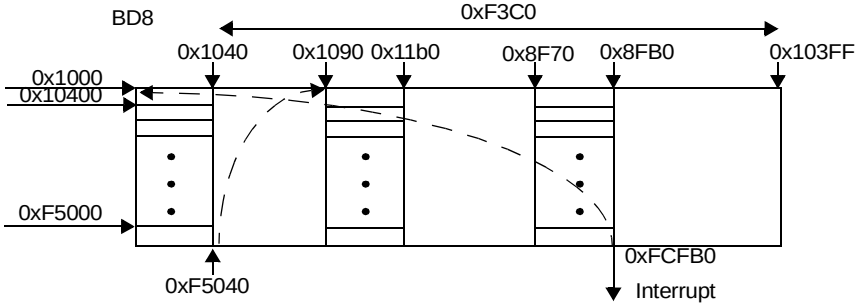


Figure 14-12. Three-Dimensional Cyclic Buffer

Table 14-11 shows the configuration of a cyclic buffer designated as channel BD8. A 0x40000 (0x100 × 10 × 40) three-dimensional block is read from address 0x1000. The basic buffer is 0x40 byte. The offset between each 0x40 byte transaction is 0xF3C0 (0x10400 – 0x1040). The 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 14-27).

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 14-30 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 14-34). The EDF logic compares each counter value with the threshold value. If a counter value equals the threshold value, EDF logic sets the corresponding sticky bit in the pending register.

14.3.2.2 Counter Control

The EDF field in the source BD_ATTR of the channel defines the EDF logic behavior when source BD_SIZE reached zero.

14.3.2.3 Clock Source to the Counters

All the counters share the same clock source. There are four clock sources: 3 external (to the DMA) clock sources and the DMA clock divided by 16.

14.4 Interrupts

The DMA controller uses two types of interrupts: maskable and nonmaskable interrupts.

14.4.1 Maskable Interrupts

The DMA controller can issue one maskable interrupt per channel at the end of a buffer or end of a transfer. For multi-dimension buffers, the interrupt may be issued on any of the dimensions. For each maskable interrupt request, a bit in the DMA Status Register (DMASTR) indicates the interrupt source. The interrupt mask is configured via bits in DMA Mask Register (DMAMR). Maskable interrupts are output as 16 individual interrupts, one for each unidirectional destination channel.

Most of the maskable interrupts are issued to request service or indicate that a transfer is available. The exception is EDF threshold violation error interrupt. This interrupt can be masked for each counter.

14.4.2 Nonmaskable Interrupts

Except for the DMA counters threshold violation, nonmaskable interrupts are error interrupts and are all output by one level-error interrupt. The DMA controller issues unmasked interrupts for one of the following sources:

- EDF violation (the only maskable source of a DMA error; it can be masked per counter)
- Port 0 transfer error
- Port 1 transfer error
- Any parity error
- Buffer size of zero

Note: In some cases in which a PRAM parity error occurs, the BD size zero error may also be set.

For each error source, a bit in the DMA Error Register (DMAERR) indicates the error source. DMAERR also samples the first channel that caused the first bus error and the first channel that caused the first parity error.

14.5 Profiling

The DMA supports system-level profiling via the Channel Profiled (CHAPRO) and Destination Channel Profiled (DEST) bits of the DMA Local Profiling Configuration Register (DMALPCR) (see **Table 14-26,** *DMALPCR Field Descriptions*, on page 14-39). These bits provide the following indications:

- DMA channel active.
- Arbitration winner.
- End of buffer for a DMA channel.
- Bus request.
- Consecutive grant.

14.6 DMA 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 14-24
  - DMA Controller Channel Configuration Registers 0–15 (DMACHCR[0–15]), page 14-25
  - DMA Controller Global Configuration Register (DMAGCR), page 14-27
  - DMA Channel Enable Register (DMACHER), page 14-28
  - DMA Channel Disable Register (DMACHDR), page 14-28
  - DMA Channel Freeze Register (DMACHFR),
  - DMA Channel Defrost Register (DMACHDFR),
  - DMA EDF Time-to-Dead Line Registers 0–15 (DMAEDFDL[0–15]), page 14-30
  - DMA EDF Control Register (DMAEDFCTRL), page 14-31
  - DMA EDF Mask Register (DMAEDFMR), page 14-32
  - DMA EDF Mask Update Register (DMAEDFMUR), page 14-32
  - DMA EDF Status Register (DMAEDFSTR), page 14-34
  - DMA Mask Register (DMAMR), page 14-34
  - DMA Mask Update Register (DMAMUR), page 14-35
  - DMA Status Register (DMASTR), page 14-36
  - DMA Error Register (DMAERR), page 14-37
  - DMA Debug Event Status Register (DMADESR), page 14-39
  - DMA Local Profiling Configuration Register (DMALPCR), page 14-39
  - DMA Round-Robin Priority Group Update Register (DMARRPGUR), page 14-40
  - DMA Channel Active Status Register (DMACHASTR), page 14-41
  - DMA Channel Freeze Status Register (DMACHFSTR), page 14-41
- DMA Channel Buffer Descriptors
  - Buffer Attributes (BD_ATTR), page 14-44
  - Multi-Dimensional Buffer Attributes (BD_MD_ATTR), page 14-47

Note: The DMA controller registers use a base address of: 0xFFF10000.

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

DMABDBRx each holds the user-programmable address of the BD tables for DMA channel x. There are two pointers for each channel: SRCBDPT and DESBDPT in the DMA Controller Channel Configuration Registers (DMACHCRx). The SRCBDPT field points to the location of the source BD in the source table. DESBDPT points to the location of the destination BD in the destination 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 14-42.

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 14-28 and the DMA Channel Disable Register on page 14-28. **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 14-42. **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 14-40) **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	

14.6.2.1 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.

14.6.3 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 reset, 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.

14.6.4 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.

14.6.5 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.

14.6.6 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

14.6.7 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.	

Table 14-18. DMAEDFTDL[0–15] Field Descriptions (Continued)

Bits	Reset	Write By	Description	Settings
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.	

14.6.8 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.	

14.6.9 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.

14.6.10 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.
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.

14.6.11 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.

14.6.12 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 a bit in the DMASTR. For the Sx bit, there is no meaning, because the interrupts are generated for end-of-channel or destination BDs only. The Sx bits should be cleared for future compatibility. If set, each Dx bit enables the generation of interrupt request signal on the corresponding interrupt line. DMAMR is cleared at reset and you can enable a channel interrupt request by setting the appropriate Dx bit.

14.6.13 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	EN2
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	Always set (1) for channel interrupt for destination BD or end of channel.
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	Always set (1) for channel interrupt for destination BD or end of channel.
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	Always set (1) for channel interrupt for destination BD or end of channel.
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	Always set (1) for channel interrupt for destination BD or end of channel.
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	

14.6.14 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/W															
Reset	0	0	0	0	0	0	0	0	0	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

DMASTR is a status register that can be read and written. Writing a 0 has no effect. Writing a 1 clears the corresponding bit. 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.

The Sx bits are set for the corresponding channel source BD or end of channel. The Dx bits are set for the corresponding channel destination BD or end of channel. Although the Sx bits can be used for polling, the Dx bits can also be used for interrupt generation. Note that both the Dx and Sx bits are set whenever the channel ends, regardless of the SST value in the BD.

Note: You must clear the Dx and Sx bits before enabling the channel.

14.6.15 DMA Error Register (DMAERR)

DMAERR DMA Error Register Offset 0x370

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	BDSZ	PACH						PADEST	—	PBCH						PBDEST
Type	R	R/W							R	R/W						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	PAE	PBE	—	THV	PRTYP	PRTYF	PRTYB	PRTY	—	PRTYCH						PRTYD
Type	R/W		R													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMAERR holds the source for the error interrupt. The error interrupt output is unmasked in the DMA controller. A bit is cleared by writing a value of one to it. Writing zero does not affect a bit value. Several bits can be cleared at one time. If a port error occurs, all channels assigned to the port enter a freeze state. You must reprogram the channel that caused the error and reactivate it. You may decide to defrost other port assigned channels and continue normally. For a BD size error or parity error, only the channel that caused the error is frozen.

Table 14-24. DMAERR Description

Bits	Reset	Description	Settings
BDSZ 31	0	**Buffer Descriptor Size** Indicates whether the buffer descriptor size is programmed to zero. See also **Table 14-28** and **Table 14-30**.	0 No BD_SIZE or MD_BD_SIZE of 0 detected. 1 BD_SIZE or MD_BD_SIZE of 0 detected.
PACH 30–25	0	**First Port 0 Channel to Cause Bus Error** Indicates which channel caused the first error on bus interface port A.	000000 - 001111: channel number 01xxxx: Reserved 10xxxx: Reserved
PADEST 24	0	**Error of Port 0 Destination Channel** Indicates whether the last error on port 0 was caused by channel source or destination.	0: Source transaction error. 1: Destination transaction error.
— 23	0	Reserved. Write to zero for future compatibility.	
PBCH 22–17	0	**First Port 1 Channel to Cause Bus Error** Indicates which channel caused the first error on bus interface port B.	000000–001111: channel number. 01xxxx Reserved 10xxxx: Reserved

Table 14-24. DMAERR Description (Continued)

Bits	Reset	Description	Settings
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.	
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.

14.6.16 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.

14.6.17 DMA Local Profiling Configuration Register (DMALPCR)

DMALPCR DMA Local Profiling Configuration Register Offset 0x378

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

DMALPCR selects the channel for system profiling.

Table 14-26. DMALPCR Field Descriptions

Bits	Reset	Description	Settings
— 31–7	0	Reserved. Write to zero for future compatibility.	

Table 14-26. DMALPCR Field Descriptions (Continued)

Bits	Reset	Description	Settings
CHAPRO 6–1	0	**Channel Profiled** Selects the channel to be profiled. **Write by:** User	000000–001111: Channel number. 01xxxx Reserved. 10xxxx Reserved.
DEST 0	0	**Destination Channel Profiled** Specifies whether source or destination requests are profiled. **Write by:** User	0 Channel source requests are profiled. 1 Channel destination requests are profiled.

14.6.18 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 14-27).

Table 14-27. DMARRPGUR Field Descriptions

Bits	Description	Settings
— 31–10	Reserved. Write to zero for future compatibility.	
CH 9–4	**Channel Number** The channel number to which DMACHCR[RRPG] should be changed.	000000–001111: Channel number. 01xxxx Reserved. 1xxxxx Reserved.
NRRPG 3–1	**New Channel Round-Robin Priority Group** The new value of RRPG to be written to the corresponding CHCR of the channel.	000 Highest priority. ... 011 Lowest priority. 1xx Reserved.
EN 0	**Enable RRPG Update** Enables the RRPG update. Then the DMA controller clears this bit	

14.6.19 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

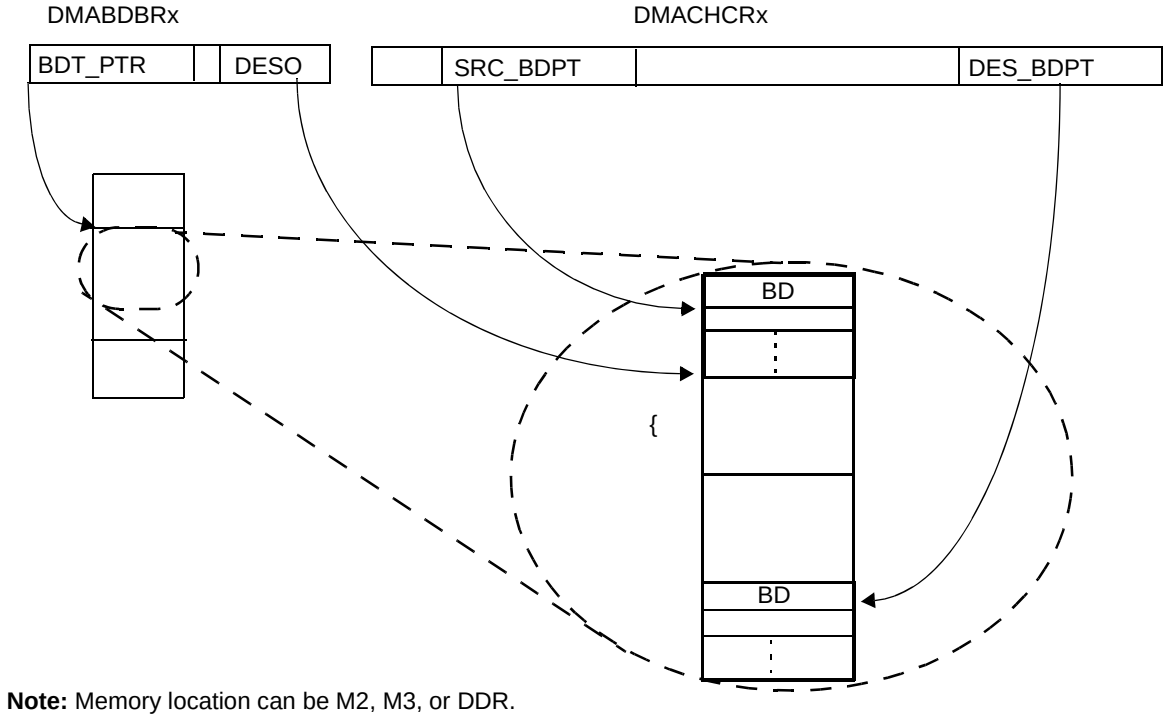
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	D15	S15	D14	S14	D13	S13	D12	S12	D11	S11	D10	S10	D9	S9	D8	S8
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	D7	S7	D6	S6	D5	S5	D4	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.

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 + CHCR[DMADESBDPT] × 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.

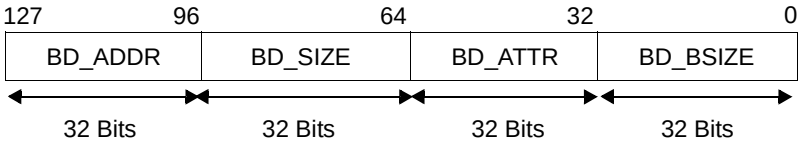


Figure 14-14. DMA Channel BD One-Dimensional Line

Figure 14-15 shows the structure of a multi-dimensional BD, which is a 256-bit entry.

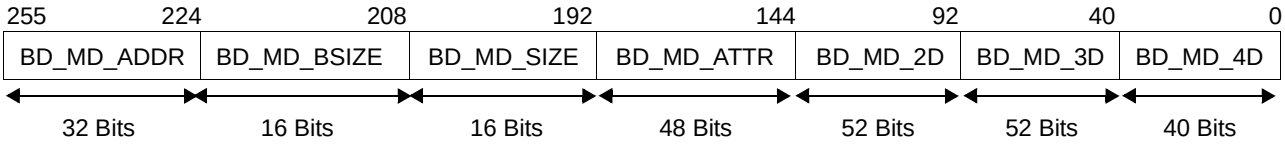


Figure 14-15. DMA Channel Multi-Dimensional Buffer Descriptor

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.

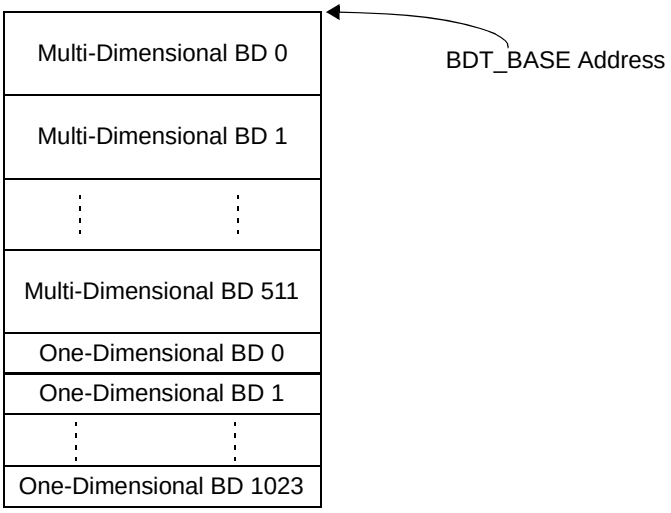


Figure 14-16. Example BD Table with a Mixed Dimensional Structure

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 14-25). **Table 14-28** lists the channel parameters for a one-dimensional BD.

Table 14-28. One-Dimensional BD Field Descriptions

Bits	Description
BD_ADDR 127–96	**Current Buffer Address** Holds the buffer address pointer. This value increments on every transaction the DMA controller issues for this buffer. If the buffer is cyclic, the original address value is restored when the BD_SIZE value reaches zero. For details, refer to **Section 14.2.2**, *One-Dimensional Cyclic Buffer,* on page 14-4.
BD_SIZE 95–64	**Size of Transfer Left for Current Buffer** Contains the remaining size of the buffer. This value decrements by the transfer size each time the DMA controller issues a transaction until it reaches zero. When BD_SIZE reaches zero, the value is restored to the value of BD_BSIZE. **Note:** The BD_SIZE must not be programmed as zero. A value of zero sets DMAERR[BDSZ] and freezes the channel. To reactivate the channel, you must disable the channel, reprogram the BDs, and reactivate the channel.
BD_ATTR 63–32	**Buffer Attributes** This 32-bit parameter describes the attributes of the channel handling this buffer. The fields of the BD_ATTR parameter are described in **Table 14-29**.
BD_BSIZE 31–0	**Buffer Base Size** Holds the base size of the buffer.

14.6.21.1 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 14-29. BD_ATTR Field Descriptions

Bits	Description	Settings
SST 31	**Set Status** Indicates whether to set the associated status bit in DMASTR when size reaches zero and the last data transaction ends. Setting this bit in the destination buffer issues a masked interrupt request.	0 Do not set status 1 Set status.
CYC 30	**Cyclic Address** Indicates the behavior of BD_ADDR when BD_SIZE reaches zero. For details on cyclic buffers, see **Section 14.2**, *Buffer Types,* on page 14-2.	0 Sequential address: BD_ADDR increments. 1 Cyclic address: BD_ADDR is restored to the original value for a one-dimensional buffer.
CONT 29	**Continuous Buffer Mode** Indicates whether buffer is to close when BD_SIZE reaches zero.	0 Buffer closes. 1 Buffer continues operating.

Table 14-29. BD_ATTR Field Descriptions (Continued)

Bits	Description	Settings
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.	
CNT 15–14	**Channel Counter** This field is valid only when the arbitration is time-based. It characterizes the time-based arbitration mechanism for continuous buffers when the buffer size reaches zero and applies only when switching buffers.	00 Continuous: Channel and counter continue working normally. 01 Reserved. 10 Reserved. 11 Resets the channel counter.
PP 13–12	**Port priority** Define priority for this buffer. The bus Interface will set priority for this buffer. Note that the user must map the priority in system level.	00 Priority 0 (lowest). 11 Priority 3 (highest).
TSZ 11–8	**Transfer size** Indicates the maximum transfer size that the DMA will issue when request is detected.	0000 512 bytes 0001 1 byte. 0010 2 bytes. 0011 4 bytes. 0100 8 bytes. 0101 16 bytes. 0110 32 bytes. 0111 64 bytes. 1000 128 bytes. 1001 256 bytes. 1010 512 bytes. 1011 1024 bytes. 11xx Reserved.
— 7	Reserved. Write to zero for future compatibility.	
FRZ 6	**Freeze channel** When size reaches zero, the channel can be frozen. The already serviced requests continue normally. No further requests are issued for the associated channel until the host defrosts it.	0 Normal operation 1 Freeze channel.

Table 14-29. BD_ATTR Field Descriptions (Continued)

Bits	Description	Settings
MR 5	**Mask Requests Until Data Reached Destination** Indicates the behavior of the logic when BD_SIZE reaches zero. Typically, in continuous buffers, the channel should not be masked. However, there is an automatic mask when continuous buffers switch between ports. The DMA controller unmasks the requests when the last data reaches the destination.	0 Normal operation. 1 Mask requests until data reached destination.
— 4–3	Reserved. Write to zero for future compatibility.	
BTSZ 2–0	**Basic Transfer Size** The basic transfer size issued for the request. If BTSZ is greater than TSZ, the DMA controller uses TSZ for the transfer size.	000 64 bytes 001 1 byte 010 2 bytes 011 4 bytes 100 8 bytes 101 16 bytes 110 32 bytes 111 64 bytes

Table 14-30 shows the DMA channel parameters for a multi-dimensional buffer.

Table 14-30. Multi-Dimensional BD Field Descriptions

Bits	Description
BD_MD_ADDR 255–224	**Current Buffer Address** Holds the buffer address pointer. This value increments on every transaction the DMA controller issues for this buffer. When BD_MD_SIZE reached zero and the next dimension is active, the next dimension offset is added to the BD_MD_ADDR. For details, see **Section 14.2**, *Buffer Types,* on page 14-2.
BD_MD_BSIZE 223–208	**First Dimension Buffer Base Size** Contains the base size for the buffer first dimension.
BD_MD_SIZE 207–192	**First Dimension Buffer Size** Holds the remaining size for the buffer first dimension. This value is incremented by the transfer size each time the DMA controller issues a transaction, until it reaches zero. When BD_MD_SIZE reaches zero, its value is restored to the value of BD_MD_BSIZE. **Note:** BD_MD_SIZE must not be programmed to zero. Programming the value to zero sets DMAERR[BDSZ] and freezes the channel. To reactivate the channel, you must disable the channel, wait for the active status bit to go to clear, reprogram the BDs, and reactivate the channel.
BD_MD_ATTR 191–144	**Multi-Dimensional Buffer Attributes** This 48-bit parameter describes the multi-dimensional attributes of the channel handling this buffer. The BD_MD_ATTR parameters are described in **Table 14-31**.
BD_MD_2D 143–92	**Two-Dimensional Buffer Offset, Bcount, and Count** This 52-bit parameter holds the two-dimensional parameters of the channel handling this buffer. It holds the base count value, current count, and address offset. The Multi Dimension fields are described in **Table 14-32,** *BD_MD_2D Field Descriptions,* on page 14-50.
BD_MD_3D 91–40	**Three-Dimensional Buffer Offset, Bcount, and Count** This 52-bit parameter holds the three-dimension parameters of the channel handling this buffer. It holds the base count value, current count, and address offset. The Multi Dimension fields are described in **Table 14-33,** *BD_MD_3D Field Descriptions,* on page 14-50.

Table 14-30. Multi-Dimensional BD Field Descriptions (Continued)

Bits	Description
BD_MD_4D 39–0	**Four-Dimensional Buffer Offset and Count** This 40-bit parameter holds the four-dimensional parameters of the channel handling this buffer. It holds the base count value and current count. The multi-dimensional fields are described in **Table 14-34,** *BD_MD_4D Field Descriptions,* on page 14-50.

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 14-31. BD_MD_ATTR Field Descriptions

Bits	Description	Settings
SST 47	**Set Status** Indicates whether to set the associated status bit in DMASTR when size reaches zero and the last data transaction ends. Setting this bit in the destination buffer issues a masked interrupt request. See also the SSTD bit.	0 Do not set status. 1 Set status when the size of the dimension selected by SSTD reaches zero.
CYC 46	**Cyclic Address** Indicates the behavior of BD_MD_ADDR when BD dimension count reaches zero. For details on cyclic buffers, see **Section 14.2**, *Buffer Types,* on page 14-2. **Notes:** **1.** This field must not be set for four dimensional buffers. **2.** This field is not valid for BD_MD_ATTR[CONTD]<BD_MD+ATTTR[BD].	0 Sequential address: BD_MD_ADDR is incremented. 1 Cyclic address: the next dimension offset is added to BD_MD_ADDR.
CONT 45	**Continuous Buffer Mode** Specifies whether the buffer closes when CONTD dimension count reaches zero.	0 Buffer closes. 1 Buffer continues operating.
NPRT 44	**Next Port** When size reaches zero and CONT is set and the CONTD dimension count reaches zero, DMACHCR[SPRT] (for source buffers) or DMACHCR[DPRT] (for destination buffers) is updated according to the NPRT field.	0 Next buffer port is MBus port 0. 1 Next buffer port is MBus port 1.

Table 14-31. BD_MD_ATTR Field Descriptions (Continued)

Bits	Description	Settings
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. 11xx Reserved.
— 23	Reserved. Write to zero for future compatibility.	
FRZ 22	**Freeze Channel** When size reached zero the channel can be freeze. The already serviced requests continue normally. No further requests are issued for the associated channel until the host defrost it. See also the FRZD field.	0 Normal operation. 1 Freeze channel when size reaches zero on the dimension selected by FRZD.
MR 21	**Mask Requests Until Data Reached Destination** Indicates the behavior of the logic when BD_MD_SIZE reaches zero. In continuous buffers, the channel is usually not masked. There is an automatic mask when ports are switched in a continuous buffer. The DMA controller unmasks the requests when last data reaches the destination. See also the MRD field.	0 Normal operation. 1 Mask requests until data reached destination on the dimension selected by MRD.

Table 14-31. BD_MD_ATTR Field Descriptions (Continued)

Bits	Description	Settings
— 20–19	Reserved. Write to zero for future compatibility.	
BTSZ 18–16	**Basic Transfer Size** The basic transfer size issued by the request. If BTSZ is greater than TSZ, the DMA controller uses the TSZ value.	000 64 bytes. 001 1 byte. 010 2 bytes. 011 4 bytes. 100 8 bytes. 101 16 bytes. 110 32 bytes. 111 64 bytes.
— 15–11	Reserved. Write to zero for future compatibility.	
LAST 10	**Last Buffer In Chain** Set this bit only to avoid an endless task condition when CONT is set and CONTD is smaller than BD.	0 Not the last buffer in the chain. 1 Last buffer in the chain.
BD 9–8	**Buffer Dimension** Indicates the dimension of the buffer.	00 1 dimension. 01 2 dimensions. 10 3 dimensions. 11 4 dimensions.
SSTD 7–6	**Set Status Dimension** When BD_MD_SIZE reaches zero and BD_MD_ATTR[SST] = 1, the status bit is set for the channel in DMASTR. For details, see page 14-36. The SSTD field defines the dimension on which the status bit is set.	00 BD_MD_SIZE reached zero, 1D. 01 M2D_COUNT reached zero, 2D. 10 M3D_COUNT reached zero, 3D. 11 M4D_COUNT reached zero, 4D.
FRZD 5–4	**Freeze Dimension** When the selected dimension is processed, the internal logic masks all channel requests and freezes the channel. The host must defrost the channel for further service.This field is valid only if FRZ is set in the BD_MD_ATTR.	00 Mask and freeze channel when first dimension ends. 01 Mask and freeze channel when second dimension ends. 10 Mask and freeze channel when third dimension ends. 11 Mask and freeze channel when fourth dimension ends.
CONTD 3–2	**Continuous Buffer Mode Dimension Select** Indicates the dimension after which the channel switches to the next multi-dimensional BD. This field is valid only if CONT is set in the BD_MD_ATTR. **Notes:** **1.** The value must not be greater than BD. **2.** When NBD is equal to the current BD and CONT is set, CONTD must equal BD. **3.** If CONT is set and CONTD < BD, the BD area is updated by the DMA controller. Do not access this area until the DMA task is completed. **4.** If CONT is set, CONTD < BD, and NPRT is different from the current port, MR must be set an MRD must be equal to CONTD.	00 Switch to next BD when BD_MD_SIZE reaches zero, 1D 01 Switch to next BD when M2D_COUNT reaches zero, 2D. 10 Switch to next BD when M3D_COUNT reaches zero, 3D. 11 Switch to next BD when M4D_COUNT reaches zero, 4D.

Table 14-31. BD_MD_ATTR Field Descriptions (Continued)

Bits	Description	Settings
MRD 1–0	**Mask Requests Dimension** Indicates the dimension after which the channel masks requests until the data reaches its destination. This field is valid only if MR is set in the BD_MD_ATTR.	00 Mask requests when BD_MD_SIZE reaches zero, 1D. 01 Mask requests when M2D_COUNT reaches zero, 2D. 10 Mask requests when M3D_COUNT reaches zero, 3D. 11 Mask requests when M4D_COUNT reaches zero, 4D.

Table 14-32. BD_MD_2D Field Descriptions

Bits	Description
M2D_COUNT 51–40	**Second Dimension Current Count** Decrements each time the BD_MD_SIZE reaches zero. The field is reloaded with the M2D_BCOUNT each time it reaches zero. **Note:** If the buffer is two dimensional or more, this field cannot be 0.
M2D_BCOUNT 39–28	**Second Dimension Base Count** Holds the second dimension base count. **Note:** If the buffer is more than two dimensional, this field cannot be 0.
M2D_OFFSET 27–0	**Second Dimension Offset** Written in two's complement. The offset is added to the BD_MD_ADDR each time BD_MD_SIZE reaches zero.

Table 14-33. BD_MD_3D Field Descriptions

Bits	Description
M3D_COUNT 51–40	**Third Dimension Current Count** Decrements each time the BD_MD_SIZE and M2D_COUNT reach zero. The field is reloaded with the M3D_BCOUNT each time it reaches zero. **Note:** If the buffer is three dimensional or more, this field cannot be 0.
M3D_BCOUNT 39–28	**Third Dimension Base Count** Holds the third dimension base count. **Note:** If the buffer is more than three dimensional, this field cannot be 0.
M3D_OFFSET 27–0	**Third Dimension Offset** Written in two's complement. The offset is added to the BD_MD_ADDR each time BD_MD_SIZE and M2D_COUNT reach zero.

Table 14-34. BD_MD_4D Field Descriptions

Bits	Description
M4D_COUNT 39–28	**Fourth Dimension Current Count** Decrements each time BD_MD_SIZE, M2D_COUNT, and M3D_COUNT reach zero. **Note:** If the buffer is four dimensional, then this field cannot be 0.
M4D_OFFSET 27–0	**Fourth Dimension Offset** Written in two's complement. The offset is added to the BD_MD_ADDR each time BD_MD_SIZE, M2D_COUNT, and M3D_COUNT reach zero.

PCI 15

The PCI interface is designed to comply with the *PCI Local Bus Specification*, Rev. 2.2. This chapter describes the PCI controller and provides a basic description of the PCI bus operations. The specific emphasis is directed at how this device implements the PCI specification. System designers incorporating PCI devices should refer to the respective specification for a thorough description of the PCI buses.

Note: Much of the available PCI literature refers to a 16-bit quantity as a WORD and a 32-bit quantity as a DWORD. Because this is inconsistent with the terminology in this manual, the terms *word* and *double word* are not used. Instead, bus width is defined by the exact number of bits or bytes indicated.

The PCI controller acts as a bridge between the PCI interface and the core processors and internal memory system with an I/O sequencer to buffer the data. This interface acts as both initiator and target device. The PCI controller uses a 32- bit multiplexed, address/data bus that can run at frequencies up to 66 MHz. The interface provides address and data parity with error checking and reporting. The interface provides for three physical address spaces—64-bit address memory, 32-bit address I/O, and PCI configuration space.

The PCI controller functions as a peripheral device on the PCI bus (referred to as agent mode). After a power-on reset, the PCI configuration space is locked to allow changing the size of the inbound window. This may be done by the boot program (see **Chapter 6**, *Boot Program*) or by the user depending on the setting of the TCWHR[PCI] bit (see **Chapter 5**, *Reset*). The PCI controller ignores all PCI memory accesses except those to the memory-mapped registers) until inbound address translation is enabled. It can be configured through the PCI controller. An address translation mechanism is provided to map PCI memory windows between the PCI bus and the internal bus.

The VCOP includes:

- 32-bit PCI interface support.
- Agent mode support.
- Supports accesses to all PCI address spaces.
- 64-bit dual-address cycle (DAC) support (as a target only).
- Internal configuration registers accessible from PCI and internal buses.

- Contains an internal cache line (32 byte) to allow PCI-to-memory and memory-to-PCI streaming.
- Memory prefetching of PCI read accesses and support for delayed read transactions.
- Supports posting of processor-to-PCI and PCI-to-memory writes.
- Inbound and Outbound address translation units for address mapping between PCI and internal buses.
- Supports parity.
- PCI 3.3-V compatible.

15.1 Functional Description

The following sections discuss the operation of the PCI controller.

15.1.1 PCI Operation Modes

The PCI controller can be programmed to work as an initiator only device, a target only device, or a both an initiator and target device.

- As the initiator, the PCI controller acts as a bridge between the MSC8144E interconnect and the PCI bus. The VCOP uses $\overline{\text{REQ}}$ to issue requests to an external arbiter, and uses $\overline{\text{GNT}}$ to receive grants from the external arbiter. PCI initiator mode is enabled by setting PCICCR[BMST].
- As the target, the PCI controller acts as a bridge between the PCI bus and the MSC8144E interconnect. The PCI can be programmed to transfer transactions initiated by the PCI initiator to any of the following: M2 memory, M3 memory, DDR memory, or the configuration registers (CCSR). PCI target mode is enabled by setting PCICCR[MEM].

The three operation modes are described in **Table 15-1**

Table 15-1. PCI Operation Modes

Mode	Settings	Description
Initiator only	PCICCR[BMST] = 1 PCICR[MEM] = 0 C2GPR[PMDRD] = 0 or 1 C2GPR[PPE] = 0 or 1	The four DSP cores, the DMA controller, the QUICC Engine subsystem, the serial RapidIO controller, and the TDM modules can initiate transactions toward the PCI. In this mode, the delayed read transaction can be disabled (by setting C2GPR[PMDRD]) to permit cascading of read transactions to a single stream. In this mode, the internal controller pipeline can be enabled (by setting C2GPR[PPE]) to improve the PCI controller performance.

Table 15-1. PCI Operation Modes (Continued)

Mode	Settings	Description
Target only	PCICCR[BMST] = 0 PCICR[MEM] = 1 C2GPR[PMDRD] = 0 C2GPR[PPE] = 0	No master is allowed to initiate transactions toward the PCI, the PCI master delayed read must be enabled, and the controller internal pipeline must be disabled.
Initiator and target	PCICCR[BMST] = 1 PCICR[MEM] = 1 C2GPR[PMDRD] = 0 C2GPR[PPE] = 0	• The four DSP cores, the DMA controller, or the QUICC Engine subsystem can initiate transactions toward PCI. • The serial RapidIO controller and TDM modules must not initiate PCI transactions • The PCI master delayed read must be enabled. • The internal PCI controller pipeline must be disabled.

15.1.2 Bus Commands

PCI bus commands indicate the type of transaction occurring on the bus. These commands are encoded on PCI_C/$\overline{\text{BE}}$[3–0] during the address phase of the transaction. PCI bus commands are described in **Table 15-2**.

Table 15-2. PCI Command Definitions

Command Type	PCI_C/$\overline{\text{BE}}$[3–0]	Definition	Supported as:	
			Initiator	Target
Interrupt acknowledge	0b0000	A read implicitly addressed to the system interrupt controller. The size of the vector to be returned is indicated on the byte enables after the address phrase.	Yes	No
Special cycle	0b0001	Provides a simple message broadcast mechanism.	Yes	No
I/O read	0b0010	Accesses agents mapped in I/O address space.	Yes	No
I/O write	0b0011	Accesses agents mapped in I/O address space.	Yes	No
—	0b010x	Reserved. No response occurs.	—	—
Memory read	0b0110	Accesses agents mapped in memory address space. A read from prefetchable space, when seen as a target, fetches a cache line of data (32 bytes) from the starting address, even though all 32 bytes may not actually be sent to the initiator.	Yes	Yes
Memory write	0b0111	Accesses agents mapped in memory address space.	Yes	Yes
—	0b100x	Reserved. No response occurs.	—	—
Configuration read	0b1010	Accesses the PCI configuration space.	No	Yes
Configuration write	0b1011	Accesses the PCI configuration space.	No	Yes
Memory read multiple	0b1100	Causes a prefetch of the next cache line.	Yes	Yes
Dual address cycle	0b1101	Transfers an 8-byte address to devices.	No	Yes
Memory read line	0b1110	Indicates that the initiator intends to transfer an entire cache line of data.	Yes	Yes
Memory write and invalidate	0b1111	Indicates that the initiator will transfer an entire cache line of data, and if PCI has any cacheable memory, this line needs to be invalidated.	No	Yes

Note: As an initiator, 32-byte burst reads are translated either to a memory read line or memory read multiple depending on the configuration of C2GPR[PRCS] (see **Chapter 4**, *Chip-Level Arbitration and Switching System (CLASS)*).

15.1.3 PCI Protocol Fundamentals

The bus transfer mechanism on the PCI bus is called a burst. A burst is comprised of an address phase and one or more data phases. All signals are sampled on the rising edge of the PCI clock. Each signal has a setup and hold window with respect to the rising clock edge, in which transitions are not allowed. Outside this aperture, signal values or transitions have no significance. PCI data transfers are controlled by the following three fundamental signals:

- $\overline{\text{PCI_FRAME}}$ is driven by an initiator to indicate the beginning and end of a transaction.
- $\overline{\text{PCI_IRDY}}$ (initiator ready) is driven by an initiator, allowing it to force wait cycles.
- $\overline{\text{PCI_TRDY}}$ (target ready) is driven by a target, allowing it to force wait cycles.

The bus is idle when both $\overline{\text{PCI_FRAME}}$ and $\overline{\text{PCI_IRDY}}$ are deasserted. The first clock cycle in which $\overline{\text{PCI_FRAME}}$ is asserted indicates the beginning of the address phase. The address and the bus command code are transferred in that cycle. The next cycle ends the address phase and begins the data phase.

During the data phase, data is transferred in each cycle that both $\overline{\text{PCI_IRDY}}$ and $\overline{\text{PCI_TRDY}}$ are asserted. Once the VCOP, as an initiator, has asserted $\overline{\text{PCI_IRDY}}$, it does not change $\overline{\text{PCI_IRDY}}$ or $\overline{\text{PCI_FRAME}}$ until the current data phase completes, regardless of the state of $\overline{\text{PCI_TRDY}}$. Once the VCOP, as a target, has asserted $\overline{\text{PCI_TRDY}}$ or $\overline{\text{PCI_STOP}}$ it does not change $\overline{\text{PCI_DEVSEL}}$, $\overline{\text{PCI_TRDY}}$, or $\overline{\text{PCI_STOP}}$ until the current data phase completes.

When the VCOP (as an initiator) intends to complete only one more data transfer, $\overline{\text{PCI_FRAME}}$ is deasserted and $\overline{\text{PCI_IRDY}}$ is asserted (or kept asserted) indicating the initiator is ready. After the target indicates it is ready ($\overline{\text{PCI_TRDY}}$ asserted) the bus returns to the idle state.

15.1.4 Addressing

The PCI specification defines three physical address spaces—memory, I/O, and configuration. The memory and I/O address spaces are standard for all systems. The configuration address space has been defined specifically to support PCI hardware configuration. Each PCI device decodes the address for each PCI transaction with each agent responsible for its own address decode. The information contained in the two lower address bits (AD1 and AD0) depends on the address space. In the I/O address space, all 32 address/data lines provide the full byte address. AD[1–0] are used for the generation of $\overline{\text{PCI_DEVSEL}}$ and indicate the least significant valid byte involved in the transfer. In the configuration address space, accesses are decoded to a 4-byte address using AD[7–2]. An agent determines if it is the target of the access when a configuration command is decoded, IDSEL is asserted, and AD[1–0] are 0b00; otherwise, the agent ignores the current transaction.

For memory accesses, the address is decoded using AD[31–2]; thereafter, the address is incremented internally by 4 bytes until the end of the burst transfer. Another initiator in a memory access should drive 0b00 on AD[1–0] during the address phase to indicate a linear

incrementing burst order. The VCOP checks AD[1–0] during a memory command access and provides the linear incrementing burst order. On reads, if AD[1–0] is 0b10, which represents a cache line wrap, the VCOP linearly increments the burst order starting at the critical 64-bit address, wraps at the end of the cache line, and disconnects after reading one cache line. If AD[1–0] is 0bx1 (a reserved encoding) and the PCI_C/$\overline{\text{BE}}$[3–0] signals indicate a memory transaction, it executes a target disconnect after the first data phase is completed. Note that AD[1–0] are included in parity calculations.

15.1.5 Device Selection

As a target, the VCOP drives $\overline{\text{PCI_DEVSEL}}$ one clock following the address phase as indicated in the configuration space status register; see **page 15-24** for more information. The VCOP as a target qualifies the address/data lines with $\overline{\text{PCI_FRAME}}$ before asserting $\overline{\text{PCI_DEVSEL}}$. The $\overline{\text{PCI_DEVSEL}}$ signal is asserted at or before the clock edge at which the VCOP enables its $\overline{\text{PCI_TRDY}}$, $\overline{\text{PCI_STOP}}$, or data (for a read). The $\overline{\text{PCI_DEVSEL}}$ signal is not deasserted until $\overline{\text{PCI_FRAME}}$ is deasserted, with $\overline{\text{PCI_IRDY}}$ asserted and either $\overline{\text{PCI_STOP}}$ or $\overline{\text{PCI_TRDY}}$ asserted. The exception to this is a target-abort; see **Section 15.1.8.2** for more information. As an initiator, if the VCOP does not see the assertion of $\overline{\text{PCI_DEVSEL}}$ within 4 clocks of $\overline{\text{PCI_FRAME}}$, it terminates the transaction with an initiator-abort as described in **Section 15.1.8.2**.

15.1.6 Byte Enable Signals

The byte enable signals ($\overline{\text{BE}}$[3–0]) indicate which byte lanes carry valid data. The byte enable signals may enable different bytes for each of the data phases. The byte enable signals are valid on the edge of the clock that starts each data phase and remain valid for the entire data phase.

If the VCOP, as a target, sees no byte enable signals asserted, it completes the current data phase with no permanent change. This implies that on a read transaction, the VCOP expects the data not to be changed, and on a write transaction, the data is not stored.

15.1.7 Bus Driving and Turnaround

The turnaround-cycle is one clock cycle and is required to avoid contention. This cycle occurs at different times for different signals. $\overline{\text{PCI_IRDY}}$, $\overline{\text{PCI_TRDY}}$, and $\overline{\text{PCI_DEVSEL}}$ use the address phase as their turnaround-cycle. $\overline{\text{PCI_FRAME}}$, PCI_C/$\overline{\text{BE}}$[3–0], and AD[31–0] use the idle cycle between transactions as their turnaround-cycle. (An idle cycle in PCI is when both $\overline{\text{PCI_FRAME}}$ and $\overline{\text{PCI_IRDY}}$ are deasserted). Byte lanes not involved in the current data transfer are driven to a stable condition even though the data is not valid.

15.1.8 Bus Transactions

The timing diagrams in this section show the relationship of significant signals involved in bus transactions.

Note the following conventions:

- When a signal is drawn as a solid line, it is actively being driven by the current initiator or target.
- When a signal is drawn as a dashed line, no agent is actively driving it.
- Three-stated signals with slashes between the two rails have indeterminate values.
- The terms edge and clock edge refer to the rising edge of the clock.
- The terms asserted and deasserted refer to the globally visible state of the signal on the clock edge, and not to signal transitions.
- The symbol  represents a turnaround-cycle.

15.1.8.1 Read and Write Transactions

Both read and write transactions begin with an address phase followed by a data phase. The address phase occurs when $\overline{\text{PCI_FRAME}}$ is asserted for the first time, and the AD[31–0] signals contain a byte address and the PCI_C/$\overline{\text{BE}}$[3–0] signals contain a bus command. The data phase consists of the actual data transfer and possible wait cycles; the byte enable signals remain actively driven from the first clock of the data phase through the end of the data transfer.

A read transaction starts when $\overline{\text{PCI_FRAME}}$ is asserted for the first time and the PCI_C/$\overline{\text{BE}}$[3–0] signals indicate a read command. **Figure 15-1** shows an example of a single beat read transaction.

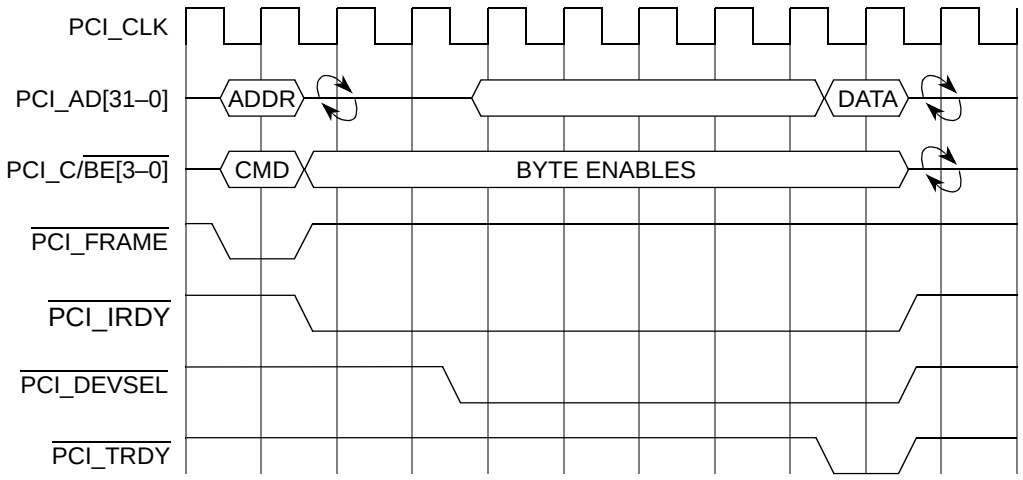


Figure 15-1. Single Beat Read Example

Figure 15-2 shows an example of a burst read transaction.

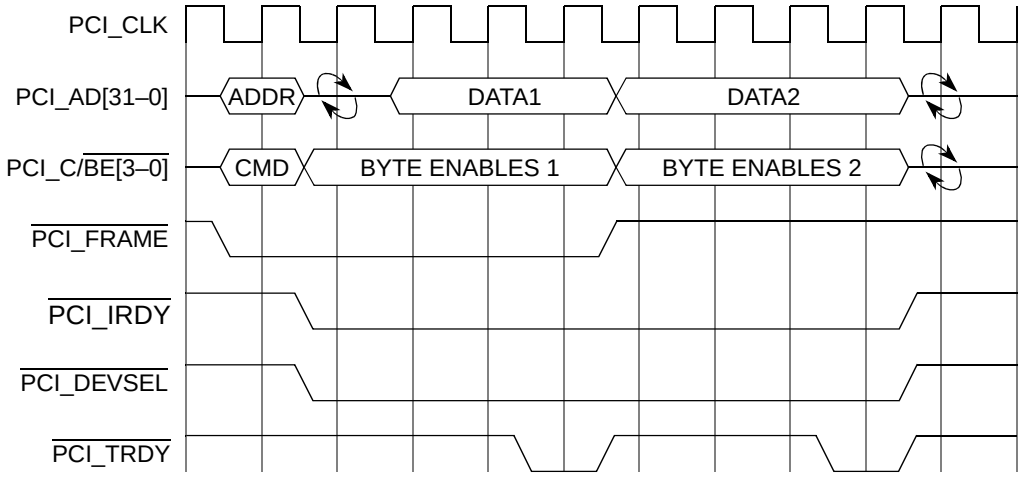


Figure 15-2. Burst Read Example

During the turnaround-cycle following the address phase, the PCI_C/BE[3–0] signals indicate which byte lanes are involved in the data phase. The turnaround-cycle must be enforced by the target with the PCI_TRDY signal if using fast PCI_DEVSEL assertion. The earliest the target can provide valid data is one cycle after the turnaround-cycle. The target must drive the AD[31–0] signals when PCI_DEVSEL is asserted except during the turnaround-cycle.

The data phase completes when data is transferred, which occurs when both PCI_IRDY and PCI_TRDY are asserted on the same clock edge. When either is deasserted a wait cycle is inserted and no data is transferred. To indicate the last data phase PCI_IRDY must be asserted when PCI_FRAME is deasserted.

A write transaction starts when PCI_FRAME is asserted for the first time and the PCI_C/BE[3–0] signals indicate a write command. **Figure 15-3** shows an example of a single beat write transaction.

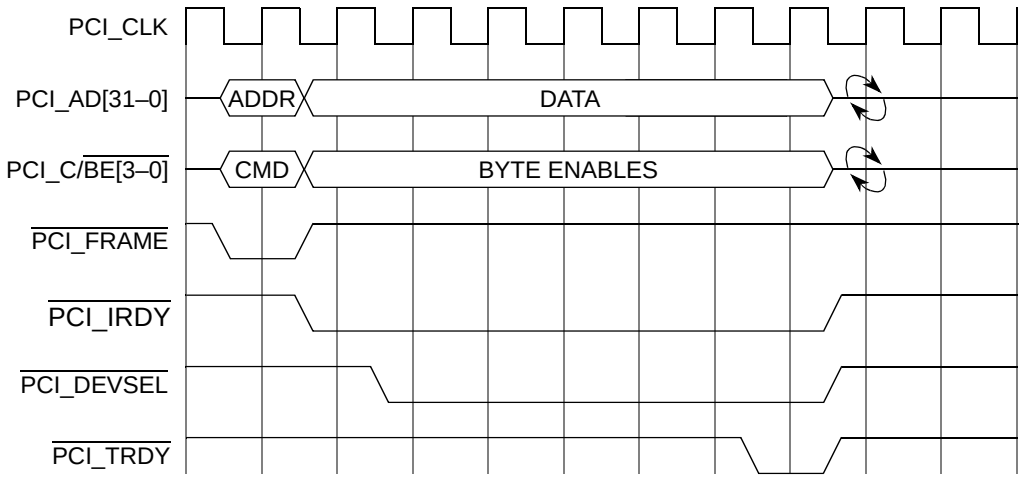


Figure 15-3. Single Beat Write Example

Figure 15-4 shows an example of a burst write transaction.

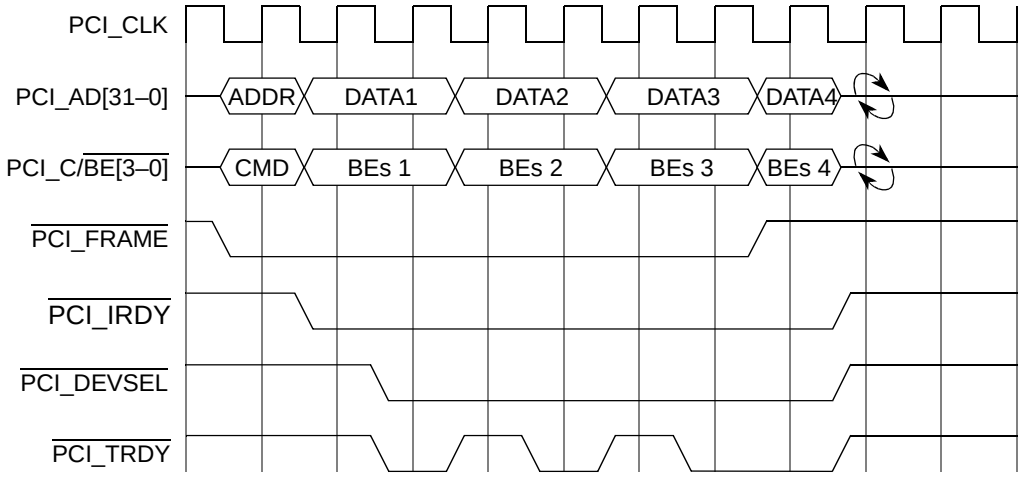


Figure 15-4. Burst Write Example

A write transaction is similar to a read transaction except no turnaround cycle is needed following the address phase because the initiator provides both address and data. Data phases are the same for both read and write transactions.

15.1.8.2 Transaction Termination

The termination of a PCI transaction is orderly and systematic, regardless of the cause of the termination. All transactions end when PCI_FRAME and PCI_IRDY are both deasserted, indicating the idle cycle.

The VCOP as an initiator terminates a transaction when PCI_FRAME is deasserted and PCI_IRDY is asserted. This indicates that the final data phase is in progress. The final data transfer occurs when both PCI_TRDY and PCI_IRDY are asserted. An initiator-abort is an abnormal case of an initiator initiated termination. If the VCOP detects that PCI_DEVSEL has remained deasserted for more than four clocks after the assertion of PCI_FRAME, it deasserts PCI_FRAME and then, on the next clock, deasserts PCI_IRDY. On aborted reads, the VCOP returns 0xFFFF_FFFF. The data is lost on aborted writes.

When the VCOP as a target needs to suspend a transaction, it asserts PCI_STOP. Once asserted, PCI_STOP remains asserted until PCI_FRAME is deasserted. Depending on the circumstances, data may or may not be transferred during the request for termination. If PCI_TRDY and PCI_IRDY are asserted during the assertion of PCI_STOP**,** data is transferred. This type of target-initiated termination is called a disconnect B, shown in **Figure 15-5**. If PCI_TRDY is asserted when PCI_STOP is asserted but PCI_IRDY is not, PCI_TRDY must remain asserted until PCI_IRDY is asserted and the data is transferred. This is called a “disconnect A” target-initiated termination, also shown in **Figure 15-5**. However, if PCI_TRDY is deasserted when PCI_STOP is asserted, no more data is transferred, and the initiator therefore does not have to wait for a final data transfer (see the retry diagram in **Figure 15-5**).

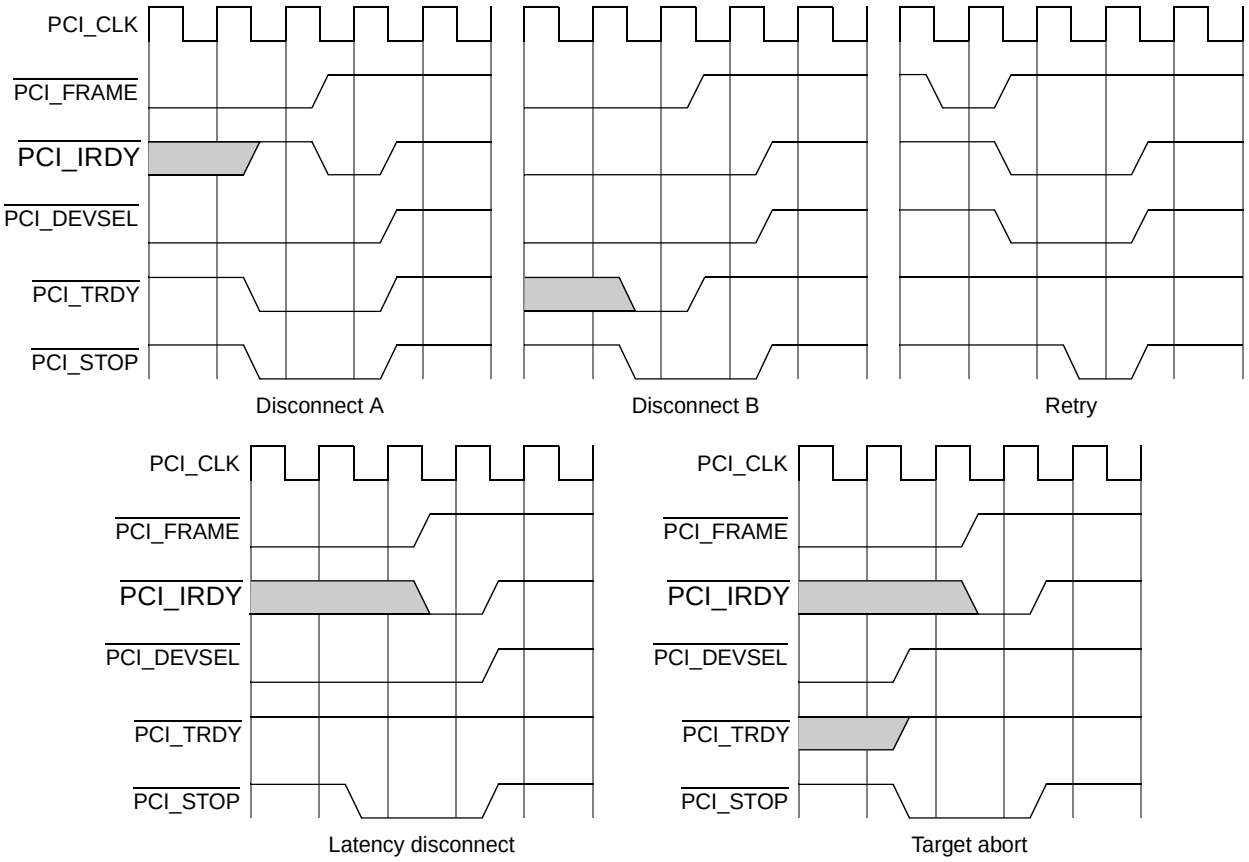


Figure 15-5. Target-Initiated Terminations

Note that when an initiator is terminated by $\overline{\text{PCI_STOP}}$, it must deassert its $\overline{\text{REQ}}$ signal for a minimum of two PCI clocks (of which one clock is needed for the bus to return to the idle state). If the initiator intends to complete the transaction, it should reassert its $\overline{\text{REQ}}$ immediately following the two clocks or potential starvation may occur. If the initiator does not intend to complete the transaction, it can assert $\overline{\text{REQ}}$ whenever it needs to use the PCI bus again.

The VCOP terminates a transaction in the following cases:

- Eight PCI clock cycles have elapsed between data phases. This is a latency disconnect (see **Figure 15-5**).
- AD[1–0] is 0bx1 **(**a reserved burst ordering encoding) during the address phase and one data phase has completed.
- The PCI command is a configuration command and one data phase has completed.
- A streaming transaction crosses a 4K page boundary.
- A streaming transaction runs out of I/O sequencer buffer entries.
- A cache line wrap transaction has completed a cache line transfer.

Another target-initiated termination is the retry termination. Retry refers to termination requested because the target is currently in a state where it is unable to process the transaction. This can

occur because no buffer entries are available in the I/O sequencer, or the sixteen clock latency timer has expired without transfer of the first data. The target latency timer of the VCOP can be optionally disabled (see **page 15-33**).

When the VCOP is in agent mode and the CFG_LOCK — configuration lock bit is set (see **page 15-33** for details) the VCOP will retry all transactions to the PCI Configuration Space or the internal (on-chip) memory-mapped register space. Note that all retried accesses need to be completed. An example of a retry is shown in **Figure 15-5**.

Note that because a target can determine whether or not data is transferred (when both $\overline{\text{PCI_IRDY}}$ and $\overline{\text{PCI_TRDY}}$ are asserted), if it wants to do only one more data transfer and then stop, it may assert $\overline{\text{PCI_TRDY}}$ and $\overline{\text{PCI_STOP}}$ at the same time.

Target-abort refers to the abnormal termination that is used when a fatal error has occurred, or when a target will never be able to respond. Target-abort is indicated when $\overline{\text{PCI_STOP}}$ is asserted and $\overline{\text{PCI_DEVSEL}}$ is deasserted. This indicates that the target requires the transaction to be terminated and does not want the transaction tried again. Note that any transferred data may have been corrupted.

If the VCOP is the intended target of a transaction and an address parity error occurs, or a data parity error occurs on a write transaction to system memory, it continues the transaction on the PCI bus but aborts internally. The VCOP does not target-abort in this case.

If the VCOP is initiating a transaction and the transaction terminates with a target-abort, undefined data will be returned on a read and write data will be lost. An example of a target-abort is shown in **Figure 15-5**.

An initiator may retry any target disconnect accesses, except target-abort, at a later time starting with the address of the next non-transferred data. Retry is actually a special case of disconnect where no data transfer occurs at all and the initiator must start the entire transaction over again.

15.1.8.3 Other Bus Operations

The following sections provide information on additional PCI bus operations.

15.1.8.3.1 Fast Back-to-Back Transactions

In the two types of fast back-to-back transactions, the first type places the burden of avoiding contention on the initiator while the second places the burden on all potential targets. The VCOP as a target supports both types of fast back-to-back transactions but does not support them as an initiator. The VCOP as a target has the fast back-to-back enable bit hardwired to one, that is, enabled.

For the first type (governed by the initiator), the initiator may only run a fast back-to-back transaction to the same target. For the second type, when the VCOP detects a fast-back-to-back

operation and did not drive $\overline{\text{PCI_DEVSEL}}$ in the previous cycle, it delays the assertion of $\overline{\text{PCI_DEVSEL}}$ and $\overline{\text{PCI_TRDY}}$ for one cycle to allow the other target to get off the bus.

15.1.8.3.2 Dual Address Cycles

The VCOP supports dual address cycle (DAC) commands (64-bit addressing on PCI bus) as a target only. DACs are different from single address cycles (SACs) in that the address phase takes two PCI beats instead of one PCI beat to transfer (64-bit vs. 32-bit addressing). Only PCI memory commands can use DAC cycles; I/O, configuration, interrupt acknowledge, and special cycle command cannot use DAC cycles. The VCOP supports single-beat and burst DAC transactions.

15.1.8.3.3 Data Streaming

The VCOP provides data streaming for PCI transactions to and from prefetchable memory. In other words, when the VCOP is a target for a PCI initiated transaction, it supplies or accepts multiple cache lines of data without disconnecting. For PCI transactions to non-prefetchable space, the VCOP disconnects after the first data phase so that no streaming can occur.

For PCI memory reads, streaming is achieved by performing speculative reads from memory in prefetchable space. A block of memory may be marked as prefetchable by setting the PCI configuration registers bit for the inbound address translation (see **page 15-42** for details) in the following cases:

- When reads do not alter the contents of memory (reads have no side effects)
- When reads return all bytes regardless of the byte enable signals
- When writes can be merged without causing errors

For a memory read command or a memory read line command, the VCOP reads one cache line from memory. If the PCI read or read line transaction crosses a cache line boundary, the VCOP starts the read of a new cache line. For a memory read multiple command, the VCOP reads two cache lines from memory. When the PCI transaction finishes the read for the first cache line, the VCOP performs a speculative read of a third cache line. The VCOP continues this prefetching until the end of the transaction.

For PCI writes to memory, streaming is achieved by buffering the transaction in the space available within the I/O sequencer. This allows PCI memory writes to execute with no wait states.

A disconnect occurs if the VCOP runs out of buffer space on writes, or the VCOP cannot supply consecutive data beats for reads within eight PCI bus clocks of each other. A disconnect also occurs if the transaction crosses a 4K page boundary.

15.1.8.3.4 Configuration Access

The PCI responds to remote host generated PCI configuration accesses to the PCI interface. This is indicated by decoding the configuration command along with the VCOP's IDSEL being asserted. A remote host can use PCI Configuration access to address the 256-byte PCI configuration area within the VCOP.

15.1.8.3.5 Special Cycle Command

A special cycle command contains no explicit destination address but is broadcast to all PCI agents. Each receiving agent must determine whether the message is applicable to itself. No assertion of $\overline{\text{PCI_DEVSEL}}$ in response to a special cycle command is necessary.

A special cycle command is like any other bus command in that it has an address phase and a data phase. The address phase starts like all other commands with the assertion of $\overline{\text{PCI_FRAME}}$ and completes when $\overline{\text{PCI_FRAME}}$ and $\overline{\text{PCI_IRDY}}$ are deasserted. Special cycles terminate with an initiator-abort. (In the special cycle case, the received-initiator-abort bit in the configuration status register is not set.)

The address phase contains no valid information other than the command field. Even though there is no explicit address, the address/data lines are driven to a stable state and parity is generated. During the data phase, the address/data lines contain the message type and an optional data field. The message is encoded on the sixteen least-significant bits (AD[15-0]). The data field is encoded on AD[31-16]. When running a special cycle, the message and data are valid on the first clock $\overline{\text{PCI_IRDY}}$ is asserted.

When the CONFIG_ADDRESS register gets written with a bus number of 0x00, the device number is all ones, the function number is all ones and the register number is zero, the next time the CONFIG_DATA register is accessed the VCOP does either a special cycle or an interrupt acknowledge command. When the CONFIG_DATA register is written, the VCOP generates a special cycle encoding on the command/byte enable lines during the address phase, and drives the data from the CONFIG_DATA register onto the address/data lines during the first data phase.

15.1.8.3.6 Interrupt Acknowledge

When the CONFIG_ADDRESS register gets written with a value such that the bus number is 0x00, the device number is all ones, the function number is all ones and the register number is zero, the next time the CONFIG_DATA register is accessed the VCOP does either a special cycle command or an interrupt acknowledge command. When the CONFIG_DATA register is read, the VCOP generates an interrupt acknowledge command encoding on the command/byte enable lines during the address phase. During the address phase, AD[31-0] do not contain a valid address but are driven with stable data and valid parity (PCI_PAR). During the data phase, the byte enable signals determine which bytes are involved in the transaction. The interrupt vector must be returned when $\overline{\text{PCI_TRDY}}$ is asserted.

An interrupt acknowledge transaction can also be issued on the PCI bus by reading from the PCI_INT_ACK register.

15.1.8.4 Error Functions

This section discusses PCI bus errors.

15.1.8.4.1 Parity

During valid 32-bit address and data transfers, parity covers all 32 address/data lines and the 4 command/byte enable lines regardless of whether or not all lines carry meaningful information. Byte lanes not actually transferring data are driven with stable (albeit meaningless) data and are included in the parity calculation. During configuration, special cycle or interrupt acknowledge commands, some address lines are not defined but are still driven to stable values and included in the parity calculation.

Even parity is calculated for all PCI operations: the value of PCI_PAR is generated such that the number of ones on PCI_AD[31–0], PCI_C/$\overline{\text{BE}}$[3–0] and PCI_PAR equals an even number. The PCI_PAR signal is driven when the address/data lines are driven and follow the corresponding address or data by one clock.

The VCOP checks the parity after all valid address phases (the assertion of $\overline{\text{PCI_FRAME}}$) and for valid data transfers ($\overline{\text{PCI_IRDY}}$ and $\overline{\text{PCI_TRDY}}$ asserted) involving the VCOP. When an address or data parity error is detected, the detected-parity-error bit in the configuration space status register is set (see **page 15-24** for details).

15.1.8.4.2 Error Reporting

Except for setting the detected-parity-error bit, all parity error reporting and response is controlled by the parity-error-response bit (see **page 15-23** for details). If the parity-error-response bit is cleared, the VCOP completes all transactions regardless of parity errors (address or data). If the bit is set, the VCOP asserts $\overline{\text{PCI_PERR}}$ two clocks after the actual data transfer in which a data parity error is detected, and keeps $\overline{\text{PCI_PERR}}$ asserted for one clock. The VCOP asserts $\overline{\text{PCI_PERR}}$ when acting as an initiator during a read transaction or as a target involved in a write to system memory. **Figure 15-6** shows the possible assertion points for $\overline{\text{PCI_PERR}}$ if the VCOP detects a data parity error.

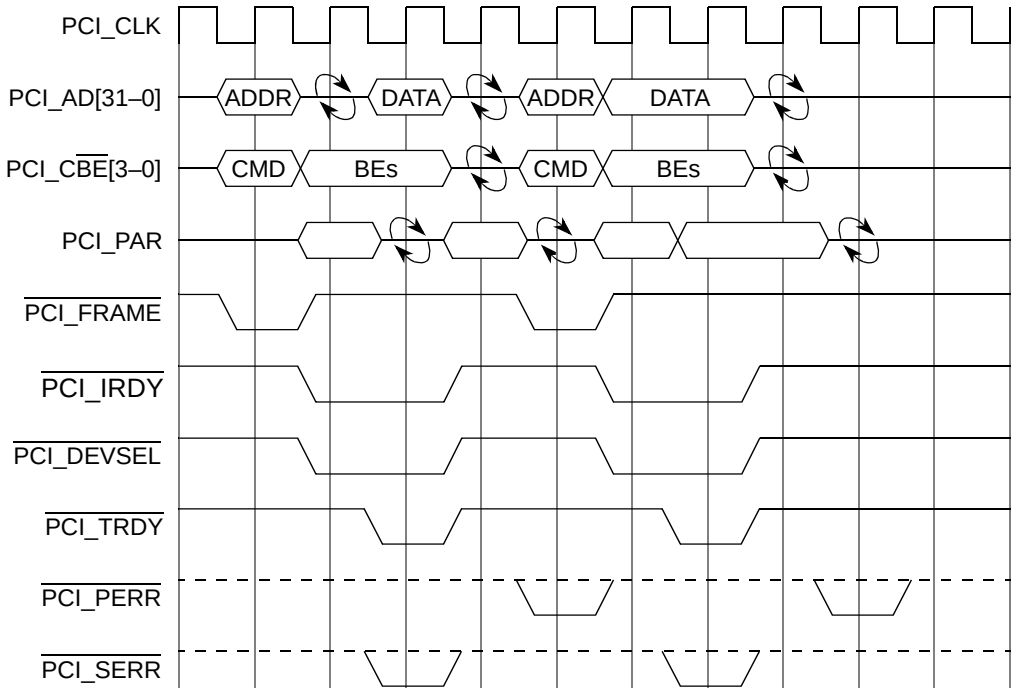


Figure 15-6. PCI Parity Operation

As an initiator, the VCOP attempts to complete the transaction on the PCI bus if a data parity error is detected and sets the data-parity-reported bit in the configuration space status register. If a data parity error occurs on a read transaction, the VCOP aborts the transaction internally. As a target, the VCOP completes the transaction on the PCI bus even if a data parity error occurs. If parity error occurs during a write to system memory, the transaction completes on the PCI bus but is aborted internally, insuring that potentially corrupt data does not go to memory.

When the VCOP asserts PCI_SERR, it sets the signaled-system-error bit in the configuration space status register. Additionally, if the error is an address parity error, the parity-error-detected bit is set; reporting an address parity error on PCI_SERR is conditioned on the parity-error-response bit being enabled in the command register. PCI_SERR is asserted when the VCOP detects an address parity error while acting as a target. **Figure 15-6** shows where the VCOP could detect an address parity error and assert PCI_SERR or where the VCOP, acting as an initiator, checks for the assertion of PCI_SERR signaled by the target detecting an address parity error.

As a target that asserts PCI_SERR on an address parity, the VCOP completes the transaction on the PCI bus, aborting internally if the transaction is a write to system memory. If PCI_PERR is asserted during a VCOP write to PCI, the VCOP attempts to continue the transfer, allowing the target to abort/disconnect if desired. If the VCOP detects a parity error on a read from PCI, the VCOP aborts the transaction internally and continues the transfer on the PCI bus, allowing the target to abort/disconnect if desired.

In all cases of parity errors on the PCI bus, regardless of the parity-error-response bit, information about the transaction is logged in the PCI error control capture register, the PCI error address capture register and the PCI error data capture register; an interrupt is also asserted to the core as an option.

15.1.8.5 PCI Inbound Address Translation

For inbound transactions (transactions generated by an external initiator on the PCI bus where the VCOP responds as a target device), the VCOP only responds to PCI addresses within the windows mapped by the PCI inbound base address registers (PIBARs) or PIMMR base address register (PIMMBACR). If there is an address hit in PIMMRBACR, the PCI address is translated from PCI space to a 32 Mbyte address space starting at address 0xFE000000 in the local memory space. This allows an external initiator to access local memory mapped registers. If there is an address hit in one of the PIBARs, the PCI address is translated from PCI space to local memory space through the associated PCI inbound translation address registers (PITARs). This allows an external initiator to access local memory. Each PIBAR register is associated with a PITAR and PIWAR which are located in the VCOP PCI CSR space. **Figure 15-7** shows an example translation window for inbound memory accesses.

There are three full sets of inbound translation registers, in addition to the PIMMR base address register, allowing four simultaneous translation windows, one to a fixed destination and three programmable. Only two of the programmable windows can be mapped anywhere in the 64-bit PCI address space. Window 0 and the PIMMR window can only be mapped within the lowest 4-Gbyte space. Software can move the programmable translation base addresses during run-time to access different portions of local memory, but the PCI inbound translation windows may not overlap or be translated to the PCI Outbound Window (0xE0000000 to 0xE7FFFFFF in the local memory space).

The translation windows are disabled after reset, that is, after reset, the VCOP acknowledges, by asserting PCI_DEVSEL, only externally initiated transactions toward the PIMMR window on the PCI bus. Other transactions are not acknowledged until the inbound translation windows are enabled.

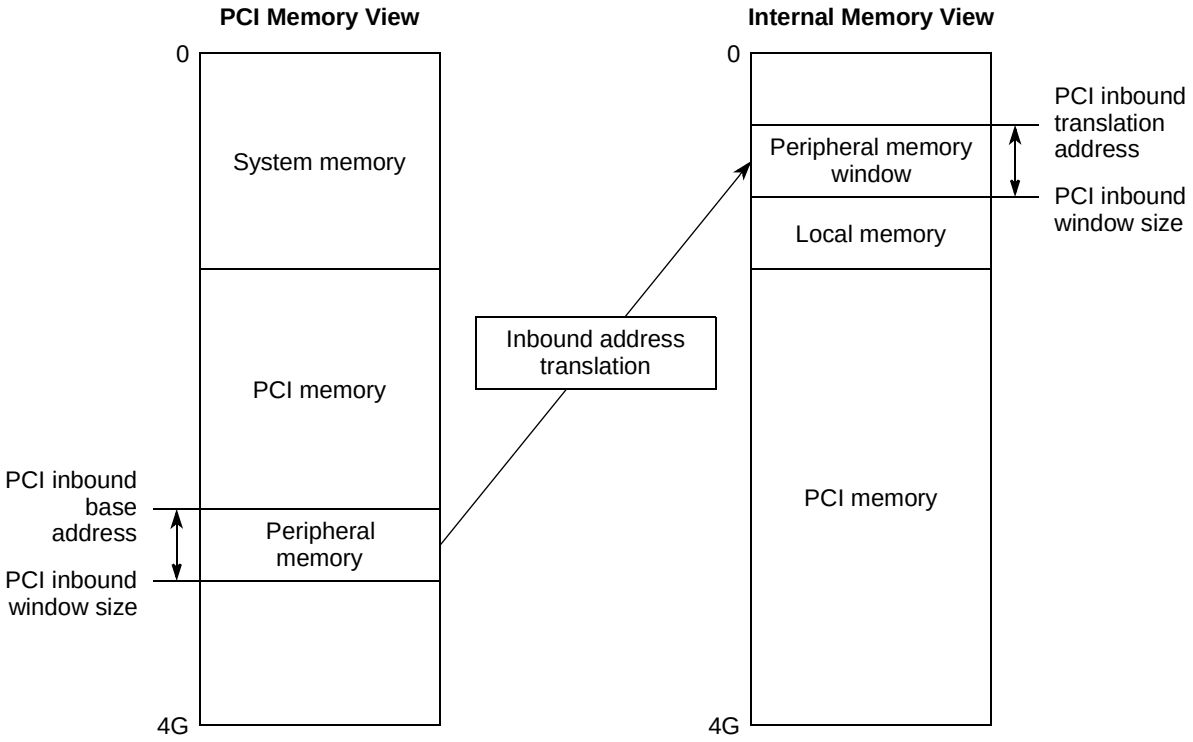


Figure 15-7. Inbound PCI Memory Address Translation

15.1.8.6 PCI Outbound Address Translation

Transactions to the PCI Outbound Window (0xE0000000 to 0xE7FFFFFF in the local memory space) are forwarded to the PCI Configuration Access Registers or to the PCI port. If the address matches the PCI Configuration Access Registers memory space (0xE7FFFFF0 to 0xE7FFFFFF in the local memory area), the transaction is forwarded to one of the PCI Configuration Access Registers. Otherwise, if the address hits any of the outbound translation windows, the transaction is forwarded to PCI port.

Note: Do not attempt to access the PCI Outbound Window until the PCICCR[BMST] bit is set. See **Section 15.2.2.3**, *PCI Command Configuration Register (PCICCR)*, on page 15-23 for details about the BMST bit.

Outbound address translation is provided to allow the outbound transactions to access any address over the PCI memory or I/O space. Translation window base addresses are defined in the PCI outbound base address registers. See **page 15-44** for details. Transactions to these address ranges are issued on the PCI bus with a translated address. The translation addresses are defined

in the associated PCI outbound translation address registers (**page 15-43** for details). **Figure 15-8** shows an example translation window for outbound memory accesses.

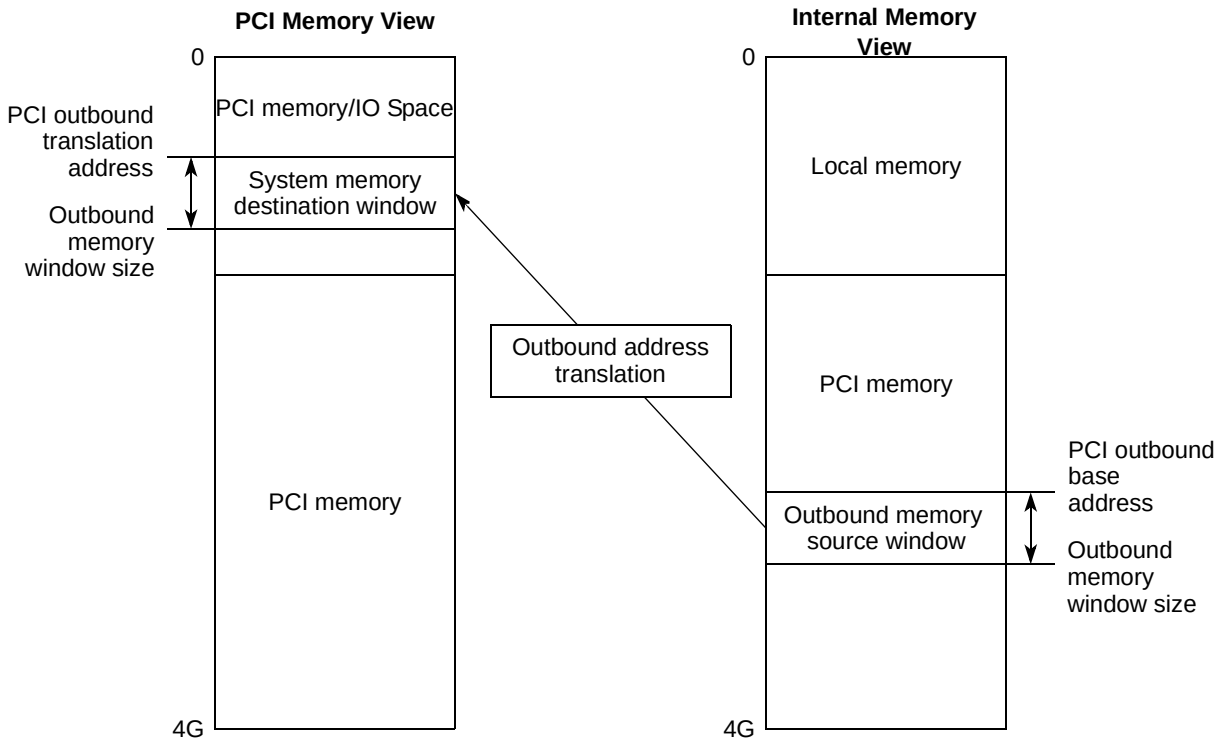


Figure 15-8. Outbound PCI Memory Address Translation

The six sets of outbound translation registers allow six simultaneous translation windows to the PCI port. Software can move and adjust the memory window translations and sizes during run-time. This allows software to access different PCI memory/IO spaces on the fly, but the PCI outbound translation source windows must not overlap. However, outbound translation destination windows can be overlapped.

15.1.8.7 Transaction Ordering

The following rules are applied to maintain proper ordering of transactions:

- An Outbound read transaction from the internal memory pulls out of the PCI any posted Inbound writes that originated on the PCI port and were posted before the read data arrives from the PCI.
- The PCI is always able to accept an Inbound write transaction from PCI port without forcing the PCI port to first accept an Outbound read.

15.1.9 Initialization Sequence

The following sequences must be followed in agent mode:

- Optionally initialize subsystem vendor ID/device ID
- Initialize PCI inbound window size in PIWAR 0–2 desired window size
- Unlock configuration lock in PCI function configuration register

15.2 Programming Model

The VCOP has three types of registers:

- PCI configuration access registers
- PCI configuration space registers
- PCI memory mapped registers

The PCI configuration access registers are used to access PCI internal configuration space and generate special cycle or interrupt acknowledge transactions on the PCI bus. These registers are accessed internally via the PCI outbound window. The PCI configuration access registers include:

- PCI Configuration Address Register (CONFIG_ADDRESS), see **page 15-20**.
- PCI Configuration Data Register (CONFIG_DATA), see **page 15-21**.
- PCI Interrupt Acknowledge Register (PCI_INT_ACK), see **page 15-22**.

Note: The PCI configuration access registers use a base address of: 0xE7FFFFF0.

The PCI configuration space registers are defined by the PCI specification. These registers are accessed by PCI initiators using configuration accesses, or by a local initiator using PCI configuration access registers.The PCI configuration space registers include:

- Vendor ID Configuration Register (VIDCR), see **page 15-22**
- Device ID Configuration Register (DIDCR), see **page 15-23**
- PCI Command Configuration Register (PCICCR), see **page 15-23**
- PCI Status Configuration Register (PCISCR), see **page 15-24**
- Revision ID Configuration Register (RIDCR), see **page 15-25**
- Standard Programming Interface Configuration Register (SPICR), see **page 15-26**
- Subclass Code Configuration Register (SCCR), see **page 15-26**
- Base Class Code Configuration Register (BCCCR), see **page 15-26**
- Cache Line Size Configuration Register (CLSCR), see **page 15-27**
- Latency Timer Configuration Register (LTCR), see **page 15-27**
- Header Type Configuration Register (HTCR), see **page 15-28**

- BIST Control Configuration Register (BISTCCR), see **page 15-28**
- PIMMR Base Address Register Configuration Register (PIMMRBACR), see **page 15-28**
- GPL Base Address Register 0 (GPLBAR0), see **page 15-29**
- GPL Base Address Registers 1–2 (GPLBAR[1–2]), see **page 15-29**
- GPL Extended Base Address Registers 1–2 (GPLEXTBAR[1–2]), see **page 15-30**
- Subsystem Vendor ID Configuration Register (SVIDCR), see **page 15-31**
- Subsystem Device ID Configuration Register (SDIDCR), see **page 15-31**
- Capabilities Pointer Configuration Register (CAPPTRCR), see **page 15-32**
- Interrupt Line Configuration Register (INTLINCR), see **page 15-32**
- Interrupt Pin Configuration Register (INTPINCR), see **page 15-32**
- MIN GNT Configuration Register (MINGNTCR), see **page 15-33**
- MAX LAT Configuration Register (MAXLATCR), see **page 15-33**
- PCI Function Configuration Register (PCIFCR), see **page 15-33**

Note: In addition, some PCI features are controlled by the C2GPRs (see **Chapter 4**, *Chip-Level Arbitration and Switching System (CLASS)*.

The PCI memory-mapped registers are used to manage error functions, general control and status, and address translation control for the inbound and outbound paths. They can be accessed by PCI initiators via the VCOP to the internal memory interface via the PIMMR inbound window. The PCI memory-mapped registers include:

- PCI Error Status Register (PCI_ESR), see **page 15-34**
- PCI Error Capture Disable Register (PCI_ECDR), see **page 15-35**
- PCI Error Enable Register (PCI_EER), see **page 15-36**
- PCI Error Attributes Capture Register (PCI_EATCR), see **page 15-37**
- PCI Error Address Capture Register (PCI_EACR), see **page 15-39**
- PCI Error Extended Address Capture Register (PCI_EEACR), see **page 15-39**
- PCI Error Low Data Capture Register (PCI_ELDCR), see **page 15-40**
- PCI Inbound Translation Address Registers 0–2 (PITAR[0–2]), see **page 15-40**
- PCI Inbound Base Address Registers 0–2 (PIBAR[0–2]), see **page 15-41**
- PCI Inbound Extended Base Address Registers 1–2 (PIEBAR[1–2]), see **page 15-41**
- PCI inbound window attributes registers 0–2 (PIWAR[0–2]), see **page 15-42**
- PCI Outbound Translation Address Registers 0–5 (POTAR[0–5]), see **page 15-43**
- PCI Outbound Base Address Registers 0–5 (POBAR[0–5]), see **page 15-44**
- PCI Outbound Comparison Mask Registers 0–5 (POCMR[0–5]), see **page 15-44**
- Discard Timer Control Register (DTCR), see **page 15-46**

Note: The PCI memory-mapped registers use a base address of: 0xFFF7A000.

15.2.1 PCI Configuration Access Registers

This section describes the registers that are used to allow an internal bus initiator to access the PCI internal PCI configuration space, and generate special cycle or interrupt acknowledge transactions on the PCI bus.

15.2.1.1 PCI Configuration Address Register (CONFIG_ADDRESS)

CONFIG_ADDRESS PCI Configuration Address Register Offset 0x0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EN	—							BN							
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
	DN					FN			RN						—	
Type	W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CONFIG_ADDRESS register holds the address for an access to the PCI configuration space from the internal bus. This register must be programmed before accessing CONFIG_DATA to perform the transaction. Only 32-bit accesses are permitted. The combination EN = 1, BN = 0, DN = 0, and FN = 0 should be used to access the internal PCI configuration registers. If EN = 1, BN = 0, DN = 31, FN = 7, and RN = 0, writing to CONFIG_DATA generates a special cycle transaction and reading from CONFIG_DATA generates an interrupt acknowledge transaction. **Table 15-3** shows the bit settings of the CONFIG_ADDRESS register.

Table 15-3. CONFIG_ADDRESS Field Descriptions

Bits	Description	Settings
EN 31	**Enable Configuration Transaction** This bit determines the type of transaction to generate.	0 No configuration transaction generated. Accesses are passed through to the PCI bus. 1 Configuration access generated by accessing the CONFIG_DATA register.
— 30–24	Reserved. Write to 0 for future compatibility.	
BN 23–16	**Bus Number** Although these bits are writable, only the value 0 is legal.	0 Allows transactions to PCI internal configuration space, special cycles, and interrupt acknowledge transactions. All other values are invalid.
DN 15–11	**Device Number** Although these bits are writable, only the values 0 and 31 are legal.	00000 Transactions to PCI internal configuration space enabled. 11111 Special cycles and interrupt acknowledge transactions enabled. All other values are invalid.

Table 15-3. CONFIG_ADDRESS Field Descriptions (Continued)

Bits	Description	Settings
FN 10–8	**Function Number** Although these bits are writable, only the values 0 and 7 are legal.	000 Transactions to PCI internal configuration space enabled. 111 Special cycles and interrupt acknowledge transactions enabled. All other values are invalid.
RN 7–2	**Register Number** This field selects a 4-byte word in the PCI controller internal configuration space.	
— 1–0	Reserved. Write to 0 for future compatibility.	

15.2.1.2 PCI Configuration Data Register (CONFIG_DATA)

CONFIG_DATA PCI Configuration Data Register Offset 0x4

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

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

An access to CONFIG_DATA usually generates an access to the PCI Controller internal configuration space if the EN bit of CONFIG_ADDRESS is set. There are some exceptions contained in the description of CONFIG_ADDRESS. This register may be accessed with a byte, 16-bit, or 32-bit access, depending on the width of the register targeted by the transaction. **Table 15-4** shows the CONFIG_DATA bit field.

Table 15-4. CONFIG_DATA Field Descriptions

Bits	Description
CFG_DATA 31–0	**Configuration Data** Contains data used for transactions to the PCI controller internal configuration space and for special cycle transactions.

15.2.1.3 PCI Interrupt Acknowledge Register (PCI_INT_ACK)

PCI_INT_ACK PCI Interrupt Acknowledge Register Offset 0x8

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

x = undefined value

Reading this register generates an interrupt acknowledge transaction on the PCI bus. The value that is read is undefined.

15.2.2 PCI Configuration Space Registers

This section describes the PCI configuration space registers. Some fields are common to registers in both spaces to ensure consistency. These fields are discussed in the register definitions. The offsets are within the defined PCI configuration space.

15.2.2.1 Vendor ID Configuration Register (VIDCR)

VIDCR Vendor ID Configuration Register Offset 0x00

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

Table 15-6 shows the bit settings of the VIDCR.

Table 15-5. DIDCR Field Descriptions

Bits	Description
VID 15–0	**Vendor ID** This field identifies the vendor ID. For Freescale Semiconductor, the VID is 0x1957.

15.2.2.2 Device ID Configuration Register (DIDCR)

DIDCR Device ID Configuration Register Offset 0x02

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

Table 15-6 shows the bit settings of the DIDCR.

Table 15-6. DIDCR Field Descriptions

Bits	Description
DID 15–0	**Device ID** This field identifies the device ID. For the MSC8144, the ID is 0x1400.

15.2.2.3 PCI Command Configuration Register (PCICCR)

PCICCR PCI Command Configuration Register Offset 0x04

Bit	15	14	13	12	11	10	9	8
	—						FB-B	SERREN
Type	R							R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—	PERRR	—	MWI	SC	BMST	MEM	IO
Type	R	R/W	R	R	R	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

Table 15-7 shows the bit settings of the PCICCR.

Table 15-7. PCICCR Field Descriptions

Bits	Description	Settings
— 15–10	Reserved. Write to 0 for future compatibility.	
FB-B 9	**Fast Back-to-Back** Hard-wired to 0.	
SERREN 8	**SERR Enable** The enable bit for the SERR driver. Address parity errors are reported (PCI_SERR asserted) only if this bit and PERRR (bit 6) are set (1).	0 SERR disabled 1 SERR enabled
— 7	Reserved. Write to 0 for future compatibility. The field is hard-wired to 0 internally	
PERRR 6	**Parity Error Response** This bit controls the VCOP response to a parity error.	0 Parity errors ignored 1 Standard parity error handling
— 5	Reserved. Write to 0 for future compatibility. The field is hard-wired to 0 internally	

Table 15-7. PCICCR Field Descriptions (Continued)

Bits	Description	Settings
MWI 4	**Memory Write and Invalidate** Hard-wired to 0.	
SC 3	**Special Cycles** Hard-wired to 0.	
BMST 2	**Bus Initiator** Controls the PCI controller ability to be an initiator on the PCI bus.	0 PCI controller does not generate PCI accesses 1 PCI controller behaves as a bus initiator
MEM 1	**Memory Space** Controls the response to memory space accesses.	0 PCI controller does not respond to memory space accesses 1 As a target, the PCI controller responds to memory space accesses
IO 0	**I/O** Hard-wired to 0.	

15.2.2.4 PCI Status Configuration Register (PCISCR)

PCISCR PCI Status Configuration Register Offset 0x06

Bit	15	14	13	12	11	10	9	8
	DPERR	SSERR	RMA	RTA	STA	DEVSEL_T		DPD
Type	R/W					R		R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FB-BC	—	66M	CL	—			
Type	R							
Reset	1	0	1	0	0	0	0	0

This register is used to record status information for PCI bus-related events. Some of the bits are hard-wired to indicate the capabilities of the VCOP. Other bits can be cleared by writing 1 to the bit location. **Table 15-8** shows the bit settings of the PCISCR.

Table 15-8. PCISCR Field Descriptions

Bits	Description	Settings
DPERR 15	**Detected Parity Error.** This bit is set whenever the VCOP detects a parity error on the PCI bus, even if parity error handling is disabled (as controlled by bit 6 in the PCI Command register).	0 No parity error 1 Parity error detected
SSERR 14	**Signaled System Error** This bit is set whenever PCI asserts $\overline{\text{PCI_SERR}}$.	0 $\overline{\text{PCI_SERR}}$ not asserted 1 $\overline{\text{PCI_SERR}}$ asserted

Table 15-8. PCISCR Field Descriptions (Continued)

Bits	Description	Settings
RMA 13	**Received Initiator Abort** This bit is set whenever the VCOP, acting as the PCI initiator on the PCI bus, terminates a transaction (except for a special-cycle) using initiator-abort.	0 Normal operation 1 Received initiator abort
RTA 12	**Received Target Abort** This bit is set whenever a transaction initiated by this VCOP on the PCI bus is terminated by a target-abort.	0 Normal operation 1 Received target abort
STA 11	**Signalled Target Abort** This bit is set whenever the VCOP, acting as the PCI target on the PCI bus, issues a target-abort to a PCI initiator.	0 No parity error 1 Parity error detected
DEVSEL_T 10–9	**DEVSEL Timing** Hard-wired to 00.	
DPD 8	**Initiator Data Parity Error** This bit is set when a data parity error is detected on the PCI bus, if the VCOP is the initiator that initiated the transaction and bit 6 in the PCI command register is set.	0 No parity error 1 Parity error detected
FB-BC 7	**Fast Back-to-Back Capable** Hard-wired to 1.	
— 6	Reserved. Write to 0 for future compatibility. The field is hard-wired to 0 internally	
66M 5	**66-MHz Capable** Hard-wired to 1.	
CL 4	**Capabilities List** Hard-wired to 0.	
— 3–0	Reserved. Write to 0 for future compatibility. The field is hard-wired to 0 internally	

15.2.2.5 Revision ID Configuration Register (RIDCR)

RIDCR Revision ID Configuration Register Offset 0x08

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

Note: The value of this register is represented by x in the register diagram.

Table 15-9 shows the bit settings of the RIDCR.

Table 15-9. RIDCR Field Descriptions

Bits	Description
RID 7–0	**Revision ID** This field specifies a revision code for the PCI controller.

15.2.2.6 Standard Programming Interface Configuration Register (SPICR)

SPICR Standard Programming Interface Configuration Register Offset 0x09

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

This is the lower byte of the class code. **Table 15-10** shows the bit settings of the SPICR.

Table 15-10. SPICR Field Descriptions

Bits	Description
PI 7–0	**Programming Interface** This field is hard-wired internally as 0x00.

15.2.2.7 Subclass Code Configuration Register (SCCR)

SCCR Subclass Code Configuration Register Offset 0x0A

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

This is the middle byte of the class code. **Table 15-11** shows the bit settings of the SCCR.

Table 15-11. SCCR Field Descriptions

Bits	Description
SC 7–0	**Subclass Code** This field is hard-wired internally as 0x80.

15.2.2.8 Base Class Code Configuration Register (BCCCR)

BCCCR Base Class Code Configuration Register Offset 0x0B

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

This is the upper byte of the class code. **Table 15-12** shows the bit settings of the BCCCR.

Table 15-12. BCCCR Field Descriptions

Bits	Description
BCC 7–0	**Base Class Code** This field is hard-wired internally as 0x06.

15.2.2.9 Cache Line Size Configuration Register (CLSCR)

CLSCR Cache Line Size Configuration Register Offset 0x0C

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

Table 15-13 shows the bit settings of the CLSCR.

Table 15-13. CLSCR Field Descriptions

Bits	Description
CLS 7–0	**Cache Line Size** This field represents the cache-line size of the system in terms of 32-bit words. Although the register is writable, only the value 0x08 is legal.

15.2.2.10 Latency Timer Configuration Register (LTCR)

LTCR Latency Timer Configuration Register Offset 0x0D

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

Table 15-14 shows the bit settings of the LTCR.

Table 15-14. LTCR Field Descriptions

Bits	Description
LT 7–3	**Latency Timer** This field specifies, with a granularity of 8 PCI clocks, the length of time that the VCOP, when initiating a transaction, may hold the bus as the result of a bus grant. Refer to the PCI 2.2 specification for the rules by which the VCOP completes transactions when the timer has expired.
— 2–0	Reserved. Write to 0 for future compatibility. The field is hard-wired to 0 internally

15.2.2.11 Header Type Configuration Register (HTCR)

HTCR Header Type Configuration Register Offset 0x0E

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

HTCR specifies the header type and is hard-wired to 0x00.

15.2.2.12 BIST Control Configuration Register (BISTCCR)

BISTCCR BIST Control Configuration Register Offset 0x0F

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

BISTCCR specifies the BIST control and is hard-wired to 0x00.

15.2.2.13 PIMMR Base Address Configuration Register (PIMMRBACR)

PIMMRBACR PIMMR Base Address Configuration Register Offset 0x10

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

The PIMMRBACR is provided to allow access to local memory mapped registers. The Window size and translation are hard-wired and define a 32 Mbyte address space starting at address 0xFE000000 in the local memory space. **Table 15-15** shows the PIMMRBACR bit fields.

Table 15-15. PIMMARBACR Field Descriptions

Bits	Description
BA 31–4	**Base Address** This field defines the low portion of the base address for the internal memory-mapped register space.
PRE 3	**Prefetchable** This read-only bit is hardwired internally to 0.
T 2–1	**Type** Hard-wired internally to 00.
MSI 0	**Memory Space Indicator** Hard-wired internally to 0.

15.2.2.14 GPL Base Address Register 0 (GPLBAR0)

GPLBAR0 GPL Base Address Register 0 Offset 0x14

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

The GPL base address register 0 is provided to allow access to local memory space. This register is closely tied to PIBAR0 and PIWAR0 in the CSR memory space. A write to GPL base address register 0 also causes a change in the base address bits that are not masked according to the IWS field of PIWAR0 in PIBAR0. Note that this write operation does not change the bits that are masked by the IWS field. For read operations, these masked bits always return zeros. **Table 15-16** shows the GPLBAR0 bit fields.

Table 15-16. GPLBAR0 Field Descriptions

Bits	Description
BA 31–12	**Base Address** This field defines the low portion of the base address for the inbound window.
— 11–4	Reserved. Write to 0 for future compatibility. The field is hard-wired to 0 internally.
PRE 3	**Prefetchable** This read-only bit contains the value of the PIWAR0[PF] bit.
T 2–1	**Type** Hard-wired internally to 00.
MSI 0	**Memory Space Indicator** Hard-wired internally to 0.

15.2.2.15 GPL Base Address Registers 1–2 (GPLBAR[1–2])

GPLBAR1 GPL Base Address Registers 1–2 Offset 0x18
GPLBAR2 Offset 0x20

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

The general purpose local access base address registers are provided to allow access to local memory space. These registers are closely tied to PIBAR[1–2] and PIWAR[1–2] in the CSR memory space. A write to a GPL Base Address Register also causes a change in the base address bits that are not masked according to the IWS field of PIWAR[1–2] in the corresponding PIBAR[1–2]. Note that this write operation does not change the bits that are masked by the IWS field. For read operations, these masked bits always return zeros. **Table 15-17** shows the GPLBAR[1–2] bit fields.

Table 15-17. GPLBAR[1–2] Field Descriptions

Bits	Description
BA 31–12	**Base Address** This field defines the low portion of the base address for the inbound window.
— 11–4	Reserved. Write to 0 for future compatibility. The field is hard-wired to 0 internally.
PRE 3	**Prefetchable** This read-only bit contains the value of the PIWAR[1–2][PF] bit.
T 2–1	**Type** Hard-wired internally to 10.
MSI 0	**Memory Space Indicator** Hard-wired internally to 0.

15.2.2.16 GPL Extended Base Address Registers 1–2 (GPLEXTBAR[1–2])

GPLEXTBAR1 GPL Extended Base Address Registers 1–2 Offset 0x1C
GPLEXTBAR2 Offset 0x24

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

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

Two general purpose local access base address registers are provided to allow access to local memory space. These registers are closely tied to PIBAR[1–2], PIEBAR[1–2], and PIWAR[1–2] in the CSR memory space. A write to a GPL extended base address register also causes a change in the base address bits that are not masked, according to the IWS field of PIWAR[1–2] in the corresponding PIBAR[1–2]/PIEBAR[1–2]. Note that this write operation does not change the bits that are masked by the IWS field. For read operations, these masked bits always return zeros. **Table 15-18** shows the GPLEXTBAR[1–2] bit field.

Table 15-18. GPLEXTBAR[1–2] Field Descriptions

Bits	Description
EBA 31–0	**Extended Base Address** This field defines the high portion of the base address for the inbound window.

15.2.2.17 Sub-System Vendor ID Configuration Register (SVIDCR)

SVIDCR Sub-System Vendor ID Configuration Register Offset 0x2C

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

Table 15-20 shows the bit settings of the SVIDCR.

Table 15-19. SVIDCR Field Descriptions

Bits	Description	Settings
SVID 15–0	**Sub-System Vendor ID** This field identifies the board or sub-system that contains this device.	

15.2.3 Sub-System Device ID Configuration Register (SDIDCR)

SDIDCR Sub-System Device ID Configuration Register Offset 0x2E

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

Table 15-20 shows the bit settings of the SDIDCR.

Table 15-20. SDIDCR Field Descriptions

Bits	Description	Settings
SDID 15–0	**Sub-System Device ID** This field identifies the board or sub-system that contains this device.	

15.2.3.1 Capabilities Pointer Configuration Register (CAPPTRCR)

CAPPTRCR Capabilities Pointer Configuration Register Offset 0x34

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

CAPPTRCR specifies the byte offset in the PCI configuration space that contains the first item in the capabilities list.

15.2.3.2 Interrupt Line Configuration Register (INTLINCR)

INTLINCR Interrupt Line Configuration Register Offset 0x3C

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

Table 15-21 describes the bit field of the INTLINCR.

Table 15-21. INTLINCR Field Descriptions

Bits	Description	Settings
IL 7–0	**Interrupt Line** This value is used to communicate interrupt line routing information. The value has no effect on PCI controller operation.	

15.2.3.3 Interrupt Pin Configuration Register (INTPINCR)

INTPINCR Interrupt Pin Configuration Register Offset 0x3D

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

Table 15-22 shows the bit settings of the INTPINCR.

Table 15-22. INTPINCR Field Descriptions

Bits	Description	Settings
INT_PIN 7–0	**Interrupt Pin** This field is hard-wired to 0x00.	

15.2.3.4 MIN GNT Configuration Register (MINGNTCR)

MINGNTCR MIN GNT Configuration Register Offset 0x3E

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

15.2.3.5 MAX LAT Configuration Register (MAXLATCR)

MAXLATCR MAX LAT Configuration Register Offset 0x3F

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

15.2.3.6 PCI Function Configuration Register (PCIFCR)

PCIFCR PCI Function Configuration Register Offset 0x44

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—					MSD	—				CFG_ LOCK	—		TLTD	MLTD	—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1

Table 15-23 shows the bit settings of the PCIFCR.

Table 15-23. PCIFCR Field Descriptions

Bits	Description	Settings
— 15–11	Reserved. Write to 0 for future compatibility.	
MSD 10	**Master Streaming Disable** This bit can prevent the PCI from streaming as an initiator.	0 Streaming is enabled when operating as an initiator. 1 Streaming is disabled when operating as an initiator.
— 9–6	Reserved. Write to 0 for future compatibility.	
CFG_LOCK 5	**Configuration Lock** This bit controls access to the PCI configuration space from the PCI port.	0 Access to the configuration spaces is permitted. 1 Any inbound PCI access to the PCI configuration space is retried.
— 4–3	Reserved. Write to 0 for future compatibility.	

Table 15-23. PCIFCR Field Descriptions (Continued)

Bits	Description	Settings
TLTD 2	**Target Latency Timeout Disable** This bit determines whether the VCOP, while acting as a PCI target, times out when the first data phase of a transaction has not completed in 16 PCI cycles.	0 Target latency timeout enabled. 1 Target latency timeout disabled.
MLTD 1	**Initiator Latency Timer Disable** This bit determines whether the VCOP, while acting as a PCI initiator, terminates a transaction upon the expiration of the initiator latency timer.	0 Initiator latency timer enabled. 1 Initiator latency timer disabled.
0	Reserved. Write to 0 for future compatibility.	

15.2.4 PCI Memory-Mapped Control and Status Registers

This section describes the control and status registers.

15.2.4.1 PCI Error Status Register (PCI_ESR)

PCI_ESR PCI Error Status Register Offset 0x000

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—					APAR	PCI SERR	MP ERR	TP ERR	NO RSP	TABT	—				
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCI_ESR contains status bits for various types of error conditions captured by the VCOP. Each status bit is set when the corresponding error condition is captured. PCI_ESR is a write-1-to-clear type register. A bit is cleared whenever the register is written, and the data in the corresponding bit location is a 1. **Table 15-24** shows the bit settings of the PCI_ESR.

Table 15-24. PCI_ESR Field Descriptions

Bits	Description	Settings
MERR 31	**Multiple Errors** This bit is set if any other bit of this register is 1 and the same error type occurs again.	0 No multiple error 1 Multiple errors detected
— 30–11	Reserved. Write to 0 for future compatibility.	
APAR 10	**Address Parity Error** This bit is set when there is an address parity error on a PCI access initiated by a device other than this VCOP.	0 No error 1 Error detected

Table 15-24. PCI_ESR Field Descriptions (Continued)

Bits	Description	Settings
PCISERR 9	**PCI System Error** This bit is set when the PCI_SERR input signal is asserted. See **Table 15-25** for more information on PCI_SERR.	0 No error 1 Error detected
MPERR 8	**Initiator Parity Error** This bit is set when the PCI_PERR input signal is asserted on a write access initiated by this VCOP or when a data parity error is detected by this VCOP on a read access that it initiated.	0 No error 1 Error detected
TPERR 7	**Target Parity Error** This bit is set when this VCOP is the target of a transaction and the PCI_PERR input signal is asserted on a read access or a data parity error is detected by this VCOP on a write access.	0 No error 1 Error detected
NORSP 6	**No Response** This bit is set when there is no response to a transaction initiated by this VCOP on the PCI bus (no PCI_DEVSEL assertion).	0 Normal transaction response 1 No response
TABT 5	**Target Abort** This bit is set when a PCI target abort occurs on a transaction initiated by this VCOP.	0 No target abort 1 Target abort
— 4–0	Reserved. Write to 0 for future compatibility.	

15.2.4.2 PCI Error Capture Disable Register (PCI_ECDR)

PCI_ECDR PCI Error Capture Disable Register Offset 0x004

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
	—					APAR	PCI SERR	MP ERR	TP ERR	NO RSP	TABT	—				
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The PCI_ECDR controls the capture of the transaction that caused an error. Each bit corresponds to the error condition reported in the PCI_ESR. Note that only the first error is captured, so disabling the capture of some error types may allow greater visibility of the significant errors. **Table 15-25** shows the bit settings of the PCI_ECDR.

Table 15-25. PCI_ECDR Field Descriptions

Bits	Description	Settings
— 31–11	Reserved. Write to 0 for future compatibility.	
APAR 10	**Address Parity Error** This bit disables capture for an address parity error.	0 Capture is enabled. 1 Capture is disabled.

Table 15-25. PCI_ECDR Field Descriptions (Continued)

Bits	Description	Settings
PCISERR 9	**PCI System Error** This bit disables capture for a PCI system error.	0 Capture is enabled. 1 Capture is disabled.
MPERR 8	**Initiator Parity Error** This bit disables capture for an initiator parity error.	0 Capture is enabled. 1 Capture is disabled.
TPERR 7	**Target Parity Error** This bit disables capture for a target parity error.	0 Capture is enabled. 1 Capture is disabled.
NORSP 6	**No Response** This bit disables capture for a no response error.	0 Capture is enabled. 1 Capture is disabled.
TABT 5	**Target Abort** This bit disables capture for a target abort error.	0 Capture is enabled. 1 Capture is disabled.
— 4–0	Reserved. Write to 0 for future compatibility.	

15.2.4.3 PCI Error Enable Register (PCI_EER)

PCI_EER PCI Error Enable Register Offset 0x008

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
	—					APAR	PCI SERR	MP ERR	TP ERR	NO RSP	TABT	—				
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCI_EER enables the assertion of an interrupt for the error conditions reported in the PCI error status register (PCI_ESR). **Table 15-26** shows the bit settings of the PCI_EER.

Table 15-26. PCI_EER Field Descriptions

Bits	Description	Settings
— 31–11	Reserved. Write to 0 for future compatibility.	
APAR 10	**Address Parity Error** Generates an interrupt when the corresponding bit in the PCI_ESR is set.	0 No interrupt. 1 Interrupt generated.
PCISERR 9	**PCI System Error** Generates an interrupt when the corresponding bit in the PCI_ESR is set.	0 No interrupt. 1 Interrupt generated.
MPERR 8	**Initiator Parity Error** Generates an interrupt when the corresponding bit in the PCI_ESR is set.	0 No interrupt. 1 Interrupt generated.

Table 15-26. PCI_EER Field Descriptions (Continued)

Bits	Description	Settings
TPERR 7	**Target Parity Error** Generates an interrupt when the corresponding bit in the PCI_ESR is set.	0 No interrupt. 1 Interrupt generated.
NORSP 6	**No Response** Generates an interrupt when the corresponding bit in the PCI_ESR is set.	0 No interrupt. 1 Interrupt generated.
TABT 5	**Target Abort** Generates an interrupt when the corresponding bit in the PCI_ESR is set.	0 No interrupt. 1 Interrupt generated.
— 4–0	Reserved. Write to 0 for future compatibility.	

15.2.4.4 PCI Error Attributes Capture Register (PCI_EARCR)

PCI_EATCR PCI Error Attributes Capture Register Offset 0x00C

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CMD				—				BE				—		PB	VI
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCI_EATCR stores information associated with the PCI error captured. **Table 15-27** shows the bit settings of the PCI_EATCR.

Table 15-27. PCI_EATCR Field Descriptions

Bits	Description	Settings
— 31	Reserved. Write to 0 for future compatibility.	
ERRTYPE 30–28	**First Error Type** This field is encoded to indicate the type of the first PCI error captured.	000 Address parity error 001 Write data parity error 010 Read data parity error 011 Initiator abort 100 Target abort 101 System error indication received 110 Parity error indication received on a read 111 Parity error indication received on a write

Table 15-27. PCI_EATCR Field Descriptions (Continued)

Bits	Description	Settings
BN 27-24	**Beat Number** This field provides the data beat number on which the error occurred for data parity errors. The value of this field is undefined for other error types.	0000 1st beat 0001 2nd beat 0010 3rd beat 0011 4th beat 0100 5th beat 0101 6th beat 0110 7th beat 0111 8th beat 1000 9th beat or beyond (transaction larger than one cache line) Others Reserved
— 23–22	Reserved. Write to 0 for future compatibility.	
TS 21–20	**Transaction Size** This field contains the size of the transaction in units of doublewords (8 bytes). If the transaction crossed a cache line (32-byte) boundary, this field indicates the number of actual doublewords in the cache line on which the error occurred. This field is valid only if the VCOP was the initiator of the transaction.	00 4 double words 01 1 double word 10 2 double words 11 3 double words
— 19–16	Reserved. Write to 0 for future compatibility.	
CMD 15–12	**PCI Command** This field contains the PCI command PCI_C/$\overline{\text{BE}}$[3–0] of the transaction.	
— 11–8	Reserved. Write to 0 for future compatibility.	
BE 7–4	**PCI Byte Enables** This field contains the PCI byte enables PCI_C/$\overline{\text{BE}}$[3–0] for the data word.	
— 3–2	Reserved. Write to 0 for future compatibility.	
PB 1	**Parity Bit** This bit contains the PCI parity bit for the captured data word.	
VI 0	**Error Information Valid** The bit indicates that the error information captured in this register, PCI_EACR, PCI_EEACR, and PCI_EDCR is valid.	0 No valid error information 1 Error information is valid

15.2.4.5 PCI Error Address Capture Register (PCI_EACR)

PCI_EACR PCI Error Address Capture Register Offset 0x010

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

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

PCI_EACR stores the low portion of the address associated with the first PCI error captured. **Table 15-28** shows the PCI_EACR bit field.

Table 15-28. PCI_EACR Field Descriptions

Bits	Description
PCI_EA 31–0	**PCI Error Address** Contains the low portion of the address associated with the first detected error.

15.2.4.6 PCI Error Extended Address Capture Register (PCI_EEACR)

PCI_EEACR PCI Error Extended Address Capture Register Offset 0x014

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

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

PCI_EEACR stores the high portion of the address associated with the first PCI error captured. **Table 15-29** shows the PCI_EEACR bit field.

Table 15-29. PCI_EEACR Field Descriptions

Bits	Description
PCI_EEA 31–0	**PCI Error Extended Address** Contains the high portion of the address associated with the first detected error.

15.2.4.7 PCI Error Data Low Capture Register (PCI_EDLCR)

PCI_EDLCR PCI Error Data Low Capture Register Offset 0x018

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

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

PCI_EDLCR stores the data associated with the first PCI error captured. **Table 15-30** shows the PCI_EDLCR bit field.

Table 15-30. PCI_EDLCR Field Descriptions

Bits	Description
PCI_EDR 31–0	**PCI Error Data** Contains the data associated with the first detected error.

15.2.4.8 PCI Inbound Translation Address Registers 0–2 (PITAR[0–2])

PITAR0 PCI Inbound Translation Address Registers 0–2 Offset 0x068
PITAR1 Offset 0x050
PITAR2 Offset 0x038

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

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

PITAR[0–2] define the starting point of the corresponding inbound translation windows in the local memory space. **Table 15-31** describes the PITAR[0–2] bit fields.

Table 15-31. PITAR[0–2] Field Descriptions

Bits	Description
— 31–20	Reserved. Write to 0 for future compatibility.
TA 19–0	**Translation Address** This field contains the starting address of the inbound translated address. This 20-bit field corresponds to bits 31–12 of a 32-bit address.

15.2.4.9 PCI Inbound Base Address Registers 0–2 (PIBAR[0–2])

PIBAR0 PCI Inbound Base Address Registers 0–2 Offset 0x070
PIBAR1 Offset 0x058
PIBAR2 Offset 0x040

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

PIBAR[0–2] define the starting point of the corresponding inbound windows in the PCI memory space. A write to a PIBAR[0–2] also causes a change in the corresponding GPL base address register in the PCI configuration space.

Note: Base addresses written to PIBAR[0–2] must be aligned to the Inbound Window Size set defined in the corresponding PIWAR[0–2].

Table 15-32 shows the PIBAR[0–2] bit field.

Table 15-32. PIBAR[0–2] Field Descriptions

Bits	Description
BA 31–0	**Base Address** This field contains the starting address of the inbound window in the PCI memory space. The field corresponds to bits 43–12 of a 64-bit address. In PIBAR0, the upper 12 bits are reserved because only 32-bit addresses are supported.

15.2.4.10 PCI Inbound Extended Base Address Registers (PIEBAR[1–2])

PIEBAR2 PCI Inbound Extended Base Address Registers 1–2 Offset 0x044
PIEBAR1 Offset 0x05C

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

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

PIEBAR[1–2] define the high portion of the starting point of the inbound windows in the PCI memory space. **Table 15-33** shows the PIEBAR[1–2] bit fields.

Table 15-33. PIEBAR[1–2] Field Descriptions

Bits	Description
— 31–20	Reserved. Write to 0 for future compatibility.
EBA 19–0	**Extended Base Address** This field contains the starting address in the PCI memory space of the inbound base address in the PCI memory space. This 20-bit field corresponds to bits 63–44 of a 64-bit address.

15.2.4.11 PCI Inbound Window Attribute Registers 0–2 (PIWAR[0–2])

PIWAR0 PCI Inbound Window Attribute Registers 0–2 Offset 0x078
PIWAR1 Offset 0x060
PIWAR2 Offset 0x048

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

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

PIWAR[0–2] defines the size of an inbound translation window and some properties of the window. **Table 15-34** shows the PIWAR[0–2] bit fields.

Table 15-34. PIWAR[0–2] Field Descriptions

Bits	Description	Settings
EN 31	**Enable** This bit enables the address translation window. PCI addresses that match the definition of the window will be recognized by the VCOP and translated to the local memory space.	0 Address translation is disabled for this window. 1 Address translation is enabled for this window.
— 30	Reserved. Write to 0 for future compatibility.	
PF 29	**Prefetchable** This bit defines whether the transactions that are translated through this window are prefetchable on the internal MBus. Streaming the transactions requires the memory space to be prefetchable.	0 Not prefetchable 1 Prefetchable
— 28–6	Reserved. Write to 0 for future compatibility.	
IWS 5–0	**Inbound Window Size** This field determines the size of the inbound translation window. Inbound translation window size N which is the encoded 2^(N+1) bytes window size. The smallest window is 4 Kbytes (N = 11).	000000–001010Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011110 2-Gbyte window size 011111–111111Reserved

15.2.4.12 PCI Outbound Translation Address Registers 0–5 (PORAR[0–5])

POTAR0 PCI Outbound Translation Address Registers 0–5 Offset 0x100
POTAR1 Offset 0x118
POTAR2 Offset 0x130
POTAR3 Offset 0x148
POTAR4 Offset 0x160
POTAR5 Offset 0x178

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

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

POTAR[0–5] define the starting address of the outbound window in the PCI memory space. **Table 15-35** describes the POTAR[0–5] bit fields.

Table 15-35. POTAR[0–5] Field Descriptions

Bits	Description
— 31–20	Reserved. Write to 0 for future compatibility.
TA 19–0	**Translation Address** This field contains the starting address of the outbound translated address. This 20-bit field corresponds to bits 31–12 of a 32-bit address.

15.2.4.13 PCI Outbound Base Address Registers 0–5 (POBAR[0–5])

POBAR0	PCI Outbound Base Address Registers 0–5	Offset 0x108
POBAR1		Offset 0x120
POBAR2		Offset 0x138
POBAR3		Offset 0x150
POBAR4		Offset 0x168
POBAR5		Offset 0x180

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

POBAR[0–5] define the starting address of the corresponding outbound window in the local memory space. **Table 15-36** describes the POBAR[0–5] bit fields.

Table 15-36. POBAR[0–5] Field Descriptions

Bits	Description
— 31–20	Reserved. Write to 0 for future compatibility.
BA 19–0	**Base Address** This field contains the starting address of the outbound translated window. This 20-bit field corresponds to bits 31–12 of a 32-bit address.

15.2.4.14 PCI Outbound Comparison Mask Registers 0–5 (POCMR[0–5])

POCMR0	PCI Outbound Comparison Mask Registers 0–5	Offset 0x110
POCMR1		Offset 0x128
POCMR2		Offset 0x140
POCMR3		Offset 0x158
POCMR4		Offset 0x170
POCMR5		Offset 0x188

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

POCMR[0–5] define the size and destination of the outbound translation window and some properties of the window in the PCI address space. **Table 15-37** shows the POCMR[0–5] bit fields.

Table 15-37. POCMR[0–5] Field Descriptions

Bits	Description	Settings
EN 31	**Enable** This bit enables the address translation window. Local addresses that match the definition of the window will be recognized by the VCOP and translated to the PCI memory space.	0 Address translation is disabled for this window. 1 Address translation is enabled for this window.
IO 30	**I/O Space** This bit determines whether the window is mapped to the PCI memory space or PCI I/O space.	0 Memory space 1 I/O space
SE 29	**Streaming Enable** This bit defines whether the transactions that are translated through this window may be streamed on the PCI bus. When this bit is set, the PCI controller may combine consecutive cache line transfers into one PCI transaction.	0 Streaming disabled 1 Streaming enabled
— 28–20	Reserved. Write to 0 for future compatibility.	
CM 20–0	**Comparison Mask** This field contains the size of the translation window. The bits that are 1 in this field indicate bits of the transaction address that should be matched to the value in the PCI outbound base address register and translated in case of a match. This field corresponds to the most-significant 20 bits of a 32-bit address. 15-38 lists the CM values.	See **Table 15-38**.

Table 15-38. Comparison Mask (CM) Values

Value	Meaning
1111_1000_0000_0000_0000	128 MB
1111_1100_0000_0000_0000	64 MB
1111_1110_0000_0000_0000	32 MB
1111_1111_0000_0000_0000	16 MB
1111_1111_1000_0000_0000	8 MB
1111_1111_1100_0000_0000	4 MB
1111_1111_1110_0000_0000	2 MB
1111_1111_1111_0000_0000	1 MB
1111_1111_1111_1000_0000	512 KB
1111_1111_1111_1100_0000	256 KB
1111_1111_1111_1110_0000	128 KB
1111_1111_1111_1111_0000	64 KB
1111_1111_1111_1111_1000	32 KB
1111_1111_1111_1111_1100	16 KB
1111_1111_1111_1111_1110	8 KB

Table 15-38. Comparison Mask (CM) Values (Continued)

Value	Meaning
1111_1111_1111_1111_1111	4 KB
Others	Reserved

15.2.4.15 Discard Timer Control Register (DTCR)

DTCR Discard Timer Control Register Offset 0x1F8

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

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

DTCR configures the discard timer used to put a time limit on PCI delayed read transactions from non-prefetchable memory. The purpose of the timer is to release the stuck buffers in case of malfunctioning or disconnected initiators that never return to read the data. In the same way that prefetchable reads can be discarded whenever the PCI is full and needs to allocate another buffer. Non-prefetchable delayed reads may not be discarded until the originator actually gets the data. **Table 15-39** shows the DTCR bit fields.

Table 15-39. DTCR Field Descriptions

Bits	Description	Settings
EN 31	**Enable** This bit enables the discard timer.	0 Disabled 1 Enabled
— 30–24	Reserved. Write to 0 for future compatibility.	
PTV 23–0	**Preload Timer Value** This field contains the preload value for the discard timer. PCI delayed reads from non-prefetchable address space are discarded after (2^{24} – PTV) internal clock cycles if the initiator has not repeated the transaction. 0xFFFFFF is not valid for PTV. For example, to discard a delayed completion if the PCI initiator has not repeated the transaction in 2^{15} PCI clocks, **Note:** Assuming the internal frequency is twice the PCI frequency, the PTV should equal $2^{24} – 2^{16}$ (0xFF0000).	

Serial RapidIO® Controller 16

The RapidIO controller supports a high-performance, point-to-point, low-pin-count, packet-switched system-level interconnect that can be used in a variety of applications as an open standard. The RapidIO architecture provides a rich variety of features, including high data bandwidth, low-latency capability, and support for high-performance I/O devices. RapidIO technology provides message passing and software-managed programming models. The MSC8144E serial RapidIO subsystem consists of a serial RapidIO controller and a RapidIO message unit (RMU), both of which comply with the *RapidIO Interconnect Specification, Revision 1.2* (see **Figure 16-1**). The MSC8144E device can directly connect either to a host, another MSC8144E device, or a serial RapidIO switch. Each port in the switch is point-to-point connected to the MSC8144E device through a serial RapidIO link that typically carries packets in both directions. Packets ready for processing are transported from the host to the MSC8144E, and the processed packets are transported from the MSC8144E device back to the host.

The Serial RapidIO controller directs the traffic flow between the MSC8144E and any other RapidIO device through the RMU for messages and doorbells and through the RapidIO DMA channels for NWRITEs, NWRITE_Rs, NREADs, and SWRITEs.

The host and the MSC8144E communicate as follows:

- The host sends messages to the destination MSC8144E 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 MSC8144E and the MSC8144E 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 MSC8144E 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 MSC8144E 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 MSC8144E 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.

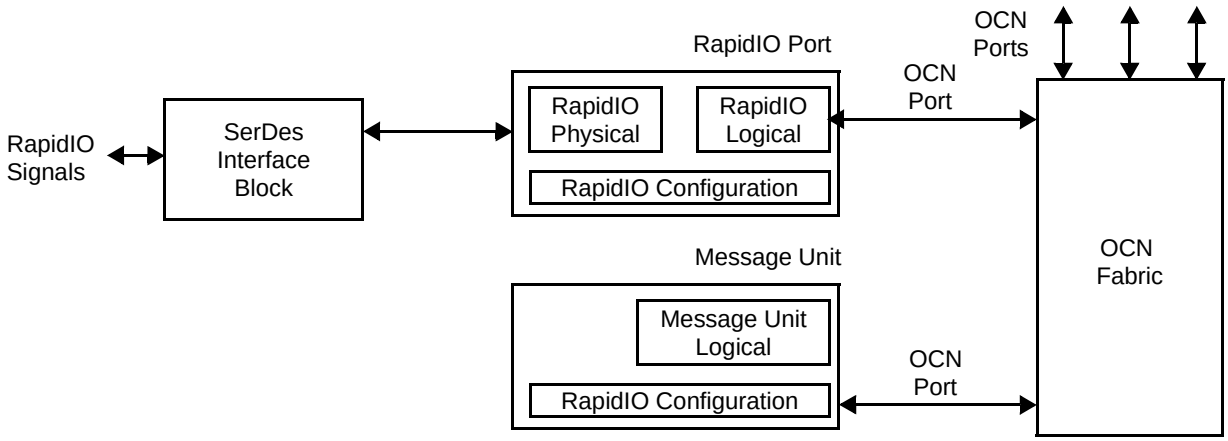


Figure 16-1. RapidIO Controller and RMU

16.1 Introduction

This section summarizes the Serial RapidIO controller features, operating modes, and signal functionality.

16.1.1 Features

The RapidIO port incorporates the following features of the *RapidIO Interconnect Specification, Revision 1.2*:

- Small or large size transport information field.
- 34-bit addressing.
- Up to 256-byte data payload.
- Up to eight outstanding unacknowledged RapidIO transactions.
- Hardware recovery only.
- All transaction flows and all priorities.
- Register and register bit extensions as described in the *RapidIO Interconnect Specification, Revision 1.2*, Part VIII: Error Management Extensions Specification.
- Packet types as defined in the *RapidIO Interconnect Specification, Revision 1.2*, Part II: Message Passing Logical Specification. Type 10 (send a short message) and Type 11 (send a message).
- NWRITE, SWRITE, and NWRITE_R write operations.
- Receiver-controlled flow control only.

- External message passing unit can handle inbound and outbound message passing, inbound and outbound doorbells, and inbound maintenance port-write operations.
- Inbound transactions to the configuration registers are limited to 32-bit accesses.
- Accepts and generates packet types as defined in the *RapidIO Interconnect Specification, Revision 1.2, Part II: Message Passing Logical Specification*. Specifically: Type 10 (send/receive a short message), and Type 11 (send/receive a message).
- Accepts and generates NREAD, NWRITE, SWRITE, and NWRITE_R, and MAINTENANCE read and write operations.
- Outbound maintenance transactions can be any valid size.

The RMU incorporates the following features of the *RapidIO Interconnect Specification, Revision 1.2*:

- Two outbound message controllers.
- Two inbound message controllers.
- One outbound doorbell controller.
- One inbound doorbell controller.
- One inbound port-write controller.

The RMU incorporates the following general features of the RapidIO specification:

- Small or large size transport information field.
- All transaction flows and all priorities.
- Register and register bit extensions as described in Part VIII: Error Management Extensions Specification of the *RapidIO Interconnect Specification, Revision 1.2.*

The RMU supports the following user-defined features:

- Performance monitor interface.

The RapidIO endpoint incorporates the following user-defined features:

- Nine outbound ATMU windows.
- Five inbound ATMU windows.
- Logical outbound packet time-to-live counter to prevent local processor from hanging when the RapidIO interface fails.
- Accept-all mode of operation for failover support.
- RapidIO random bit error injection.
- Performance monitor interface for selected events.

RapidIO endpoint does not support or has limited support for the following features of the *RapidIO Interconnect Specification, Revision 1.2*:

- 50-bit and 66-bit addressing.
- Software assisted error recovery.
- ATOMIC transaction in either direction.
- Coherent (CC-NUMA) transactions.
- Transmitter-controlled flow control.
- No support for multicast event control symbols.

The RapidIO endpoint incorporates the following features of RapidIO 1x/4x LP-serial interfaces:

- Both 1x and 4x LP-serial link interfaces.
- Transmission rates of 1.25, 2.5, and 3.125 Gbaud (data rates of 1.0, 2.0, and 2.5 Gbps, respectively).
- Auto detection of 1x and 4x mode operation during port initialization.
- Error detection for packets and control symbols.
- Link initialization, synchronization, error recovery, and time-out.

RapidIO endpoint does not support the following features of RapidIO 1x/4x operation:

- RapidIO endpoint cannot be configured as four 1x ports.

16.1.2 Operating Modes

The main operating modes of the RapidIO port are as follows:

- 1x or 4x LP-Serial link interfaces.
- Transmission rates of 1.25. 2.5, and 3.125 Gbaud (data rates of 1.0, 2.0, and 2.5 Gbps, respectively).
- Small or large size transport information field.
- Accept-all mode of operation; all packets are accepted regardless of the target ID.

The main operating modes of the message unit are as follows:

- Outbound message controllers:
  - *Direct mode*. There are no descriptors, and software must initialize the required fields before starting a transfer.
  - *Chaining mode*. Software must initialize descriptors in memory and the required fields before starting a transfer.
  - *Multicast mode*. Single-segment messages can be transferred up to 32 destinations.
- Inbound message controllers:
  - *Direct mode*. There are no descriptors, and software must initialize the required fields before starting a transfer.

16.1.3 1x/4x LP-Serial Signals

Table 16-1 describes the serial RapidIO signal functionality. For details on electrical characteristics, refer to the *RapidIO Interconnect Specification, Revision 1.2*.

Table 16-1. Serial Rapid I/O Interface Signals

Port Name	Description	System Connection	I/O	Active State	Reset
$\overline{\text{SD_TX[0–3]}}$/ SD_TX[0–3]	**Transmit Data** Serial data output for the 1x or 4x link. One differential pair output per link. This implementation supports data rates of 1.25, 2.5, or 3.125 Gbaud.	Primary output pads. Asynchronous outputs.	O	—	x
$\overline{\text{SD_RX[0–3]}}$/ SD_RX[0–3]	**Receive Data** Serial data input for 1x or 4x link. One differential pair input per link. This implementation supports data rates of 1.25, 2.5, or 3.125 Gbaud.	Primary input pads. Asynchronous inputs.	I	—	x

16.2 RapidIO Interface Basics

This section summarizes the RapidIO transactions, packet format, and control symbols. It also discusses how the configuration registers are accessed via the RapidIO packets and the operation of the ATMU translation windows for translating RapidIO addresses to local physical addresses and vice versa.

16.2.1 RapidIO Transactions

The RapidIO endpoint supports limited inbound and outbound RapidIO transactions and all RapidIO message passing transactions, as listed in **Table 16-2**.

Table 16-2. RapidIO I/O Inbound Transactions

RapidIO Transaction	ftype	ttype	Status	Description
NREAD	0010	0100	NA	Read
NWRITE	0101	0100		Write with no response
NWRITE_R		0101		Write with response
SWRITE	0110	N/A		Streaming-Write
MAINT read	1000	0000		Maintenance read
MAINT write		0001		Maintenance write
MAINT read response		0010	0000	Done maintenance read response
			0111	Error response
MAINT write response		0011	0000	Done maintenance write response
			0111	Error response
MAINT port-write		0100	NA	Maintenance port-write[1]
RESPONSE without data	1101	0000	0000	I/O done response
			0111	I/O error response
RESPONSE with data		1000	0000	I/O done response with data

Notes: **1.** Limited to inbound RapidIO packets only.

Table 16-3 summarizes the RapidIO message passing transactions of the RapidIO endpoint

Table 16-3. RapidIO Message Passing Transactions

Message Transaction	ftype	ttype	Status	Description
DOORBELL	1010	NA	NA	Doorbell
MESSAGE	1011			Message
RESPONSE without data	1101	0000	0000	Doorbell done response
			0011	Doorbell retry response
			0111	Doorbell error response
		0001	0000	Message done response
			0011	Message retry response
			0111	Message error response

16.2.2 RapidIO Packet Format

Table 16-4 summarizes the small transport field packet formats of RapidIO transaction types for LP-Serial operation.

Note: RapidIO endpoint limits configuration read and write requests to 32-bit data accesses. The large transport field packet format extends the destination and source IDs to 16-bits each. The MSC8144E supports small and large transport fields (large at default), so, for large transport, the destination and source IDs are 16-bits wide according to the direction of the transaction.

Table 16-4. RapidIO Small Transport Field Packet Format

Transaction	Bits															
	32							32					16			64
	5	3	2	2	4	8	8	4	4	8	8	8	13	1	2	
NREAD	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	ttype	rdsize	src TID	addr			wdptr	xambs	NA
NWRITE_R,	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	ttype	wrsize	src TID	addr			wdptr	xambs	dword 0 → dword n
NWRITE	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	ttype	wrsize	don't care	addr			wdptr	xambs	dword 0 → dword n
SWRITE	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	addr(29), rsv(1)=0, xambs(2)					dword 0 -> dword n			
MAINT read	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	ttype	rd/wr size	src TID	hop cnt	cfg offset		wdptr	rsv=0	NA
MAINT write	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	ttype	rd/wr size	src TID	hop cnt	cfg offset		wdptr	rsv=0	dword 0 (32-bit)
MAINT port-write	ackID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	rd/wr size	rsv=0	hop cnt	rsv=0		wdptr	rsv=0	dword 0 → dword n
MAINT response without data	ackID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	hop cnt	rsv=0				NA
MAINT response with data	ackID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	hop cnt	rsv=0				dword 0 (32-bit)
RESPONSE without data	ackID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	NA					

Table 16-4. RapidIO Small Transport Field Packet Format (Continued)

Transaction	Bits															
	32							32					16			64
	5	3	2	2	4	8	8	4	4	8	8	8	13	1	2	
RESPONSE without data for message	ackID	rsv = 0	prio	tt	ftype	dest ID	src ID	type	status	letter(2), mbox(2), msgseg(4)	NA					
RESPONSE with data	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	type	status	tar TID	dword 0 -> dword n					
DOORBELL	ackID	rsv =0	prio	tt	ftype	des tID	src ID	rsv=0		src TID	Info-msb	Info - lsb	NA			
MESSAGE	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	msglen(4), ssize(4), letter(2), mbox(2), msgseg(4)			dword 0 -> dword n					

16.2.3 RapidIO Control Symbol Summary

Table 16-5 summarizes the 1x/4x LP-Serial control symbols and their format. Refer to the *RapidIO Interconnect Specification, Revision 1.2* Part VI: Physical Layer 1x/4x LP-Serial Specifications, Chapter 4 PCS and PMA Layers for 8B/10B data and special (/PD/, /SC/, idle, sync, skip, align) characters. The 32-bit LP-Serial control symbol is composed of the 8-bit special character and the 24-bit control symbol format.

Table 16-5. 1x/4x LP-Serial Control Symbol Format

Bits						Description
24						
stype0	param0	param1	stype1	cmd	CRC	
3	5	5	3	3	5	
000	pkt_ackID	buf_stat	—		crc	Packet accepted
001	pkt_ackID	buf_stat	—		crc	Packet retry
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.

Table 16-5. 1x/4x LP-Serial Control Symbol Format (Continued)

Bits						Description
24						
stype0	param0	param1	stype1	cmd	CRC	
3	5	5	3	3	5	
100	ackID_stat	buf_stat	—		crc	Status ackID_stat: 00000 Expecting ackID 0 00001 Expecting ackID 1 00010 Expecting ackID 2 00011 Expecting ackID 3 00100 Expecting ackID 4 00101 Expecting ackID 5 00110 Expecting ackID 6 00111 Expecting ackID 7
110	ackID_stat	port_stat	—		crc	Link-response ackID_stat: 00000 Expecting ackID 0 00001 Expecting ackID 1 00010 Expecting ackID 2 00011 Expecting ackID 3 00100 Expecting ackID 4 00010 Expecting ackID 5 00110 Expecting ackID 6 00111 Expecting ackID 7 port_stat: 00010 Error; unrecoverable 00100 Retry stopped 00101 Error stopped 10000 OK
—			000	000	crc	Start of packet
—			001	000	crc	Stomp
—			010	000	crc	End of packet
—			011	000	crc	Restart from retry
—			100	cmd	crc	Link request cmd: 011 Reset the receiving device 100 Return input port status; functions as a restart-from-error control symbol under error conditions
—			111	000	crc	NOP (ignore)

16.2.4 Accessing Configuration Registers via RapidIO Packets

The RapidIO endpoint limits requests to configuration register space to 32-bit data accesses. If the order of completion is important, inbound configuration accesses should be assumed incomplete until an appropriate response is received. There should be only one outstanding configuration request at a time to ensure that requests complete in the intended order.

16.2.4.1 Inbound Maintenance Accesses

There are two recommended methods by which RapidIO transactions can target RapidIO configuration register space in local memory.

One method is based on RapidIO NREAD and NWRITE_R requests hitting a RapidIO address window defined by the Local Configuration Space Base Address Command and Status Register (LCSxBACSR), which is described on **page 16-115**. If external configuration accesses are disabled (LLCR[ECRAB] = 1; see **page 16-146**), any configuration access through the LCSxBACSR window is denied. A 32-bit data payload of all zeros is returned for a non-maintenance configuration read.

Note: Only NWRITE_R requests can access the entire internal CCSR address space. Any other write transaction is denied and does not alter the registers.

The second method is based on RapidIO MAINT requests. This method allows an external device limited access to local RapidIO configuration register space. Any maintenance access beyond the first 64 KB of RapidIO configuration register space is denied (lower 64 KB contains RapidIO architecture registers; upper 64 KB contains RapidIO implementation registers). A 32-bit data payload of all zeros is returned for a read response.

A third method that uses an inbound ATMU window to translate RapidIO NREAD and NWRITE_R requests to configuration accesses is not recommended because it does not support the configuration access protection features offered by the LCSBA1CSR window and RapidIO MAINT requests.

16.2.4.2 RapidIO Non-Maintenance Accesses Using LCSBA1CSR

NREAD and NWRITE_R requests can be used to access RapidIO configuration register space by matching RapidIO address[0–13] with the 14-bit value of the LCSBA1CSR, which is described on **page 16-115**. Inbound requests with RapidIO addresses that fall within the window defined by LCSBA1CSR are translated to the local address range. The LCSBA1CSR hit definition and RapidIO address translation depend on the size of the configuration register space. The system configuration input device_ccsrbar_size[0–1] provides this value. The LCSBACSR hit definition and RapidIO address translation are as follows:

device_ccsrbar_size = 0 (size is 1 MB)—as defined for the MSC8144E

— A window hit is defined as LCSBA1CSR[30–17] matching RapidIO address [0–13].

— addr[0–32] = {CCSRBAR[0–15], 0b00000000000000000} + {0b0000, 0b000000000000, RapidIO address[14–30]}.

If the NREAD/NWRITE_R access hits an inbound ATMU window as well, the LCSBA1CSR window has priority in determining the RapidIO address translation. If an NWRITE/SWRITE request hits the LCSBA1CSR window, an illegal transaction decode error is generated and logged.

16.2.4.3 RapidIO Maintenance Accesses

MAINT requests can be used to access RapidIO configuration register space via the 21-bit configuration offset field (config_offset) from the RapidIO maintenance packet. The RapidIO address translation is performed via two additions:

- Adding a 128 KB multiple that represents the local physical address offset for the RapidIO configuration register space (RCAO).
- Adding the RapidIO packet configuration offset value that represents an index into the RapidIO configuration register space from the start of the device local configuration register space.

The 128 KB local physical address offset is a RapidIO port-common value and is provided by the system configuration input *RapidIO_config_addr_offset[0–4]*. The RapidIO address translation is as follows:

- addr[0–32] = { {CCSRBAR[0–15], 0b00000000000000000} + {0b0000, 0b0000000000, rapidio_config_addr_offset[0–4], 10b0000} + {0b0000, 0b0000000000000000, config_offset[8–20]} }
- addr[0–32] = { {CCSRBAR[0–15], 0b00000000000000000} + {0b0000, 0b0000000000, rapidio_config_addr_offset[0–4]}, 0b0, config_offset[8–20] }

The index into RapidIO configuration register space is represented by config_offset[8–20] only; bits for config_offset[0–7] are ignored.

16.2.4.3.1 Guidelines

The RapidIO endpoint limits configuration register space requests to 32-bit data accesses. If the order of completion is important, assume that inbound configuration accesses are incomplete until an appropriate response is received. It is suggested that only one outstanding configuration request be active at a time to ensure that requests are completed in the intended order. For inbound configuration write results that are immediately used by another transaction, perform an inbound configuration read immediately after the configuration write to ensure that the transaction uses the updated value of the accessed configuration register.

16.2.4.3.2 Outbound Maintenance Accesses

Outbound NREAD_R or NWRITE requests can be translated to a RapidIO maintenance request if the internal generated address falls within the bounds of an outbound ATMU window that is setup for generating a maintenance request. The ATMU window specifies the configuration offset, hop count, source and destination ID, and priority for the outbound RapidIO packet.

Another option to initiate an Outbound Maintenance Access is to use the maintenance requests issued by the RapidIO DMA. This bypasses the outbound ATMU and contains all outbound RapidIO packet information (see **Chapter 17**, *RapidIO Interface Dedicated DMA Controller* for details).

16.2.5 RapidIO ATMU Implementation

The ATMU uses a set of registers to translate RapidIO packets to internal packets on inbound and to translate internal packets to RapidIO packets on outbound. ATMU window misses use the window 0 register set by default, and overlapping window hits result in the use of the lowest number window register set hit. For both inbound and outbound translation, the smallest window size is 4 K and the largest window size is 16 G for inbound translation and 64 G for outbound translation.

The default window register set causes no translation of the transaction address for inbound transactions because the RapidIO address space has 34 bits and the internal interconnect address space has 36 bits. For outbound transactions, the default window maps each of the four 16 G windows to the RapidIO 16 G address space.

The inbound and outbound translation windows must be aligned based on the granularity selected by the size fields. The packet device ID fields are not used in the inbound translation process, only the address field. The RapidIO endpoint implementation allows up to a 34-bit (0–33) RapidIO address and a 36-bit (0–35) internal interconnection address. The MSC8144E is confined to 32-bit internal addresses, therefore the top 4 bits (0–3) of the Inbound translation address and the outbound base address should be set to all 0; setting any of these bits results in undefined behavior.

As with all registers, an external processor writing the ATMU registers should never assume that the write is completed until a response is received.

Note: Booting from a serial RapidIO must always use outbound ATMU window 0.

16.2.5.1 RapidIO Outbound ATMU

All outbound windows have the capability to have 2 or 4 segments, all of which are equal in size, numbered 0–1, or 0–3, respectively. Each segment assigns attributes and the target deviceID for an outbound transaction. All segments of a window translate to the same translation address in the target. Additionally, each segment can be set up with 2, 4, or 8 subsegments, all of which are equal in size. These subsegments allow a single segment to target a number of numerically adjacent target device IDs, and, again, they all translate to the same translation address in the targets. For example, a segment with 8 subsegments can be configured to generate a transaction with the same set of attributes to target deviceIDs 0, 1, 2, 3, 4, 5, 6, or 7, depending on which subsegment is addressed.

Note: Subsegments are only supported when multiple segments are chosen.

This allows a window to be configured so that aliases can be created to the same offset within the target device so that a single window can be used to generate different transaction types. Without segmented windows, achieving the equivalent behavior would require multiple windows. **Figure 16-2** shows an example of this capability. A window is defined to be 4 Kbyte in size, and is defined to have 4 segments and no subsegments. Each segment is assigned to target deviceID 0x05, and each segment is given a different write transaction type attribute - segment 0 is assigned NWRITE, segment 1 is assigned SWRITE, segment 2 is assigned NWRITE_R, and segment 3 is assigned FLUSH. Since all of the segments are assigned to target the same device, by writing to the same offset in each segment, a different write transaction can be generated to the target to the same offset in the target.

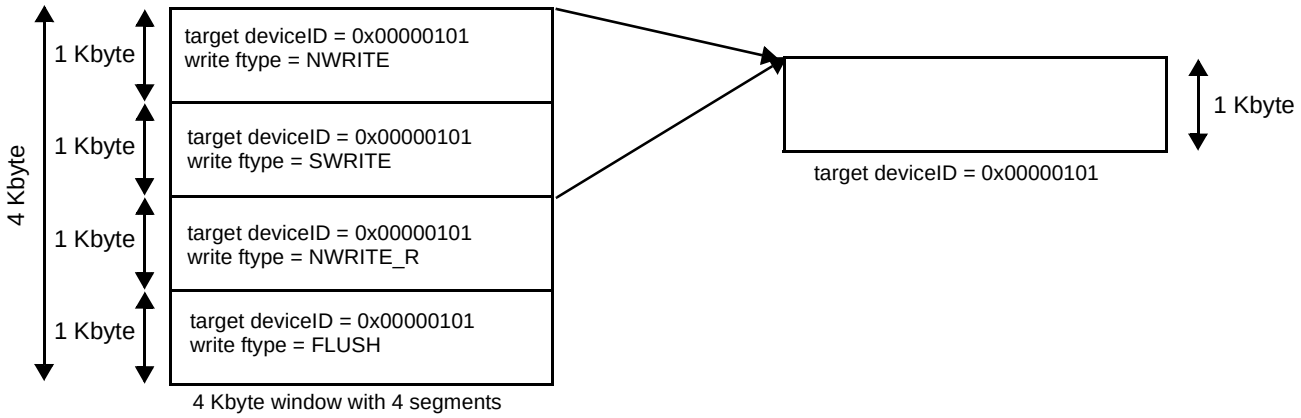


Figure 16-2. Single Target Example

So, writing to offset 0x0 in segment 0 is translated (as defined in the translation address registers) and generates a NWRITE transaction to offset 0x0 in a 1KB window in the target with deviceID = 0x05. A write to offset 0x0 in segment 2 is also translated, to the same offset in the target device as the write to segment 0, but this time a NWRITE_R transaction is generated.

Another use is that the same window can be used to target multiple devices with the same translation offset. Without segmented (and subsegmented) windows, achieving the equivalent behavior would require multiple windows. Figure 1-70 shows an example of this multi-targeting. For example, a 4kB window is set up with 2 segments of 2 subsegments. Each segment is assigned a write type of NWRITE, but each segment and subsegment has a different target deviceID. Segments 0 and 1 are assigned target deviceIDs 4 and 5, and 8 and 9, respectively.

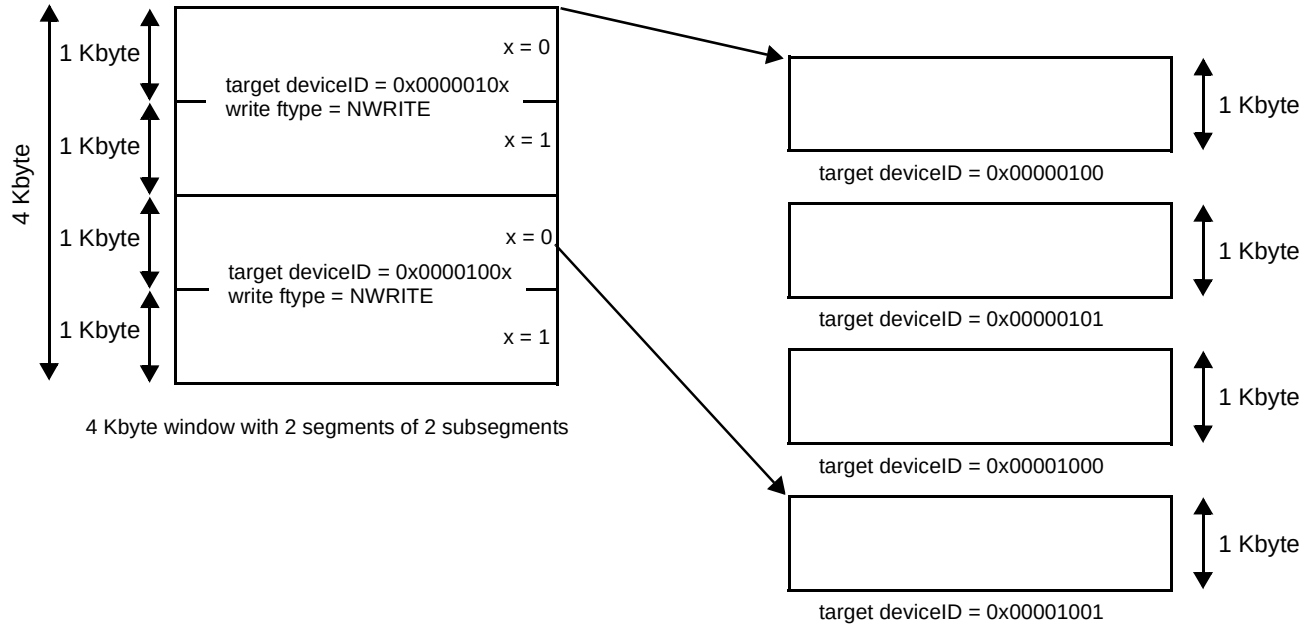


Figure 16-3. Multi-Targeting Example

In this example, a write to offset 0x0 in segment 0 is translated as defined, and a NWRITE transaction is generated targeted to deviceID 4. A corresponding write to segment 1 to offset 0x400 is also translated but also using the assigned deviceID instead of the translation address bits [22–29]. The generated NWRITE transaction has the same target device offset as the write to segment 0, but is instead targeted to deviceID 9. Combinations of aliasing and multi-targeting are also possible for a window.

16.2.5.2 Outbound Windows

RapidIO Endpoint implements nine outbound ATMU translation windows for translating local physical address to RapidIO address.

- Port n RapidIO Outbound Window Translation Address Registers 0–8 define the starting point for the RapidIO address translation and specify the RapidIO destination ID for the transaction.
- Port n RapidIO Outbound Window Attributes Registers 0–8 define the translation window size and specify the RapidIO transaction type and priority for the transaction.

- Port n RapidIO Outbound Window Segment 1–3 Registers 1–8 are used if the ATMU is segmented.

There are eight comparison windows.

- Port n RapidIO Outbound Window Base Address Register 1–8 represents the base address for each of the eight ATMU windows. The base address for each window must be aligned based on the translation window size specified in the Port n RapidIO Outbound Window Attributes Register 1–8.
- Port n RapidIO Outbound Window Translation Address Register 0 and Port n RapidIO Outbound Window Attributes Register 0 are the translation registers for the default ATMU window. It is used only if an NREAD, NWRITE, or NWRITE_R request misses all eight ATMU comparison windows.

16.2.5.3 Window Size and Segmented Windows

For the following discussion, RapidIO address[0–30] consists of xambs[0–1] and address[0–28] fields as defined in the RapidIO request packet format. RapidIO address is a 31-bit double-word physical address (or 34-bit byte address). Internal address[0–32] is a 33-bit double-word physical address (or 36-bit byte address).

The ATMU window hit definition and RapidIO address translation are as follows:

1. 4K window size (smallest window size)
   - A window hit is defined as {BEXADD[0–3], BADD[0–19]} matching internal address [0–23]
   - RapidIO addr[0–30] = {TREXAD[8–9], TRAD[0–19], internal address[24–32]}
2. 8K window size
   - A window hit is defined as {BEXADD[0–3], BADD[0–18]} matching internal address [0–22]
   - RapidIO addr[0–30] = {TREXAD[8–9], TRAD[0–18], internal address[23–32]}
3. 16K window size
   - A window hit is defined as {BEXADD[0–3], BADD[0–17]} matching internal address [0–21]
   - RapidIO addr[0–30] = {TREXAD[8–9], TRAD[0–17], internal address[22–32]}
4. Window sizes 32 K, 64 K, 128 K, 256 K, 512 K, 1 M, 2 M, 4 M, 8 M, 16 M, 32 M, 64 M, 128 M, 256 M, 512 M, 1 G, and 2 G are not shown
5. 4G window size
   - A window hit is defined as {BEXADD[0–3]} matching internal address [0–3]
   - RapidIO addr[0–30] = {TREXAD[8–9], internal address[4–32]}
6. 8G window size
   - A window hit is defined as {BEXADD[0–2]} matching internal address [0–2]

— RapidIO addr[0–30] = {TREXAD[8], internal address[3–32]}

7. 16 G window size

— A window hit is defined as {BEXADD[0–1]} matching internal address [0–1]
— RapidIO addr[0–30] = { internal address[2–32]}

A window can be defined to be non-segmented or segmented depending on the NSEG field definition in the Port n RapidIO Outbound Window Attributes Register.

- A non-segmented window uses the specified RapidIO transaction type (RDTYP, WRTYP) and designated target ID (TGTID) without additional internal address comparison.
- A segmented window divides the specified window size into smaller sub-windows. A segmented window can be further divided into sub-segments as defined by the NSSEG field definition. The use of segmented and sub-segmented windows requires additional internal address comparison. There are two reasons for using a segmented ATMU window: allow a single ATMU window to generate different transactions types; allow a single ATMU window to generate multiple RapidIO target IDs.

Table 16-6 lists the various combination options.

Table 16-6. Outbound ATMU Window Segments

Number of Segments (NSEG)	Number of Sub-Segments (NSSEG)	Transaction Type Given by the Segment Register #	Target ID Given by Segment Register # (or elsewhere)	Comments
0	0	0[1]	0[2]	Non-segmented window.
2	0	0[1], 1	0[2]	Segmented windows generating two transaction types.
4	0	0[1], 1, 2, 3	0[2]	Segmented windows generating four transaction types.
2	2	0[1], 1	0[2], 1 with TGTID[7] defined by the sub segment #	Segmented windows generating two target IDs per segment (4 total)
2	4	0[1], 1	0[2], 1 with TGTID[6–7] defined by the sub-segment #	Segmented windows generating four target IDs per segment (8 total)
2	8	0[1], 1	0[2], 1 with TGTID[5–7] defined by the sub-segment #	Segmented windows generating eight target IDs per segment (16 total)
4	2	0[1], 1, 2, 3	0[2], 1, 2, 3 with TGTID[7] defined by the sub-segment #	Segmented windows generating two target IDs per segment (8 total)
4	4	0[1], 1, 2, 3	0[2], 1, 2, 3 with TGTID[6–7] defined by the sub-segment #	Segmented windows generating four target IDs per segment (16 total)

Table 16-6. Outbound ATMU Window Segments

Number of Segments (NSEG)	Number of Sub-Segments (NSSEG)	Transaction Type Given by the Segment Register #	Target ID Given by Segment Register # (or elsewhere)	Comments
4	8	0[1], 1, 2, 3	0[2], 1, 2, 3 with TGTID[5–7] defined by the sub-segment #	Segmented windows generating eight target IDs per segment (32 total)

Notes: 1. Transaction type is given RDTYP[0–3] and WRTYP[0–3] in the RapidIO Outbound Window Attributes Register (ROWAR)
2. Target ID is given TREXAD[0–7] in the RapidIO Outbound Window Translation Address Register (ROWTAR)

The use of segmented windows impacts only the RapidIO transaction type or the destination ID. The internal address translation is a function of the Port n Outbound Window Translation Address Register and the translation window size.

16.2.5.3.1 Valid Hits to Multiple ATMU Windows

If a request hits multiple ATMU windows, window 1 has the highest priority of the nine outbound ATMU windows (windows 1–8, default). Window 2 is given the next highest priority and is followed by windows 3 through 8. The default window has the lowest priority. If a request hits (base address match) multiple ATMU windows and the transaction end address is contained within the boundary of each hit window and does not extend into another ATMU window, the translation window is the highest priority window that is hit.

If a lower priority window is programmed to lie entirely within a higher priority window, then it is possible for a transaction to cross window boundaries. Although not a practical programming application, the RapidIO endpoint handles this as follows:

1. If a request hits (base address match) an ATMU window (1–8) and the transaction end address extends into another ATMU window with lower priority but is still contained within the boundary of the hit window, the translation window is the hit window.

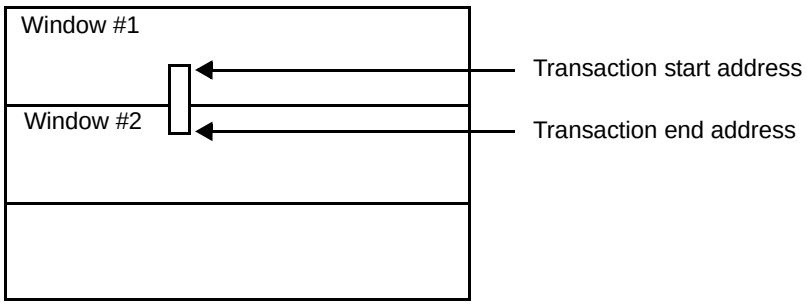


Figure 16-4. Valid Hit that Extends Into a Lower Priority Window

2. If a request hits (base address match) multiple ATMU windows (1–8) and the transaction end address extends beyond the boundary of a lower priority hit window but

is still contained within the boundary of a higher priority hit window, the translation window is the highest priority window that is hit.

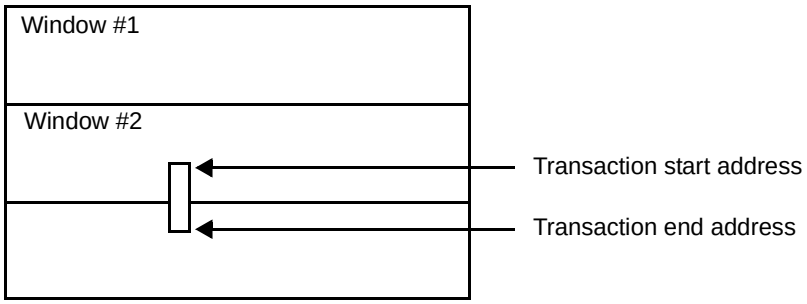


Figure 16-5. Valid Hit that Extends Beyond the Window Boundary

16.2.5.3.2 Window Boundary Crossing Errors

If a higher priority window is programmed to lie entirely within a lower priority window, then it is possible for a transaction to cross window boundaries. The RapidIO endpoint handles this as follows:

1. If a request hits (base address match) an ATMU window (1–8, default) and the transaction end address extends into another ATMU window with higher priority, an

ATMU crossed boundary error is generated and logged. The outbound request is discarded.

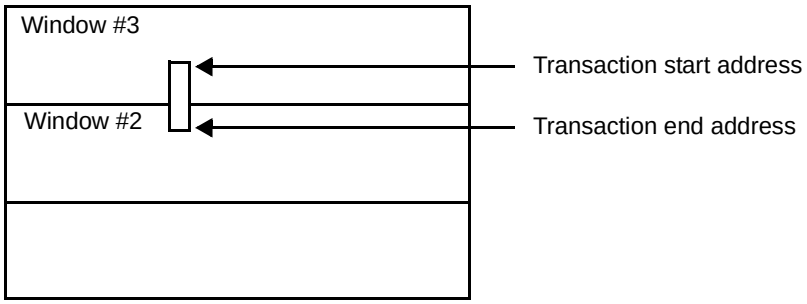


Figure 16-6. Boundary Crossing Error Due to Extension Into a Higher Priority Window

2. If a request hits multiple ATMU windows (1–8, default) and transaction end address extends beyond the boundary of a higher priority hit window, an outbound ATMU crossed boundary error is generated and logged. The outbound request is discarded.

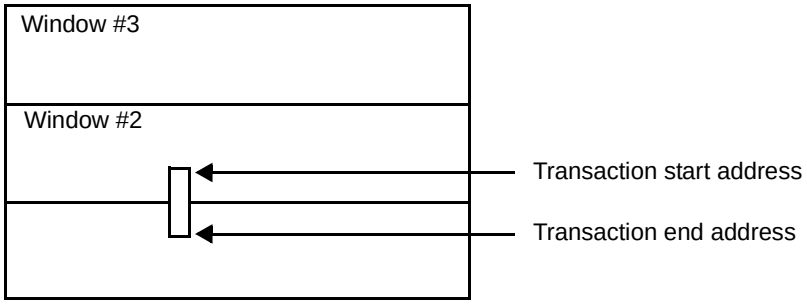


Figure 16-7. Boundary Crossing Error Due to Extension Beyond the Higher Priority Window Boundary

3. If a request hits (base address match) an ATMU window (1–8) and the transaction end address exceeds the size of the window, an outbound ATMU crossed boundary error is generated and logged. The outbound request is discarded.

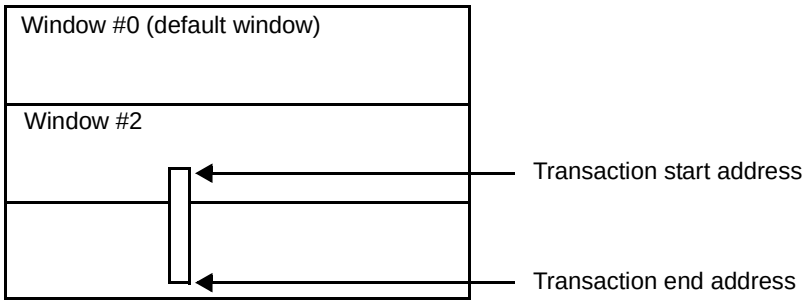


Figure 16-8. Boundary Crossing Error Due to Transaction Size Exceeding the Window Size

An internal error response is generated for internal requests that require a response. Boundary crossing errors (outbound ATMU boundary crossing, segment boundary crossing, and sub-segment boundary crossing) are logged in the LTLEDCSR[OACB] configuration register field. If a request misses all ATMU windows (1–8) and the transaction's end address exceeds the maximum size of the default window, an outbound ATMU crossed boundary error is not generated. The outbound request is forwarded to the RapidIO target device.

16.2.5.4 RapidIO Inbound ATMU

The RapidIO endpoint has five inbound ATMU translation windows for translating RapidIO addresses to local physical addresses. ATMU registers are used for inbound transactions. Their purpose is to translate RapidIO packets to on-device interconnect packets. ATMU window misses use the window 0 register set by default, and overlapping window matches result in the use of the lowest-number window register set in the match. For inbound translation, the smallest window size is 4 KB and the largest window size is 16G. The default window register set causes no translation of the transaction address for inbound transactions. The inbound translation windows must be aligned on the basis of the granularity selected by the size fields. The packet device ID fields are not used in the inbound translation process, only the address field.

The RapidIO endpoint implementation allows up to a 34-bit (0–33) RapidIO address and a 36-bit (0–35) internal addressing. The MSC8144 device is confined to 32-bit addresses, so the top 4 bits (0–3) of the inbound translation address should be set to all 0s. Other settings result in undefined behavior. An external processor should not assume that a write to any ATMU register is complete until a response is received. **Table 16-7** describes the registers for configuring the window parameters, along with the number of the page where each register is described in detail.

Table 16-7. ATMU Registers for Configuring Window Parameters

ATMU Registers	Acronym	Description	Page
Port 0 RapidIO Inbound Window Translation Address Registers 1–4	P0RIWTAR[1–4]	Define the starting-point for the RapidIO address translation.	**page 16-164**
Port 0 RapidIO Inbound Window Attributes Registers 1–4	P0RIWAR[1–4]	Define the translation window size and specify the internal priority, attributes, and the internal target port for the transaction.	**page 16-166**
Configuring the Four Comparison Windows			
The Port 0 RapidIO Inbound Window Base Address Registers 1–4	P0RIWBAR [1–4]	Represent the base address for each ATMU window. The base address must be aligned based on the translation window size specified in P0RIWAR 1–4	**page 16-166**
Port 0 RapidIO Inbound Window Translation Address Registers 0	P0RIWTAR0	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 16-164**
Port 0 RapidIO Inbound Window Attributes Register 0	P0RIWAR0		**page 16-166**

For the following discussion, RapidIO address[0–30] consists of xambs[0–1] and the address[0–28] fields as defined in the RapidIO request packet format. The RapidIO address is a 31-bit double-word physical address (or a 34-bit byte address). The ATMU translated address[0–32] is a 33-bit double-word physical address (or 36-bit byte address).

The ATMU window hit definition and RapidIO address translation are as follows:

- 4 KB window size (smallest window size):
  - A window hit is defined as {BEXADD[0–1], BADD[0–19]} matching RapidIO address [0–21].
  - Internal interconnection addr[0–32] = {TREXAD[0–3], TRAD[0–19], RapidIO address[22–30]}.
- 8 KB window size:
  - A window hit is defined as {BEXADD[0–1], BADD[0–18]} matching RapidIO address [0–20].
  - Internal interconnection addr[0–32] = {TREXAD[0–3], TRAD[0–18], RapidIO address[21–30]}.
- 16 KB window size:
  - A window hit is defined as {BEXADD[0–1], BADD[0–17]} matching RapidIO address [0–19].
  - Internal interconnection addr[0–32] = {TREXAD[0–3], TRAD[0–17], RapidIO address[20–30]}.
- Window sizes 32 KB, 64 KB, 128 KB, 256 KB, 512 KB, 1 MB, 2 MB, 4 MB, 8 MB, 16 MB, 32 MB, 64 MB, 128 MB, 256 MB, 512 MB, 1GB, 2 GB are not shown.
- 4 GB window size:
  - A window hit is defined as {BEXADD[0–1]} matching RapidIO address [0–1].
  - Internal interconnection addr[0–32] = {TREXAD[0–3], RapidIO address[2–30]}.
- 8 GB window size:
  - A window hit is defined as {BEXADD[0]} matching RapidIO address0.
  - Internal interconnection addr[0–32] = {TREXAD[0–2], RapidIO address[1–30]}.
- 16 GB window size (largest size):
  - A window hit is defined as any RapidIO address.
  - Internal interconnection addr[0–32] = {TREXAD[0–1], RapidIO address[0–30]}.

16.2.5.4.1 Hits to Multiple ATMU Windows

If a request hits multiple ATMU windows, window 1 has the highest priority of the five inbound ATMU windows (windows 1–4, default). Window 2 has the next highest priority, followed by windows 3 and 4. The default window has the lowest priority.

If a request hits (base address match) multiple ATMU windows and the transaction end address is contained within the boundary of each hit window and does not extend into another ATMU window, the translation window is the highest-priority window.

If a lower-priority window is programmed to lie entirely within a higher-priority window, then it is possible for a transaction to cross window boundaries. Although this is not a practical programming application, RapidIO Endpoint handles it as follows:

- If a request hits (base address match) an ATMU window (1–4) and the transaction end address extends into another ATMU window with a lower priority but contained within the boundary of the hit window, the translation window is the hit window.
- If a request hits (base address match) multiple ATMU windows (1–4) and the transaction end address extends beyond the boundary of a lower-priority hit window but is still contained within the boundary of a higher-priority hit window, the translation window is the highest priority window.

16.2.5.4.2 Window Boundary Crossing Errors

If a higher-priority window is programmed to lie entirely within a lower-priority window, then it is possible for a transaction to cross window boundaries. The RapidIO endpoint handles this situation as follows:

- If a request hits (base address match) an ATMU window (1–4, default) and the transaction end address extends into another ATMU window with a higher priority, an ATMU crossed boundary error is generated and logged.
- If a request hits multiple ATMU windows (1–4, default) and the transaction end address extends beyond the boundary of a higher priority hit window, an inbound ATMU crossed boundary error is generated and logged.

Other window boundary crossing errors are as follows:

- If a request hits (base address match) an ATMU window (1–4) and the transaction end address exceeds the size of the window, an inbound ATMU crossed boundary error is generated and logged.
- If a NREAD/NWRITE_R/NWRITE/SWRITE request hits (base address match) an ATMU window (1–4, default) and the transaction end address extends into the region defined as the local configuration space window, an inbound ATMU crossed boundary error is generated and logged.

A RapidIO error response is generated for RapidIO requests that require a response. RapidIO requests that do not require a response are dropped. Inbound ATMU boundary crossing errors are logged in the LTLEDCSR[IACB] configuration register. If a request misses all ATMU windows (1–4) and the transaction end address exceeds the maximum size of the default window, an inbound ATMU crossed boundary error is not generated.

16.2.6 Generating Link-Request/Reset-Device

In LP-Serial mode, the link partner cannot be reliably reset using the link-request/reset-device control symbols because the input port receiver cannot be disabled independently of the output port driver. The input port driver must be disabled to prevent non-idle control symbols from being transmitted between the four link-request/reset-device control symbols. For example, if a packet is received on the inbound side after one of the four link-request/reset-device control symbols is sent outbound, the required sequence of four link-request/reset-device symbols is interrupted with the packet acknowledgement (packet accept, packet retry, or packet not accept).

One method to reset the link partner is as follows.

1. Disable packet reception and transmission by setting the Port Lockout bit (PL bit = 1 in the Port n Control Command and Status Register).
2. Wait the appropriate amount of time for all outstanding packets transmitted on the link to be either retried, accepted or timed-out
3. Discard all outbound packets (OBDEN = 1 in Port n Physical Configuration Register) and clear all errors
4. Disable the input port receiver (IPD bit = 1 in the Port n Control Command and Status Register). This will allow the four consecutive link-request/reset-device control symbols to be generated with only idle control symbols between the link-request/reset-device control symbols.
5. Generate four link-request/reset-device control symbol using the Port n Link Maintenance Request register. Note that the link partner does not generate a link-response control symbol for a link-request/reset-device control symbol.
6. The link partner will expects the inbound and outbound Ack IDs to be 0 after being reset. Set the inbound and outbound Ack IDs to 0 (IA and OBA) by writing 0s to these fields in the Port n Local AckID Status Command and Status Register (PnLASCSR).
7. After the link partner has completed initialization indicated by the PO bit of Port n Error and Status Command and Status Register (PnESCSR), enable packets to be received and transmitted by clearing the Port Lockout bit (PL) in the Port n Control Command and Status Register (PnCCSR).

16.2.7 Outbound Drain Mode

The RapidIO port is placed into Drain mode when one of the following occurs:

- PnPCR[OBDEN] is set.
- he Failed Threshold has been encountered and the PnCCSR[SPF] and PnCCSR[DPE] are both set
- The packet time-to-live counter expires causing PnPCR[OBDEN] to be set

When the RapidIO port is placed into Drain mode, the RapidIO port discards all packets in the outgoing data stream. Since the data stream is invalid, the RapidIO port also puts its outbound port back to normal state. Any received acknowledgements and link-responses are considered invalid during this period (because the RapidIO port has cleared out all acknowledgement history).

The RapidIO port's outbound and outstanding ackID shows that all outstanding packets at the time Drain mode was entered were accepted, whether they were accepted or not. If the outbound ackID is not acceptable, then software should change it prior to taking the RapidIO port out of Drain mode. Also, if the link-partner needs to be returned to the inbound OK state, software should send a link-request/input-status. The recommended sequence for recovering from Drain mode is given in **Section 16.2.9**, *Software Assisted Error Recovery Register Support,* on page 16-25.

Setting PnPCR[OBDEN] also causes any queued packet acknowledgements to be discarded if the port is uninitialized; the RapidIO port allows them be transferred if the port is initialized. Drain mode due to Failed Threshold does not cause any packet acknowledgements to be dropped.

If a discarded packet in the outgoing data stream requires a logical response, a packet response time-out will occur if the packet response timer is enabled (PRTOCCSR is non 0).

16.2.8 Input Port Disable Mode

When PnCCSR[PD] is set, the RapidIO port is placed into Input Port Disable mode. This mode causes the following to occur:

- The RapidIO port discards all incoming data streams.
- Because the incoming stream is invalid, the RapidIO port returns it inbound port to the normal state. If an incoming packet is in progress when the RapidIO port is placed into Input Port Disable mode, the RapidIO port physical layer aborts the packet to the logical layer.
- The RapidIO port ends any packet capture that is in progress when the mode is invoked.
- The RapidIO port clears the link-request/reset-device count.
- The RapidIO port inbound ackID shows that all packets successfully received by the port before it entered Input Port Disable mode were accepted. If the outbound port entered Output Port Disable mode at the same time, however, some packet acknowledgements may be discarded by the RapidIO port. If the inbound ackID is not accepted, software should change it before taking the RapidIO port out of Input Port Disable mode.

16.2.9 Software Assisted Error Recovery Register Support

PnLMREQCSR is only supported for recovery from Drain mode. Therefore, software should only write to this register when the port is in Drain mode. The proper sequence for recovering from Drain mode is as follows:

1. Software ensures that link activity is stopped. This should include:
   a. Software polls for Port OK bit to be set
   b. Software waits longer than the link time-out value
2. Software generates a link-request/input-status to obtain the link-partner's inbound ackID value.
3. Software changes the RapidIO port's outbound ackID to this value (if necessary).
4. If the link-partner was hot-inserted, software changes the RapidIO port's inbound ackID to zero.

Note: If software can guarantee that the link-partner will not attempt to forward any packets to this RapidIO port, then software can write the outbound ackID of the link-partner to match the inbound ackID of this RapidIO port. However, if the link-partner attempts to forward another packet while this write is still outstanding, and the ackIDs is already lined up, then the write actually causes the ackIDs not to match, and the link is not recovered.

5. Software should cause the link-partner to send a link-request/input-status to ensure that the RapidIO inbound port is operating normally.
6. Software clears the Failed Encountered bit or the Output Buffer Drain Enable bit; whichever one caused the Drain mode (thus clearing Drain mode).

Software is responsible for timing software generated link-requests. If the response valid bit is not set in some reasonable period of time, the software should write another request in the register.

When software writes PnLMREQCSR, software should make sure that PnLMRESPCSR successfully reads as set; otherwise, software may read a stale ackID status/link status later.

Note: When the RapidIO port's outbound ackID is written by software using PnLASCSR, the inbound ackID is also written. Care must be taken to ensure that the inbound ackID is not written to an incorrect value. For example, PnCCSR[PL] could be set to prevent the inbound ackID from changing before PnLASCSR is written.

16.2.10 Errors and Error Handling

This section describes how the logical and physical layers detect RapidIO errors and respond to them. For details on the action of the SC3400 core when it is notified of any of these errors, see

the *RapidIO Interconnect Specification, Revision 1.2*, part VII (Error Management Extensions Specifications).

16.2.10.1 RapidIO Error Description

RapidIO errors are classified under three categories: recoverable errors, notification errors, and fatal errors.

- *Recoverable errors*. Non-fatal transmission errors, such as a corrupt packet or corrupt control symbols and general protocol errors, for which there is hardware detection and recovery as described in the *RapidIO Interconnect Specification, Revision 1.2*. In these cases, the appropriate bit is set in the Port 0 Error Detect CSR. Only the packet containing the first detected recoverable error that is enabled for error capture (by the Port 0 Error Enable CSR) is captured in the Port 0 Error Capture CSRs. No interrupt is generated or actions required for a recoverable error. Recoverable errors are detected only in the physical layer.
- *Notification errors*. Non-recoverable non-fatal errors detected by RapidIO, such as degraded threshold, port-write received, and all logical/transport layer (LTL) errors captured. Because they are non-recoverable and in some cases have caused a packet to be dropped, notification by interrupt is available. However, because they are non-fatal, a response to the interrupt is not crucial to port performance. The port is still functional. When a notification error is detected, the appropriate bit is set in the error-specific register, an interrupt is generated, and in some cases, the error is captured. In all cases, the RapidIO port continues operating. Notification errors are detected in both the physical and logical layers.
- *Fatal errors*. There are two types of fatal errors:
  - *Exceeded failed threshold*. The port fails because its recoverable error rate has exceeded a predefined failed threshold. RapidIO sets the Output Failed-encountered bit in the Port0 Error and Status CSR; the RapidIO output hardware may or may not stop (based on Stop-on-Port-Failed-Encounter-Enable and Drop-Packet-Enable bits).
  - *Exceeded consecutive retry*. The port fails because it has received too many packet retries in a row. The RapidIO controller sets the Retry Counter Threshold Trigger Exceeded bit in the Port 0 Implementation Error CSR; the RapidIO hardware continues to operate.

  In both cases, an interrupt is generated, and while the port continues operating at least partially, a system-level fix (such as reset) is recommended to clean up the RapidIO controller internal queues and resume normal operation. Fatal errors are detected only in the physical layer.

16.2.10.2 Physical Layer RapidIO Errors

Table 16-8 lists all the RapidIO link errors detected by the RapidIO endpoint physical layer and the actions taken by the RapidIO endpoint. The Error Enable column lists the control bits that can

disable the error checking associated with a particular error. If this column is blank, error checking cannot be disabled. The Cause Field column indicates which cause field is used with the associated packet-not-accept control symbol for input error recovery. The EME Error Enable/Detect column indicates which bit of the P0ERECSR (see **page 16-137**) allows the error to increment the error rate counter and lock the Port 0 Error Capture registers—and also which P0EDCSR bit is set when the error is detected (see **page 16-136**).

Table 16-9 shows the RapidIO endpoint behavior after certain preset limits are exceeded (degraded threshold, failed threshold, retry threshold). **Table 16-10** shows the threshold response.

Table 16-8. Physical RapidIO Errors Detected

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable/ Detect
Recoverable Errors						
1a	Received character has a disparity error.		Enter input error stopped. Enter output error stopped.	5: Received invalid/illegal character	Delineation Error	DE
1a	Received an invalid character or valid but illegal character.		Enter input error stopped. Enter output error stopped.	5: Received invalid/illegal character		
1b	The four control character bits associated with the received symbol do not make sense (not 0000, 1000, 1111).		Enter input error stopped. Enter output error stopped.	5: Received invalid/illegal character		
1b	Control symbol does not begin with an /SC/ or /PD/ control character.		Enter input error stopped. Enter output error stopped.	5: Received invalid/illegal character		
1c	Received a packet with embedded idles.		Enter input error stopped.	5: Received invalid/illegal character		
1d	Received a control symbol with a bad CRC.	P0PCR[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.	P0PCR[CCP] enables detect.	Enter input error stopped.	4: bad CRC on packet.	Received packet with bad CRC	CRC

Table 16-8. Physical RapidIO Errors Detected (Continued)

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable/ Detect
2a	Received packet which exceeds the maximum allowed size by the RapidIO spec.		Enter input error stopped.	7/31: General error	Received packet exceeds 276 Bytes	EM
2b	Received packet with unexpected ackID value (out-of-sequence ACKID)		Enter input error stopped.	1: Received unexpected ACKID on packet	Received packet with unexpected ackID	UA
2c	Received a non-maintenance packet when non-maintenance packet reception is stopped	Non-maint. packet reception stops when Input Port Enable = 0.	Enter input error stopped.	3: Non-maintenance packet reception stops	Not Captured	
2d	Any packet received while Port Lockout bit is set	All packet reception stops when Port Lockout bit is set.	Enter input error stopped.	3: Non-maintenance packet reception stops	Not Captured	
—	Received a Link request control symbol before servicing previous link request.	Not detected.				
2a	Received packet-not-accepted ACK control symbol.		Enter output error stopped.		Received packet-not-accepted symbol	PNA
2b	Received an ACK (accepted, or retry) control symbol when there are no outstanding packets		Enter output error stopped.		Unsolicited ACK symbol	UCS
2b	Received packet ACK (accepted) for a packet whose transmission has not finished		Enter output error stopped.			
2b	Received a Link response control symbol when no outstanding request.		Enter output error stopped.			
2c	Received an ACK (accepted or retry) control symbol with an unexpected ACKID.		Enter output error stopped.		Received ack. control symbol with unexpected ackID	AUA
2c	Link_response received with an ackID that is not outstanding		Re-enter Output Error Stopped.		Non-outstanding ackID	NOA
2d	An ACK control symbol is not received within the specified time-out interval.	PLTOCCSR [TV] > 0 enables detect.	Enter output error stopped.		Link time-out	LTO
2d	A Link response is not received within the specified time-out interval	PLTOCCSR [TV] > 0 enables detect.	(re-) Enter output error stopped.			

Table 16-9. Physical RapidIO Threshold Response

Error	Error Enable	RapidIO Endpoint Action	EME Error Type	Error Detect	Interrupt Clear*
Notification Errors					
Error rate counter exceeded the degraded threshold.	P0ERTCSR[ERDTT] > 0 and any bit in P0EECSR enables detect and interrupt generation.	Generate interrupt. Continue to operate normally.	Degraded threshold	P0ESCSR[ODE]	Write 1 to P0ESCSR [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	P0IECSR[RETE]	Write 1 to P0IECSR [RETE]
Error rate counter exceeded the failed threshold.	P0ERTCSR[ERFTT] > 0 and any bit in P0EECSR enables detect and interrupt generation.	Generate Interrupt. Port behavior depends on P0CCSR[SPF] and P0CCSR[DPE]. Port can continue transmitting packets or stop sending output packets, keeping or dropping them.	Failed threshold	P0ESCSR[OFE]	Write 1 to P0ESCSR [OFE]
Note: Information given here is minimal for clearing the interrupt. More detailed steps should be taken to find the cause of the interrupt, as described in the interrupt generation reference of the *RapidIO Interconnect Specification, Revision 1.2* Part VII (Error Management Extensions Specifications).					

16.2.10.3 Logical Layer RapidIO Errors

This section describes how the logical layer detects and responds to RapidIO errors. The action of the core processor when it is notified of these errors is minimally described. For details, see the interrupt generation reference in the *RapidIO Interconnect Specification, Revision 1.2,* part VII (Error Management Extensions Specifications).

Table 16-10 through **Table 16-22** list all the errors detected by the RapidIO endpoint logical layer and the actions taken. Error responses are sent as follows:

- When the RapidIO endpoint action includes sending an error response to the system or the RapidIO interconnect, an error response is sent only if the original transaction is a request requiring a response. Otherwise, no error response is sent.
- For multiple errors, a discard of a packet has a higher priority than an error response.
- For misaligned transactions, the error management extension registers are updated with each child.

Table 16-10. Hardware Errors For NREAD Transaction

Error	Interrupt	Status Bit Set	Error Response	Comments
Priority Priority of read transaction is 3.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	RapidIO packet is dropped.
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
Received TT that is not enabled. Error valid is when passthrough is disabled and accept_all is disabled or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
DestID DestID does not match this port DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (pass_through | accept_all) is false.	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes	
SourceID Not Checked for error.				
TransactionType Received RapidIO packet with reserved TType for this ftype.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	
RdSize Not checked for error.				
SrcTID Not checked for error.				
Address:WdPtr:Xambs Read request hits overlapping ATMU windows Refer to **Section 16.2.5.4.2**, *Window Boundary Crossing Errors,* on page 16-22.	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 16-10. 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 16-133** 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. • LTLTLTLDIDCCSR[DIDMSB] gets packet bits 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s.				

Table 16-11. Hardware Errors For Maintenance READ/WRITE Request Transaction

Error	Interrupt	Status Bit Set	Error Response	Comments
Priority Priority of maintenance read or write request transaction is 3.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	RapidIO packet is dropped.
TransportType Received reserved TT.	Yes	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
Received TT that is not enabled. Error is valid when passthrough is disabled and accept_all is disabled or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
DestID DestID does not match this port DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough | accept_all) is false.	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes	
SourceID Not Checked for error.				
TransactionType Reserved transaction type for this ftype	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	Yes	RapidIO packet is dropped.
RdSize Read/Write request size is not for 4 bytes.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	Yes	
SrcTID Not checked for error.				

Table 16-11. 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 16-133** 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. • LTLTLTLDIDCCSR[DIDMSB] gets packet bits 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s.				

Table 16-12. Hardware Errors For NWRITE, NWRITE_R

Error	Interrupt	Status Bit Set	Error Response	Comments
Priority Not UT	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped.
TransportType Received reserved TT.	Yes.	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
Received TT that is not enabled. Error is valid when passthrough is disabled and accept_all is disabled or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
DestID DestID does not match this port DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error is valid when (passthrough | accept_all) is false.	Yes if LTLEECSR[ITTE] is set.	LTLEDCSR[ITTE]	Yes for NWRITE_R. No for NWRITE.	RapidIO packet is dropped for NWRITE.
SourceID Not applicable.				
TransactionType	Yes if LTLEECSR[UT] is set.	LTLEDCSR[UT]	Yes	
TransactionType Received RapidIO packet with reserved TType for this ftype. Packet is treated as Nwrite Transaction.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	Yes for NWRITE_R. No for NWRITE	RapidIO packet is dropped.
WrSize Not unsupported transaction WrSize request is for one of reserved sizes.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	Yes for NWRITE_R. No for NWRITE.	RapidIO packet is dropped for NWRITE.
Address:WdPtr:Xambs Not unsupported transaction NWRITE request hits LCSBA1CSR.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No for NWRITE.	RapidIO packet is dropped for NWRITE.
Address:WdPtr:Xambs Not unsupported transaction Request hits a protected ATMU window or the local configuration space window.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	Yes for NWRITE_R. No for NWRITE.	RapidIO packet is dropped for NWRITE.
Address:WdPtr:Xambs Write request hits overlapping ATMU windows. Refer to **Section 16.2.5.4.2**, *Window Boundary Crossing Errors,* on page 16-22.	Yes if LTLEECSR[IACB] is set.	LTLEDCSR[IACB]	Yes for NWRITE_R. No for NWRITE.	RapidIO packet is dropped for NWRITE.
SrcTID Not Checked for error.				
Address:WdPtr:Xambs NWRITE_R address matches LCSBA1CSR with a request that is not a 32-bit read. Performed only for NWRITE_R packet.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	Yes for NWRITE_R.	

Table 16-12. Hardware Errors For NWRITE, NWRITE_R (Continued)

Error	Interrupt	Status Bit Set	Error Response	Comments
Header Size Not unsupported transaction. Header size is less than 12 bytes for small transport packet or less than 16 bytes for large transport packet; that is, no payload is present. Large transport packet has 14 valid bytes and two bytes of padding of 0s. Padding of 0s is not checked.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	Yes for NWRITE_R. No for NWRITE.	RapidIO packet is dropped for NWRITE.
PayloadSize Not unsupported transaction. Payload is greater than that indicated by the {wdptr:wrsize} field. Payload is not double word aligned or does not have any payload.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	Yes for NWRITE_R. No for NWRITE.	RapidIO packet is dropped for NWRITE.
The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a small transport packet as follows: • LTLACCSR[XA] gets packet bits 78–79. • LTLACCSR[A] gets packet bits 48–76. • LTLDIDCCSR[DIDMSB] gets 0s. • LTLDIDCCSR[DID] gets packet bits 16–23. • LTLDIDCCSR[SIDMSB] gets 0s. • LTLDIDCCSR[SID] gets packet bits 24–31. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 32–35. • LTLCCCSR[MI] gets 0s. The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a large transport packet as follows: • LTLACCSR[XA] gets packet bits 94–95. • LTLACCSR[A] gets packet bits 64–92. • LTLDIDCCSR[DIDMSB] gets packet bits 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets packet bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s.				

Table 16-13. Hardware Errors For SWRITE Transactions

Error	Interrupt	Status Bit Set	Error Response	Comments
Priority Swrite transaction priority is 3.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped.
TransportType Received reserved TT.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
Received TT that is not enabled. Error is valid when passthrough is disabled and accept_all is disabled or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
DestID DestID does not match this port DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error is valid when (passthrough | accept_all) is false.	Yes if LTLEECSR[ITTE] is set.	LTLEDCSR[ITTE]	No	RapidIO packet is dropped.
SourceID Not checked for error.				
Address:WdPtr:Xambs SWRITE request hits overlapping ATMU windows. See **Section 16.2.5.4.2**, *Window Boundary Crossing Errors,* on page 16-22.	Yes if LTLEECSR[IACB] is set.	LTLEDCSR[IACB]	No	RapidIO packet is dropped.
Address:WdPtr:Xambs Request hits a protected ATMU window or the local configuration space window.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped.

Table 16-13. Hardware Errors For SWRITE Transactions (Continued)

Error	Interrupt	Status Bit Set	Error Response	Comments
PayloadSize Payload size is not in DWs, has exceeded 256 bytes, or has no payload.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped.
The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a small transport packet as follows: • LTLACCSR[XA] gets packet bits 62–63. • LTLACCSR[A] gets packet bits 32–60. • LTLDIDCCSR[DIDMSB] gets 0s. • LTLDIDCCSR[DID] gets packet bits 16–23. • LTLDIDCCSR[SIDMSB] gets 0s. • LTLDIDCCSR[SID] gets packet bits 24–31. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 32–35. • LTLCCCSR[MI] gets 0s. The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a large transport packet as follows: • LTLACCSR[XA] gets packet bits 78–79. • LTLACCSR[A] gets packet bits 46–76. • LTLDIDCCSR[DIDMSB] gets packet bits 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets packet bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s.				

Table 16-14. Hardware Errors For Maintenance Response Transactions

Error	Interrupt	Status Bit Set	Error Response	Comments
Priority Not unsupported response. Response priority is not higher than the RapidIO maintenance priority.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped and ignored.
TransportType Received reserved TT.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No	RapidIO packet is dropped and ignored.
Received TT that is not enabled. Error is valid when passthrough is disabled and accept_all is disabled or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No	RapidIO packet is dropped and ignored.
DestID DestID does not match this port DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error is valid when (passthrough | accept_all) is false.	Yes if LTLEECSR[ITTE] is set.	LTLEDCSR[ITTE]	No	RapidIO packet is dropped and ignored.
SourceID Does not match the request DestID.	Yes if LTLEECSR[UR] is set.	LTLEDCSR[UR]	No	RapidIO packet is dropped and ignored.
Transaction Type. Not unsupported transaction. Received RapidIO packet with reserved TType for the FType.	Yes if LTLEECSR[TD] is set.	LTLEDCSR[TD]	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[TD] is set.	LTLEDCSR[TD]	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[TD] is set.	LTLEDCSR[TD]	No	RapidIO packet is dropped and ignored.
Status Not unsupported response Error Response.	Yes if LTLEECSR[IER] is set.	LTLEDCSR[IER]	Yes	OCN error response is generated to requestor
TargetTID No outstanding transaction for this TargetTID	Yes if LTLEECSR[UR] is set.	LTLEDCSR[UR]	No	RapidIO packet is dropped and ignored.
Header Size Not unsupported response Maintenance Read response—total payload size with “Done” status is not greater than 4 bytes. Maintenance Write response—total header size is less than 12 bytes for small transport packet or is less than 16 bytes for large transport packet. Padding of 0s for small or large transport packet is not verified.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[TID]	No	RapidIO packet is dropped and ignored.

Table 16-14. Hardware Errors For Maintenance Response Transactions (Continued)

Error	Interrupt	Status Bit Set	Error Response	Comments
PayloadSize Not unsupported response Maintenance writeresponse—has payload. Maintenance read response—with “Done” status and payload not matching valid request size, request size for the response is invalid, or payload size is not 64-bit aligned.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped and ignored.
Packet response time-out. Response is not received by configured time.	Yes if LTLEECSR[PRT] is set.	LTLEDCSR[PRT]	Yes	OCN response is generated to requestor.
The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a small transport packet as follows: • LTLACCSR[XA] gets packet bits 78–79. • LTLACCSR[A] gets packet bits 48–76. • LTLDIDCCSR[DIDMSB] gets 0s. • LTLDIDCCSR[DID] gets packet bits 16–23. • LTLDIDCCSR[SIDMSB] gets 0s. • LTLDIDCCSR[SID] gets packet bits 24–31. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 32–35. • LTLCCCSR[MI] gets 0s. The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a large transport packet as follows: • LTLACCSR[XA] gets packet bits 94–95. • LTLACCSR[A] gets packet bits 64–92. • 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 0s.				

Table 16-15. 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 16-15. 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. • LTLTLTLDIDCCSR[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. • LTLTLTLDIDCCSR[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 16-16. 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. • LTLTLTLDIDCCSR[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. • LTLTLTLDIDCCSR[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 16-17. 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.
- LTLTLTLDIDCCSR[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.
- LTLTLTLDIDCCSR[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 16-18. 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 16-18. 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. • LTLTLTLDIDCCSR[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. • LTLTLTLDIDCCSR[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 16-19. 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 16-19. 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 16-19**, 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. • LTLTLTLDIDCCSR[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 16-19**, 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. • LTLTLTLDIDCCSR[DIDMSB] gets packet bit 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets packet bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s.				

Table 16-20. Hardware Errors for 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[OACB] 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. • LTLTLTLDIDCCSR[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. • LTLTLTLDIDCCSR[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 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. • LTLTLTLDIDCCSR[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. • LTLTLTLDIDCCSR[DIDMSB] gets packet bits 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets packet bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s.				

Table 16-22. Hardware Errors for 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[IUT] is set.	LTLEDCSR[UT]	No	RapidIO packet is dropped.
TransportType Received reserved TT.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
Received TT that is not enabled. Error is valid when passthrough is disabled and accept_all is disabled or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough | accept_all) is false.	Yes if LTLEECSR[ITTE] is set.	LTLEDCSR[ITTE]	No	RapidIO packet is dropped.
Address:WdPtr:Xambs Swrite request hits overlapping ATMU windows. Refer to **Section 16.2.5.4.2**, *Window Boundary Crossing Errors,* on page 16-22. 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. • LTLTLTLDIDCCSR[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. • LTLTLTLDIDCCSR[DIDMSB] gets packet bits 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets packet bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s. • LTLCCCSR[MI] gets 0s.				

16.3 RapidIO Message Unit

The RapidIO message unit handles multicast and non-multicast single-segment messages and non-multicast multiple-segment messages. The RapidIO controller has two inbound and two outbound message controllers.

The message passing programming model for inter-processor and inter-device communication enables a producer to send a message across the interconnect fabric to the message hardware of a consumer, called a mailbox. The receiving mailbox hardware places the message into a queue located in local memory. A message consists of one to sixteen segments. When a configured number of messages is received, an interrupt is generated (if enabled) to the interrupt controller for the processor to process the messages. Messages can be queued for transmission in the producer memory, and the message hardware processes them sequentially. Messages can also be queued in consumer memory while software processes them sequentially. Software can configure the depths of the queues in the producer and consumer memories. A multicast function allows single-segment messages to be sent to multiple consumers.

The message unit is compliant with the message passing logical specification in the *RapidIO Interconnect Specification, Revision 1.2*. The message passing model is most commonly used in systems in which a processing element is allowed to access only memory that is local to itself, processing elements communicate with each other through message passing, and communication is address independent.

Each inbound message controller has a dedicated interrupt to notify software that a configured number of messages have been received. Similarly, each outbound message controller has a dedicated interrupt to notify software that there are no more messages for the outbound mailbox controller. The message controller is managed through a set of run-time registers.

16.3.1 Features

The message unit supports two outbound message controllers with the following features:

- Chaining and direct modes.
- Multicast up to 32 RapidIO destinations for single-segment messages.
- Transmission to any mailbox for a single-segment message.
- Transmission to any mailbox for a multi-segment message.
- Segment size up to 256 bytes.
- Up to 16 segment messages with a total payload of up to 4 KB.
- One entire message with up to a 16 message segments can be transmitted before a response is received.
- One entire single-segment message to all multicast destinations (up to 32) can be transmitted before a response is received.

- All message segment transfers for a message transaction must complete before the next message transaction begins.
- Pipelined transmission of a full message in each message controller but all responses must be received before the next message can be transmitted.
- In Chaining mode, the next descriptor can be fetched before the current message completes (descriptor prefetching).

The message unit supports two inbound message controllers with the following features:

- Reception of any mailbox and letter for a single- or multi-segment message.
- Segment size up to 256 bytes.
- Up to sixteen segment messages with a total payload of up to 4 KB.
- Full inbound line rate performance.
- Back-to-back message reception of the same or lower priority.
- Out-of-order message segment reception.
- Concurrent inbound message controller operation.

16.3.2 Outbound Message Controller Operation

The outbound message controller sends messages stored in local memory, and it can operate in three different modes:

- *Direct mode*. Software programs the necessary registers to point to the beginning of the message in memory.
- *Chaining mode*. Software programs the necessary registers to point to the beginning of the first valid descriptor in memory. The descriptor provides all the necessary registers to start the message transfer.
- *Multicast mode*. A single-segment message can be sent to multiple destinations. Multicast mode is supported in Direct or Chaining mode.

Each outbound message controller uses a unique identifying number. For example, if there are two outbound message controllers, message controller 0 uses number 0 and message controller 1 uses number 1.

16.3.2.1 Direct Mode

In Direct mode (OM*x*MR[MUTM] = 1; see **page 16-167**) the outbound message controller does not read descriptors from memory. Instead, it uses the parameters programmed in the outbound message controller registers to start the transfer. Software initializes all the parameters to start the message transmission. The message transfer starts when the outbound message controller start bit, OM*x*MR[MUS] changes from 0 to 1 and the outbound message controller is not busy. If it is busy, OM*x*MR[MUS] the transition from 0 to 1 is ignored. Software should program all the appropriate registers before setting OM*x*MR[MUS].

There are many ways in which software can interact with the message controller. One example sequence of events to start and complete a transfer in Direct mode is as follows:

1. Poll the OM*x*SR[MUB] bit to ensure that the outbound message controller is not busy.
2. Clear the following OMxSR status bits (see **Table 16-104,** *OMxSR Field Descriptions,* on page 16-170):
   - MER
   - RETE
   - PRT]
   - TE
   - QOI
   - QFI
   - EOMI
   - QEI
3. Initialize the following registers:
   - Source address (OM*x*SAR)
   - Destination port (OM*x*DPR)
   - Destination attributes (OM*x*DATR)
   - Retry error threshold (OM*x*RETCR)
   - Double-word count (OM*x*DCR).

   If multicast mode is enabled (OM*x*MR[MM]), initialize the multicast group and list in OM*x*MGR and OM*x*MLR.
4. Initialize the outbound message mode register message unit transfer mode bit, OM*x*MR[MUTM] = 1, to indicate direct mode. Other control parameters must also be initialized in the mode register.
5. Clear and then set the mode register message unit start bit, OM*x*MR[MUS], to start the message transfer.
6. The outbound message controller sets the OM*x*SR[MUB] bit to indicate that the message transfer is in progress.
7. The outbound message controller reads a message segment from local memory using the source address register (OM*x*SAR).
8. If a message has multiple segments, the outbound message controller reads the other message segments from local memory.
9. After the message read to local memory completes, the message is sent.

10. The outbound message controller clears OM*x*SR[MUB]. A non-multicast message transfer completes after all message segments complete. A multicast message transfer completes after all message segments complete for each destination. A message segment completes when one of the following occurs:
    - Done response received
    - Error response
    - Packet response time-out received
    - Retry error threshold exceeded
    - An internal error occurs during the local memory access
11. After the outbound message operation completes, the outbound message interrupt is generated if the end of message outbound message interrupt event is enabled (OMnDATR[EOMIE]).

    In Direct mode, the outbound message interrupt is generated after a message operation completes if OM*x*DATR[EOMIE] =1.

    The event causing this interrupt is indicated by OM*x*SR[EOMI]. The interrupt is held until the OM*x*SR[EOMI] bit is cleared by writing a 1 to it.

    In Direct mode, the Error/Port-Write interrupt is generated for the following reasons:
    - A message error response is received and this interrupt event is enabled (OM*x*MR[EIE]).
    - A packet response time-out occurs and this interrupt event is enabled (OM*x*MR[EIE]).
    - A retry threshold exceeded error occurs and this interrupt event is enabled (OM*x*MR[EIE]).
    - An internal error response is received and this interrupt event is enabled (OM*x*MR[EIE]).

Table 16-23 describes each of these error types.

Table 16-23. Error Types In Outbound Message Controller Direct Mode

Error Type	Message Controller Response to Error
Message Error Response	• Sets the message error response status bit (OM*x*SR[MER]). • Generates a serial RapidIO error/write-port interrupt if OM*x*MR[EIE] is set. • Stops after the message operation completes (indicated by OM*x*SR[MUB]).
Packet Response Time-Out	• Sets the packet response time-out status bit (OM*x*SR[PRT]). • Generates a serial RapidIO error/write-port interrupt if OM*x*MR[EIE] is set. • Stops after the message operation completes (indicated by OM*x*SR[MUB]).
Retry Error Threshold Exceeded	• Sets the retry threshold exceed status bit (OM*x*SR[RETE]). • Generates a Serial RapidIO error/write-port interrupt if OM*x*MR[EIE] is set. • Stops after the message operation completes (indicated by OM*x*SR[MUB]).
Internal Error During Local Memory Read	• Sets the transaction error bit (OM*x*SR[TE]) • Does not send message segments with an internal error because the message data is not available • Does not transfer memory reads generated before the internal error. • Generates but does not transfer additional memory reads for the same message operation. • Does not transfer all subsequent message segments for the same message operation, including retried message segments. • Stops after the message operation completes (indicated by OM*x*SR[MUB]). • Generates the Serial RapidIO error/write-port interrupt if OM*x*MR[EIE] is set.

16.3.2.2 Software Error Handling

When an error occurs and the Serial RapidIO error/write-port interrupt is generated, software takes the following actions:

1. Determines the cause of the interrupt and processes the error.
2. Verifies that the message controller has stopped operation by polling OM*x*SR[MUB].
3. Disables the message controller by clearing OM*x*MR[MUS].
4. Clears the error by writing a 1 to the corresponding OMxSR status bit (see **Table 16-104,** *OMxSR Field Descriptions*, on page 16-170):
   - MER
   - PRT
   - RETE
   - TE

When an error occurs and the Serial RapidIO error/write-port interrupt is not enabled, software takes the following actions:

1. Determines that an error has occurred by polling the status bits OMxSR status bit (see **Table 16-104,** *OMxSR Field Descriptions*, on page 16-170):
   - MER
   - PRT
   - RETE
   - TE
2. Verifies that the message controller has stopped operation by polling OM*x*SR[MUB].
3. Disables the message controller by clearing OM*x*MR[MUS].
4. Clears the error by writing a 1 to the corresponding status bit (listed in step 1).

16.3.2.3 Disabling and Enabling the Message Controller

Once the message controller is started, it cannot be stopped except by loss of power or reset.

16.3.2.4 Hardware Error Handling

Table 16-24 describes Direct mode hardware errors. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. When an error is detected, no additional error checking beyond the current level is performed. The first error detected in the processing pipeline updates the error management extensions registers. These error condition checks are provided by the messaging unit in addition to the error condition checks provided by the RapidIO port described in **Section 16.2.10**, *Errors and Error Handling*, on page 16-25.

Table 16-24. Outbound Message Direct Mode Hardware Errors

Transaction	Error	Description
Message request	Internal error during a read of the message segment from local memory	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if OM*x*MR[EIE] set **Status bit set**: Transaction error in the outbound message status register (OM*x*SR[TE]). Message failed in the mailbox CSR (MCSR[FA]). **Message segment sent:** No **Logical/Transport Layer Capture Register:** **Comments:** Message controller stops after the current message operation completes. The descriptor dequeue pointer is not incremented in chaining mode.
Message request	Internal error for an earlier message segment local memory read. An internal error for a subsequent message segment local memory read for the same message may or may not occur.	**Error checking level:** 2 **Interrupt generated:** **Status bit set**: None **Message segment sent:** Yes **Logical/Transport Layer Capture Register:**
Undefined packet	Reserved ftype encoding[1]	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT] **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	Reserved tt encoding[1]	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set **Status bit set**: Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]. Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	Large transport size when operating in small transport size or small transport size when operating in large transport size[1]	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Message response	Illegal destination ID[1]	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITTE] is set **Status bit set**: Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	ttype (transaction field) is not message response[1]	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.

Table 16-24. Outbound Message Direct Mode Hardware Errors

Transaction	Error	Description
Message response	Message response received and no outbound mailboxes are supported[1]	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UR] is set **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UR]. **Message segment sent:** No **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	Reserved response status (not done, retry, or error)	**Error checking level:** 4a **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set. **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	Message response packet size is incorrect	**Error checking level:** 4a **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set. **Status bit set**: Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	Incorrect source ID	**Error checking level:** 4b **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UR] is set. **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	Letter, mbox and msgseg not outstanding or letter, mbox not outstanding	**Error checking level:** 4b **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UR] is set. **Status bit set**: Unsolicited response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UR]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	RapidIO priority is less than or equal to message request	**Error checking level:** 4c **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set. **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	Error response	**Error checking level:** 5 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MER] set. Serial RapidIO error/write-port if OM*x*MR[EIE] is set. **Status bit set**: Message error response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MER]. OM*x*SR[MER] bit is set in Direct mode or Chaining mode. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the corresponding message request packet.[2] **Comments:** Message segment transfer complete. The descriptor dequeue pointer is not incremented in chaining mode.

Table 16-24. Outbound Message Direct Mode Hardware Errors

Transaction	Error	Description
Message response	Number of retries exceeds limit	**Error checking level:** 5 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[RETE] set. Serial RapidIO error/write-port if OM*x*MR[EIE] is set. **Status bit set**: Retry error threshold exceeded in the Logical/Transport Layer Error Detect CSR LTLEDCSR[RETE]. OM*x*SR[RETE] bit is set in Direct mode or Chaining mode. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the corresponding message request packet.[2] **Comments:** Message segment transfer complete. The descriptor dequeue pointer is not incremented in chaining mode.
Message response	Packet response time-out	**Error checking level:** Unrelated **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[PRT] set. Serial RapidIO error/write-port if OM*x*MR[EIE]. **Status bit set**: Packet response time-out in the Logical/Transport Layer Error Detect CSR LTLEDCSR[PRT]. OM*x*SR[PRT] bit is set in Direct mode or Chaining mode. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the corresponding message request packet.[2] The LTLDIDCCSR[SIDMSB] and LTLDIDCCSR[SID] field has a value of 0. **Comments:** Message segment transfer complete. The descriptor dequeue pointer is not incremented in chaining mode.
Notes: 1.	**Notes: 1.** These error types are actually detected in the RapidIO port, not in the message controller. **2.** In small transport size configuration using the packet, the following allocations are made: • LTLACCSR[XA] gets the extended address (packet bits 78–79). • LTLACCSR[A] gets the address (packet bits 48–76). • LTLDIDCCSR[MDID] gets 0. • LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23). • LTLDIDCCSR[MSID] gets 0. • LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31). • LTLCCCSR[FT] gets the ftype (packet bits 12–15). • LTLCCCSR[TT] gets the ttype (packet bits 32–35). • LTLCCCSR[MI] gets the msg info (packet bits 40–47) In large transport size configuration using the packet, the following allocations are made: • LTLACCSR[XA] gets the extended address (packet bits 94–95). • LTLACCSR[A] gets the address (packet bits 64–92). • LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23). • LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31). • LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32–39). • LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47). • LTLCCCSR[FT] gets the ftype (packet bits 12–15). • LTLCCCSR[TT] gets the ttype (packet bits 48–51) if the message request packet is captured or 0 if the message response packet is captured. • LTLCCCSR[MI] gets the message information (packet bits 56–63).

Table 16-25 lists programming errors that result in undefined or undesired hardware operation.

Table 16-25. Outbound Message Direct Mode Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Double word count greater than 256 bytes when multi-cast mode selected	No	None	Undefined operation
Double word count set to a reserved value	No	None	Undefined operation
Transaction flow level set to 3	No	None	Undefined operation
Target interface set to an invalid RapidIO port	No	None	Undefined operation
Source address for message read is invalid	No	No	Local memory captures the transaction and generates an interrupt.
Register values changed during operation	No	No	Undefined operation.

16.3.2.5 Chaining Mode

In Chaining mode, OM*x*MR[MUTM] = 0, message descriptors are built in local memory in a circular queue. Several options are available to the programmer. Software can build one or more descriptors before initializing the outbound message controller registers, or it can initialize the outbound message controller registers and then build the descriptors. Software maintains the enqueue pointer (OM*x*DQEPAR). The outbound message controller dequeues descriptors, processes them, and increments the dequeue pointer (OM*x*DQDPAR) to point to the next descriptor in the queue. **Figure 16-9** depicts a sample structure of the outbound portion of the message controller and a descriptor queue, with each valid descriptor queue entry pointing to a valid message. The descriptor queue has eight entries, four of which are valid. The local processor enqueues descriptors and the outbound message controller dequeues the descriptors.

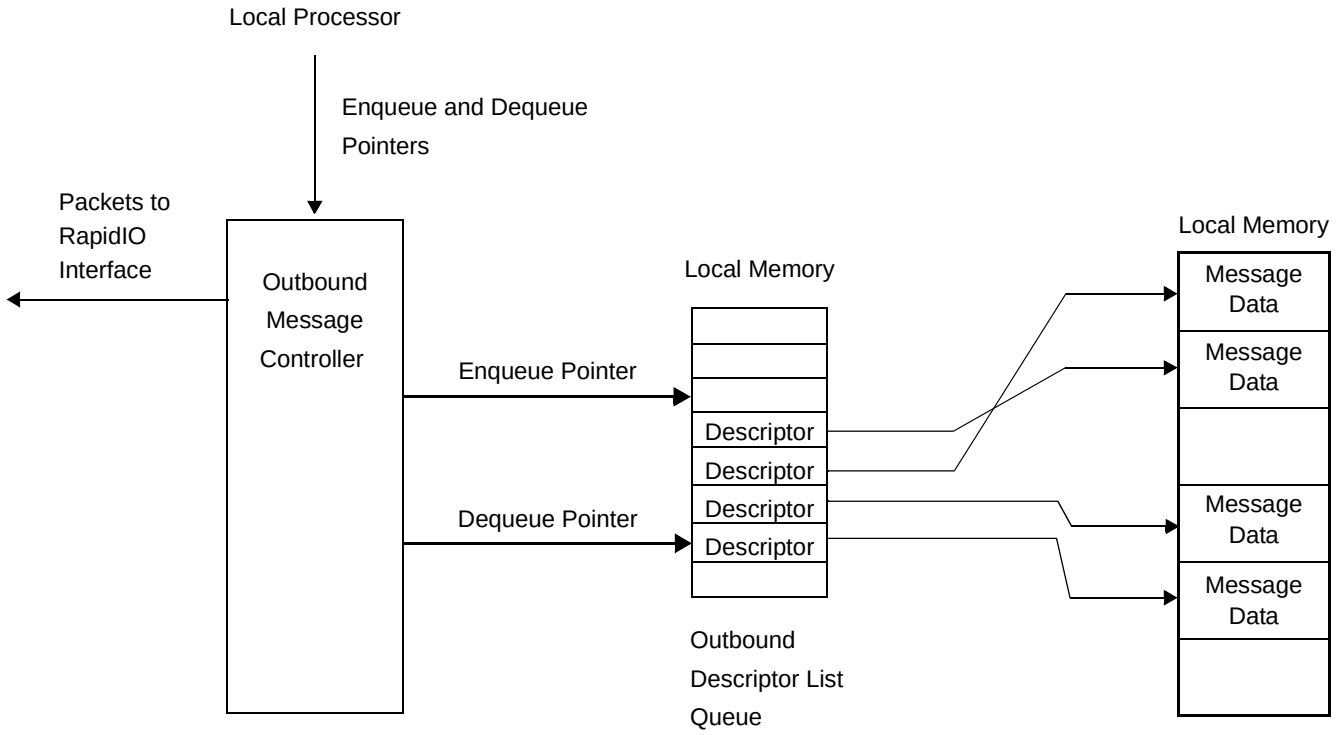


Figure 16-9. Outbound Frame Queue Structure

One of several ways software can initialize the message controller in Chaining mode is as follows:

1. Poll the status register message unit busy bit, OM*x*SR[MUB], to verify that the outbound message controller is not busy.
2. Clear the message unit start bit (OM*x*MR[MUS]).
3. Initialize the descriptor queue dequeue pointer address registers (OM*x*DQDPAR; see **page 16-171**) and the enqueue pointer address registers (OM*x*DQEPAR; see **page 16-176**) to the same value for proper operation.

   These registers must also be queue size aligned on a boundary equal to the number of queue entries × 32 bytes (the size of each queue descriptor). For example, there are 16 entries in the queue, the register must be 512-byte aligned.

   The number of queue entries is set in OMnMR[CIRQ_SIZ]. See **Section 16.6.60**, *Outbound Message x Mode Registers (OMxMR)*, on page 16-167.
4. Initialize the retry error threshold in the outbound message retry error threshold configuration register (OM*x*RETCR; see **page 16-177**).
5. Clear OM*x*MR[MUTM] for Chaining mode.
6. If you are using single-segment multicast mode, set OM*x*MR[MM].
7. Configure the other control parameters in the mode register (OM*x*MR).
8. Clear OM*x*SR[MER, PRT, RETE, TE, QOI, QFI, EOMI, and QEI]. If OM*x*SR[MER, PRT, RETE, TE, or QOI] are not cleared, the message controller does not start a new message operation. Incorrect status is indicated if the other status bits are not cleared.
9. Set the message unit start (OM*x*MR[MUS]) to enable the outbound message controller and cause the descriptor queue dequeue pointer (OM*x*DQDPAR) to be saved as the base address of the descriptor queue.

The method to start and complete transfers by adding descriptors after initializing the message unit is as follows:

1. Create one or more descriptors in local memory starting at the address to which the descriptor queue enqueue pointer address register (OM*x*DQEPAR) is pointing.
2. Either increment the enqueue pointer address registers (OM*x*DQEPAR) by setting OM*x*MR[MUI] for each descriptor entry added or directly change the enqueue pointer address register (OM*x*DQEPAR). If software sets OM*x*MR[MUI], the message controller clears this bit after successfully incrementing the enqueue pointer.
3. When the descriptor queue is not empty, the message controller reads the descriptor from local memory using the address to which the dequeue pointer (OM*x*DQDPAR) is pointing and sets the busy bit (OM*x*SR[MUB]).

4. The message controller reads the next descriptor from local memory, if available, using the address to which the dequeue pointer (OM*x*DQDPAR) is pointing. The message controller does not prefetch more than one descriptor.
5. The message controller sets OM*x*SR[MUB] to indicate that the message transfer is in progress. OM*x*SR[MUB] remains set until the descriptor queue is empty or a transaction error occurs.
6. Software can create and enqueue additional descriptors while the message controller is busy (OM*x*SR[MUB]). Software can continue adding descriptors as long as the descriptor queue is not full. If software adds descriptors using the OM*x*MR[MUI] bit, overflowing the queue can be prevented by polling the queue full bit (OM*x*SR[QF]) before creating and enqueueing the next descriptor.
7. After the descriptor memory read completes, the corresponding message segment is read from local memory.
8. If a message has multiple segments, the outbound message controller reads the other message segments from local memory.
9. After the message read to local memory completes, the message is sent.
10. If multi-cast is enabled, all the indicated targets are sent the same message.
11. A non-multicast message transfer completes after all message segments complete. A multi-cast message transfer completes after all message segments complete for each destination. A message segment completes when one of the following occurs:
    - Done response received
    - Error response received
    - Packet response time-out
    - Retry error threshold exceeded
    - Internal error during the descriptor (all message segments complete) or message read of local memory
12. When processing for the current descriptor completes and another descriptor is available, the preceding steps are repeated.
13. If a RapidIO error response is received, the message error response bit is set (OM*x*SR[MER]) and the outbound message controller operation stops after all message segments complete. If OM*x*MR[EIE] is set, the interrupt Serial RapidIO error/write-port is generated.
14. If a packet response time-out occurs, the packet response time-out bit is set (OM*x*SR[PRT]). If OM*x*MR[EIE] is set, the interrupt Serial RapidIO error/write-port is generated.
15. If the retry error threshold value is exceeded for a specific segment, the retry error threshold exceeded bit is set (OM*x*SR[RETE]) and outbound message controller

operation stops after all message segments complete. If OM*x*MR[EIE] is set, the interrupt Serial RapidIO error/write-port is generated.

16. If an internal error occurs while local memory is read, the transaction error bit is set (OM*x*SR[TE]) and outbound message controller operation stops after all message segments complete. If OM*x*MR[EIE] is set, the interrupt Serial RapidIO error/write-port is generated.

    This process continues until the descriptor queue is empty (dequeue pointer equals the enqueue pointer).

17. The message unit clears OM*x*SR[MUB] after it processes the last descriptor or a transaction error occurs.

18. If an error occurs, the message unit must be disabled, reinitialized, and reenabled before another message can be sent.

16.3.2.5.1 Changing Descriptor Queues in Chaining Mode

When software switches to another descriptor queue in local memory, it must wait for the processing of the current queue to complete, as indicated by the busy bit (OM*x*SR[MUB]). Software then disables the message controller by clearing OM*x*MR[MUS], changes the enqueue and dequeue descriptor pointers (OM*x*DQEPAR and OM*x*DQDPAR), and reenables the message unit by setting OM*x*MR[MUS].

16.3.2.5.2 Preventing Queue Overflow in Chaining Mode

Software must guarantee that descriptors are not added to an already full queue. When the increment bit is used (OM*x*MR[MUI]), software can poll the queue full bit (OM*x*SR[QF]) before enqueueing another descriptor. When software sets the enqueue pointer directly, software is responsible for not overflowing the descriptor queue.

16.3.2.5.3 Switching Between Direct and Chaining Modes

The message unit architecture allows switching from Direct mode to Chaining mode and *vice versa* after all required parameters are initialized in the appropriate registers and the message unit is not busy, as indicated by clearing of the OM*x*SR[MUB]. If OM*x*MR[MUS] is cleared and then set during a switch from Direct mode to Chaining mode, the message unit is reinitialized in Chaining mode and the outbound message descriptor dequeue pointer address is saved as the new base address of the circular queue in memory. During a switch from Chaining mode to Direct mode, OM*x*MR[MUS] must also be cleared and set.

16.3.2.5.4 Chaining Mode Descriptor Format

The descriptor contains information the message unit controller needs to transfer data. Software must ensure that each descriptor is aligned on a 32-byte boundary and that the descriptor queue is on a queue boundary, that is, on a boundary equal to the number of queue entries × 32 byte (the size of each queue descriptor). For each descriptor in the queue, the message unit controller starts

a new message operation with the control parameters specified by the descriptor, as shown in **Table 16-26**.

Table 16-26. Outbound Message Unit Descriptor Summary

Descriptor Field	Description
Reserved	—
Source address	Source address of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the Source Address Register.
Destination port	Destination port of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the Destination Port Register.
Destination attributes	Transaction attributes of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the Destination Attributes Register.
Multicast group	Logical multi-cast group. Groups are defined as a list of 32 numerically consecutive destinations by deviceID.
Multi-cast list	Bit vector list of consecutive destinations by deviceID.
Double-word count	Number of double-words for the message operation. After the message controller reads the descriptor from memory, this field is loaded into the Double-Word Count Register.
Reserved	—

Figure 16-10 depicts the queue dequeue pointer and an associated descriptor. The descriptor is valid only if the enqueue and dequeue pointers are not equal.

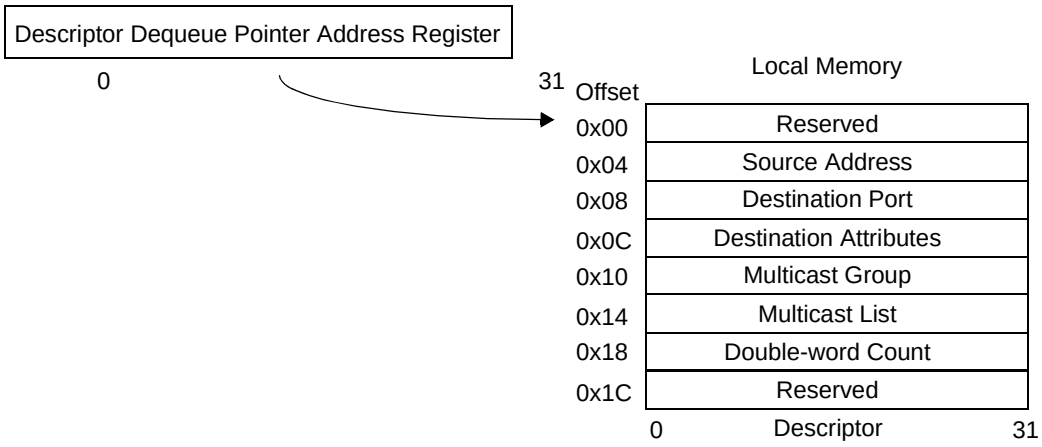


Figure 16-10. Descriptor Dequeue Pointer and Descriptor

16.3.2.5.5 Chaining Mode Controller Interrupts

An outbound message interrupt can be generated for one of the following reasons.

- *Queue Empty*. The queue goes empty and the interrupt event is enabled (OM*x*MR[QEIE] = 1). The event causing the outbound message interrupt is indicated by OM*x*SR[QEI]. The interrupt is held until the queue is not empty and the OM*x*SR[QEI] bit is cleared by writing a value of 1 to it.

- *Queue Full*. The queue is full and the interrupt event is enabled (OM*x*MR[QFIE] = 1). The event causing the outbound message interrupt is indicated by OM*x*SR[QFI]. The interrupt is held until the queue is not full and the OM*x*SR[QFI] bit is cleared by writing a value of 1 to it.
- *Queue Overflow*. The queue is full, the increment bit is set (OM*n*MR[MUI]), and the interrupt event is enabled (OM*x*MR[QOIE] = 1). The event causing the outbound message interrupt is indicated by OM*x*SR[QOI]. The message unit must be reinitialized. The interrupt is held until the OM*x*SR[QOI] bit his cleared by writing a value of 1 to it. This interrupt is also generated if the enqueue pointer is directly written and causes an overflow.
- *End-Of-Message*. The message completed and the interrupt event is enabled (OM*x*DATR[EOMIE] = 1). The event causing this interrupt is indicated by OM*x*SR[EOMI]. The interrupt is held until the OM*x*SR[EOMI] bit is cleared by writing a value of 1 to it. This allows an interrupt to be generated after a particular descriptor is processed.

An error/port-write interrupt can be generated for the following reasons in Chaining mode.

- *Message error response*. Message error response is received and this interrupt event is enabled (OM*x*MR[EIE])
- *Packet response time-out.* A packet response time-out occurs and this interrupt event is enabled (OM*x*MR[EIE]).
- *Retry error threshold exceeded*. A retry threshold exceeded error occurs and this interrupt event is enabled (OM*x*MR[EIE]).
- *Transaction error*. A message error response is received and this interrupt event is enabled (OM*x*MR[EIE]).

Table 16-23 describes each of these error types.

Table 16-27. Error Types In Outbound Message Controller Direct Mode

Error Type	Message Controller Response to Error
Message Error Response	• Sets the message error response status bit (OM*x*SR[MER]). • Generates the Serial RapidIO error/write-port if OM*x*MR[EIE] is set. • Stops after the message operation completes (indicated by OM*x*SR[MUB]).
Packet Response Time-Out	• Sets the packet response time-out status bit (OM*x*SR[PRT]). • Generates the Serial RapidIO error/write-port interrupt if OM*x*MR[EIE] is set. • Stops after the message operation completes (indicated by OM*x*SR[MUB]).
Retry Error Threshold Exceeded	• Sets the retry threshold exceed status bit (OM*x*SR[RETE]) • Generates the Serial RapidIO error/write-port interrupt if OM*x*MR[EIE] is set. • Stops after the message operation completes (indicated by OM*x*SR[MUB]).

Table 16-27. Error Types In Outbound Message Controller Direct Mode (Continued)

Error Type	Message Controller Response to Error
Transaction error	• Sets the transaction error bit (OM*x*SR[TE]). • Does not generate message segment reads and sends no message segments if the internal error occurs while the memory controller reads descriptor memory. • Does not sent message segments with an internal error because the message data is not available. • Does not transfer memory reads generated before the internal error. • Generates additional memory reads for the same message operation but does not transfer them. • Does not transfer subsequent message segments for the same message operation. This includes retried message segments. • Stops after the message operation completes (indicated by OM*x*SR[MUB]). • Generates the Serial RapidIO error/write-port interrupt if OM*x*MR[EIE] is set.

16.3.2.6 Software Error Handling

When an error occurs and the Serial RapidIO error/write-port interrupt is generated, software takes the following actions:

1. Determines the cause of the interrupt and processes the error.
2. Verifies that the message controller has stopped operation by polling OM*x*SR[MUB].
3. Disables the message controller by clearing OM*x*MR[MUS].
4. Clears the error by writing a 1 to the corresponding OM*x*SR status bit (MER, PRT, RETE, or TE).

When an error occurs and the Serial RapidIO error/write-port interrupt is not enabled, software takes the following actions:

1. Determines that an error has occurred by polling the following OM*x*SR status bits (see **Table 16-104,** *OMxSR Field Descriptions*, on page 16-170:
   - MER
   - PRT
   - RETE
   - TE
2. Verifies that the message controller has stopped by polling OM*x*SR[MUB].
3. Disables the message controller by clearing OM*x*MR[MUS].
4. Clears the error by writing a 1 to the corresponding OM*x*SR status bit (listed in step 1).

16.3.2.7 Hardware Error Handling

Table 16-28 shows error conditions in addition to those for Direct mode. All the errors listed in **Table 16-24,** *Outbound Message Direct Mode Hardware Errors*, on page 16-57 can also occur. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. When an error is detected no additional error checking beyond the current level is performed. The first error detected in the processing pipeline updates the error management extensions registers. The messaging unit provides these error condition checks in addition to the error condition checks provided by the RapidIO port and described in **Section 16.2.10**, *Errors and Error Handling*, on page 16-25.

Table 16-28. Additional Hardware Error Conditions

Transaction	Error	Description
Message request	Internal error during a read of the descriptor from local memory	**Error checking level:** 0 **Interrupt generated:** Serial RapidIO error/write-port if OM*x*MR[EIE] is set. **Status bit set**: Transaction error in the outbound message status register (OM*x*SR[TE]). Message Failed in the Mailbox CSR (MCSR[FA]). **Message segment sent:** No **Logical/Transport Layer Capture Register:** **Comments:** Message controller stops. Note that the descriptor dequeue pointer is not incremented.

Table 16-29 lists programming errors that result in undefined or undesired hardware operation. These errors are in addition to those listed in **Table 16-25,** *Outbound Message Direct Mode Programming Errors*, on page 16-60.

Table 16-29. Outbound Message Chaining Mode Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Enqueued descriptor address is invalid	No	No	Local memory captures the transaction and generates an interrupt.
Address for descriptor enqueue address pointer is invalid	No	No	Local memory captures the transaction and generates an interrupt.
Descriptor enqueue and dequeue pointers are not initialized to the same value	No	No	Undefined operation
Descriptor queue size set to a reserved value	No	No	Undefined operation
Address of descriptor enqueue pointer set to a value outside of queue	No	No	Undefined operation
Enqueueing of descriptors causes descriptor queue overflow	Outbound message interrupt enable set (OM*x*MR[QOIE])	Queue overflow (OM*x*SR[QOI])	Message controller stops.
Queue misaligned	No	No	May result in duplicate messages being sent.

16.3.2.8 Outbound Message Controller Arbitration

There are two ways to define the order in which each message controller sends messages from its message queues when both outbound message controllers are enabled:

- *Fixed priority*. The lowest numbered message unit has the highest priority. Message unit 0 has the highest priority.
- *Rotating priority*. The message units take turns sending messages in round-robin (message units 0, 1, 0, 1 and so on). A message controller can send 1–64 messages.

OM*x*MR[SCNTL] (see **page 16-167**) configures the message units for the fixed or rotating modes of arbitration.

In fixed-priority arbitration, all message segments for message unit 0 must complete before message unit 1 can start. However, in rotating priority arbitration, the other message unit can start processing a message as soon as all message segments are transmitted. Also, in fixed-priority arbitration when a message unit other than message unit 0 is processing a message, message unit 0 can start processing a message as soon as all message segments are transmitted.

16.3.3 Inbound Message Controller Operation

The inbound message controller receives messages and places them in a circular frame queue in local memory. It can receive segments of a message in any order. The address to which a message is to be written is computed as: Base address + (msgseg × ssize in double-words). In contrast to the outbound message controller for which software controls the enqueue pointer and hardware controls the dequeue pointer, the inbound message controller controls the enqueue pointer and the software controls the dequeue pointer.

Figure 16-11 depicts a sample structure of the inbound message frames and the frame pointers. In this example, the frame queue has eight entries, three of which are currently valid. The inbound controller controls the enqueue pointer and the software controls the dequeue pointer. After a configured number of messages is received, an interrupt is generated to the processor. After processing a received message, the local processor can either write the inbound message mode register mailbox increment bit (IM*x*MR[MI]]), causing the dequeue pointer to point to the next message frame in the queue, or wait until all received messages are processed and write to the dequeue pointer.

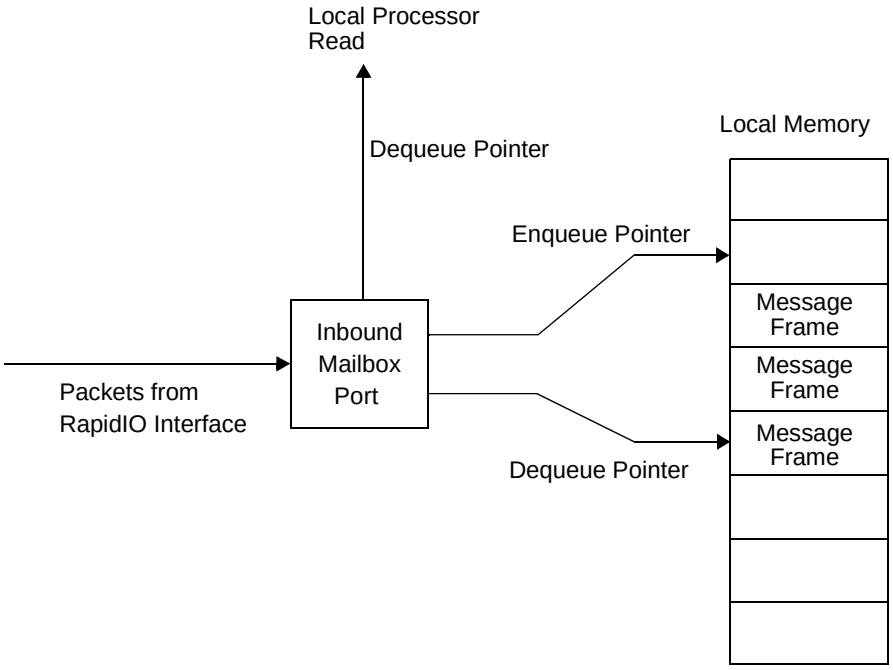


Figure 16-11. Inbound Message Structure

16.3.3.1 Inbound Message Controller Initialization

The sequence of events to initialize the inbound message controller is as follows:

1. Initialize the frame queue dequeue pointer address registers (IM*x*FQDPAR) and the frame queue enqueue pointer address registers (IM*x*FQEPAR) to the same value for proper operation. These registers must also be queue size aligned, that is, they must be aligned on a boundary equal to the number of queue entries × frame size in byte. For example, if there are eight entries in the queue and the frame size is 128 bytes, the register must be 1024-byte aligned.
2. Clear the status register (IM*x*SR).
3. In the inbound message mode register (IM*x*MR), set the mailbox enable bit (IM*x*MR[ME]) along with the other control parameters (frame queue size, message-in-queue threshold, frame size, snoop enable, and the various interrupt enables).

16.3.3.2 Inbound Controller Operation

There are many ways in which software can interact with the message controller. One method is as follows:

1. The inbound message controller receives a message segment request from the RapidIO port. If the inbound message controller is enabled (IM*x*MR[ME] =1), the inbound message controller has received all the segments for the previous message, and the frame queue is not full then the message segment is accepted.
2. The inbound message controller computes the address for each segment of the message (up to 16 segments per message) using the value of the inbound message frame queue enqueue pointer address registers and the segment number.
3. The inbound message controller writes each segment to the circular queue in local memory at the computed address.
4. When the entire message is received and all message segments are written, the inbound message controller increments the enqueue pointer to point to the next message frame in local memory. A message operation completes after all message segments for the message are complete. A message segment is complete when one of the following occurs:
   - The memory write completes (either successfully or with an internal error).
   - A message request time-out occurs (all message segments not yet received are now complete).
5. The message segments of one message can immediately be followed by the message segments of another message under the following conditions:
   - If a message segment of a new message arrives before all memory writes complete for the previous message and the RapidIO priority (PRIO field value in the message packet) of the new message is equal to or lower than that of all previous messages, the message controller processes the message and a memory write is generated to the appropriate frame queue entry.
   - If the RapidIO priority of the new message is higher than that of any previous message memory writes not yet complete, the message controller generates a retry.
   - If a message segment arrives before all previous message memory writes for the same message complete and the new message segment has a higher priority than the previous message segments, the message controller generates a retry.

   The message controller clears the IM*x*SR[MB] bit when all message operations complete.
6. An inbound message interrupt is generated to the local processor if the number of messages in the queue is greater than or equal to the configured message-in-queue threshold (IM*x*MR[MIQ_THRESH) and this event is enabled to generate the interrupt (IM*x*MR[MIQIE]).

7. By reading the inbound message status register (IM*x*SR[MIQI]), software detects that the message-in-queue interrupt bit is set and determines that the message-in-queue event caused the interrupt.
8. Software processes the frame queue entry to which the frame dequeue pointer address registers (IM*x*FQDPAR) are pointing.
9. Software increments the dequeue pointer address registers (IM*x*FQDPAR) by setting the message increment bit (IM*x*MR[MI]). Software determines whether there are any more messages to process by reading the queue empty bit (IM*x*SR[QE]). If the queue is not empty, the previous two steps are repeated.
10. Optionally, software reads the enqueue pointer address registers (IM*x*FQEPAR) and processes all the received messages. After message processing is complete, the dequeue pointer address registers (IM*x*FQDPAR) are written.
11. Software clears the message-in-queue interrupt bit (IM*x*SR[MIQI]) by writing a 1 to the IM*x*SR[MIQI] bit.

16.3.3.3 Message Steering

Messages are forwarded to the inbound message controllers as follows:

- Messages directed to mailbox 0 are forwarded to message controller 0.
- Messages directed to mailbox 1, 2, or 3 are forwarded to message controller 1.

16.3.3.4 Retry Response Conditions

The conditions to generate a logical layer retry (response retry) are as follows:

- The local memory circular queue is full and a message is received.
- The inbound message controller has received at least one segment of a multiple-segment message but has not received all message segments (as determined by a different RapidIO source ID, RapidIO destination ID, or mailbox).
- A message is received with a higher priority than all previous messages being written to memory, but it has not completed.

Note: If all inbound messages have the same RapidIO priority, this condition for generating a retry does not occur.

16.3.3.5 Inbound Message Controller Interrupts

The inbound message controller generates the inbound message interrupt for several different events. Each event can be individually enabled:

- Message-In-Queue interrupt is generated under one of the following conditions:

— The circular queue has accumulated the specified number of messages, and this interrupt event is enabled (IM*x*MR[MIQIE]). The event causing this interrupt is indicated by IM*x*SR[MIQI]. The interrupt is held until the dequeue and enqueue pointers indicate that the specified number of messages is not in the frame queue and the IM*x*SR[MIQI] bit is cleared by writing a 1 to it.
— The circular queue contains one or more messages, the specified number of messages has not accumulated, a message has not been dequeued for the maximum interrupt report interval, and this interrupt event is enabled (IM*x*SR[MIQIE]). The event causing this interrupt is indicated by IM*x*SR[MIQI]. The interrupt is held until the IM*x*SR[MIQI] bit is cleared by writing a 1 to it.

- Queue Full interrupt is generated when the circular queue becomes full and this interrupt event is enabled (IM*x*SR[QFIE]). The event causing this interrupt is indicated by IM*x*SR[QFI]. The interrupt is held until the queue is not full and the IM*x*SR[QFI] bit is cleared by writing a 1 to it.

The error/port-write interrupt is generated for the following reasons.

- An interrupt is generated after a message request time-out and this interrupt event is enabled (IM*x*MR[EIE]). The message request time-out counter starts after the first valid segment of a multi-segment message is received and the time-out counter is enabled.
- A transaction error interrupt is generated after an internal error response is received and this interrupt event is enabled (IM*x*MR[EIE]).

Table 16-30 describes each of these error types.

Table 16-30. Error Types In the Inbound Message Controller

Error Type	Message Controller Response to Error
Message Request Time-Out	• Sets the message request time-out status bit (IM*x*SR[MRT]). • Treats as complete all message segments not yet received. • Generates the Serial RapidIO error/write-port interrupt if IM*x*MR[EIE] is set. • Stops after all message segments complete (indicated by IM*x*SR[MB]).
Transaction Error	• Sets the transaction error bit (IM*x*SR[TE]) and enters the error state. • Returns an error response. Memory writes already generated before the internal error also return an error response. • Stops after the message operation completes (indicated by IM*x*SR[MB]). • Generates the Serial RapidIO error/write-port interrupt if IM*x*MR[EIE] is set.

16.3.3.6 Software Error Handling

When an error occurs and the Serial RapidIO error/write-port interrupt is generated, software takes the following actions:

1. Determines the cause of the interrupt and processes the error
2. Verifies that the message controller has stopped operation by polling IM*x*SR[MB].
3. Clears the error by writing a 1 to the corresponding status bit (IM*x*SR[MRT] and/or IM*x*SR[TE]).
4. Disables, reinitializes, and reenables the message unit before another message can be received.
5. Reinitializes and re-enables the message controller.

When an error occurs and the Serial RapidIO error/write-port interrupt is not enabled, software takes the following actions (see **Table 16-115,** *IMxSR Field Descriptions*, on page 16-182):

1. Determines that an error has occurred by polling the status bits (IM*x*SR[MRT] and/or IM*x*SR[TE]).
2. Verifies that the message controller has stopped operation by polling IM*x*SR[MB].
3. Disables the message controller by clearing IM*x*MR[ME].
4. Clears the error by writing a 1 to the corresponding status bit (IM*x*SR[MRT] and/or IM*x*SR[TE])

16.3.3.7 Hardware Error Handling

Table 16-31 lists the hardware error conditions. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. After an error is detected, no additional error checking beyond the current level is performed. Note that messages are processed in a pipeline. The first error detected in the processing pipeline updates the error management extensions registers. These error condition checks are provided by the messaging unit. These check are in addition to the error condition checks provided by the RapidIO port and described in **Section 16.2.10**, *Errors and Error Handling*, on page 16-25.

Table 16-31. Inbound Message Hardware Errors

Error	Description
Reserved ftype[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]. **Queue entry written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments**: Packet is ignored and discarded.
Reserved tt encoding[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Queue entry written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments**: Packet is ignored and discarded.
Large transport size when operating in small transport size or small transport size when operating in large transport size[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set. **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Queue entry written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments**: Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Illegal destination ID[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITTE] is set. **Status bit set**: Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments**: Packet is ignored and discarded.
Incorrect message packet size[1] Payload is not the specified ssize except for the last segment. The last segment payload can be less than or equal to the segment size but not 0. **Note:** Payload sizes are 64-bit multiples.	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set. **Status bit set**: Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. Comments: Packet is ignored and discarded.
Reserved ssize field[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set. **Status bit set**: Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet **Comments**: Packet is ignored and discarded.

Table 16-31. Inbound Message Hardware Errors

Error	Description
Message received and no mailboxes are supported as indicated by DOCAR[M][1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments**: Packet is ignored and discarded.
Message packet size larger than the configured frame size and message controller is enabled	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set. **Status bit set**: Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments**: Packet is ignored and discarded.
Inbound message packet with a RapidIO priority of 3	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set. **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Queue entry written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments**: Packet is ignored and discarded.
Not an error. The RapidIO priority is not consistent for all message segments of a message	**Error checking level:** 2 **Interrupt generated:** **Status bit set**: **Queue entry written in local memory:** Yes **Response status:** Done or retry **Logical/Transport Layer Capture Register:** **Comments:** Retry response occurs if the higher-priority message segment is received while the memory write for a corresponding lower-priority message segment is outstanding.
Message segment number is larger than the number of message segments in the message	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set. **Status bit set**: Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments:** Packet is ignored and discarded.
Duplicate message segment is received All segments of a multi- segment message must be received before the next message begins.	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set. **Status bit set**: Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments:** Packet is ignored and discarded.

Table 16-31. Inbound Message Hardware Errors

Error	Description
msglen (number of segments in a message) is not consistent in all segments of a multi-segment message	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] set. **Status bit set**: Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue entry written in local memory:** No. **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments:** Packet is ignored and discarded.
ssize (segment size) is not consistent in all segments of a multi-segment message	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set. **Status bit set**: Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments:** Packet is ignored and discarded.
Message received for an unsupported mailbox but at least one mailbox is supported	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments:** Packet is ignored and discarded. This error applies only if a mailbox is not supported. This error is not currently supported since all mailboxes are supported.
Message controller disabled and message received	**Error checking level:** 2 **Interrupt generated:** No **Status bit set**: None **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** **Comments:** Packet is ignored and discarded.
Message controller enabled but in the error state and message received	**Error checking level:** 2 **Interrupt generated:** No **Status bit set**: None **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** **Comments:** Packet is ignored and discarded.
Internal error during the write of the frame queue entry to memory	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if IM*x*MR[EIE] is set. **Status bit set**: Transaction error in the Message Status Register (IM*x*SR[TE]). Message failed in the Message CSR (MCSR[FA]). **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** **Comments:** Message controller stops after the current message operation completes. The enqueue pointer is not incremented.

Table 16-31. Inbound Message Hardware Errors

Error	Description
Internal error for an earlier posted frame queue entry memory write and a subsequent frame queue entry memory write is posted before the internal error is detected. An internal error may or may not occur during the subsequent frame queue entry memory write. The frame queue could be for the same message or a different message	**Error checking level:** 4 **Interrupt generated:** No **Status bit set**: None **Queue entry written in local memory:** Yes **Response status:** Error **Logical/Transport Layer Capture Register:** **Comments:**
Message request to request time-out for multi-segment messages	**Error checking level:** Unrelated **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MRT] set. Serial RapidIO error/write-port if OM*x*MR[EIE] is set. **Status bit set**: Message request time-out in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MRT]. IM*x*SR[MRT] bit is set. **Queue entry written in local memory:** No **Response status:** No **Logical/Transport Layer Capture Register:** Updated with the previous message request packet except that the message segment field (bits 4–7 of LTLCCCSR[MI]) is updated with the lowest message segment number not yet received.[2] **Comments:** All message segments received before the time-out update memory. The enqueue pointer is not incremented. The message operation completes.

Notes:

1. These error types are actually detected in the RapidIO port, not in the message controller.

2. In small transport size configuration using the packet, the following allocations are made:

- LTLACCSR[XA] gets the extended address (packet bits 78–79).
- LTLACCSR[A] gets the address (packet bits 48–76).
- LTLDIDCCSR[MDID] gets 0.
- LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23).
- LTLDIDCCSR[MSID] gets 0.
- LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31).
- LTLCCCSR[FT] gets the ftype (packet bits 12–15).
- LTLCCCSR[TT] gets the ttype (packet bits 32–35).
- LTLCCCSR[MI] gets the msg info (packet bits 40–47).

In large transport size configuration using the packet, the following allocations are made:

- LTLACCSR[XA] gets the extended address (packet bits 94–95)
- LTLACCSR[A] gets the address (packet bits 64–92)
- LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23)
- LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31)
- LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32–39)
- LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47)
- LTLCCCSR[FT] gets the ftype (packet bits 12–15)
- LTLCCCSR[TT] gets the ttype (packet bits 48–51)
- LTLCCCSR[MI] gets the msg info (packet bits 56–63).

16.3.3.8 Programming Errors

Table 16-32 shows a partial list of programming errors that result in undefined or undesired hardware operation.

Table 16-32. Inbound Message Programming Errors

Error	Interrupt	Status Bit Set	Comments
Reserved value of the message in queue threshold (IMxMR[MIQ_THRESH]) or reserved value of the circular frame queue size (IMxMR[CIRQ_SIZE])	No	No	Undefined operation results
The message in-queue threshold is equal to the frame queue size	No	No	Message in queue interrupt occurs when queue is full
The message in-queue threshold is greater than the frame queue size	No	No	Message in queue interrupt never occurs
Frame queue entry written to non-existent memory	No	No	Memory controller will cause the interrupt and update capture registers. IM*x*SR[TE] will be set due to the internal error.
Message enqueue and dequeue pointers are not initialized to the same value	No	No	Undefined operation results
The dequeue frame pointer register is set incorrectly.	No	No	Undefined operation results
Queue misaligned.	No	No	Undefined operation results.

16.3.3.9 Disabling and Enabling the Inbound Message Controller

When the message controller is disabled by clearing IM*x*MR[ME] the following occurs (see **Table 16-115,** *IMxSR Field Descriptions*, on page 16-182):

1. Queue full clears the IMxSR[QF] bit.
2. Message-in-queue clears the IMxSR[MIQ] bit.
3. Queue empty is set via the IMxSR[QE] bit.
4. Message busy clears the IMxSR[MB] bit after all pending frame queue entry writes complete.

Once the message controller is disabled, an error response is generated for all new message packets. If the message controller is disabled before all of the message segments for a multisegment message are received, a message request time-out must occurs and all pending frame queue writes must complete before message busy clears (IMxSR[MB]).

Before the message controller is reenabled, the message busy bit must be clear (IM*x*SR[MB = 0) and the (IM*x*MR[ME]), frame queue dequeue pointer address registers (IM*x*FQDPAR), and frame queue enqueue pointer address registers (IM*x*FQEPAR) must be initialized to the same value for proper message controller operation.

16.3.4 RapidIO Message Passing Logical Specification Register Bits

The Mailbox Command and Status Register (MCSR; see **page 16-111**) provides the status for the two inbound and the two outbound controllers. These read-only status bits indicate the state of each of the message controllers, as described in **Table 16-33**.

Table 16-33. MCSR Bits to Indicate Status of Inbound and Outbound Controllers

MCSR Bit	Description
Available (A)	Indicates the following: • The inbound message controller is enabled (IM*x*MR[ME]). • The inbound message controller is not in the internal error state (IM*x*SR[TE] = 0). • The inbound message controller did not detect a message request time-out (IM*x*SR[MRT] = 0).
Full (FU)	Reflects the inbound message controller queue full status.
Empty (EM)	Reflects the state of the outbound message controller message empty status.
Busy (B)	Reflects the state of the inbound message controller busy bit IM*x*SR[MB].
Failed (FA)	Set if any of the following bits are set: • Inbound message controller transaction error status bit IM*x*SR[TE]. • Inbound message controller message request time-out status bit IM*x*SR[MRT]. • Outbound message controller transaction error status bit OM*x*SR[TE]. • Outbound message controller packet response time-out bit OM*x*SR[PRT]. • Outbound message controller message error response received status bit OM*x*SR[MER]. • Outbound message controller retry error threshold exceeded status bit OM*x*SR[RETE].
Error (ERR)	Always has a value of 0.

16.4 RapidIO Doorbell

This section describes the operation of the doorbell controllers, which are part of the RapidIO message unit. The doorbell controllers are designed to comply with the message passing logical specification contained in the *RapidIO Interconnect Specification*, Revision 1.2. The doorbell controller generates and receives doorbells.

The doorbell controllers are controlled through a set of run-time registers.

16.4.1 Features

- Support for one outbound doorbell controllers
- Support for one inbound doorbell controllers
- The doorbell controller can sustain back-to-back inbound doorbells

16.4.2 Doorbell Controller

The RapidIO architecture specification defines a doorbell type with no data payload that is transferred using inbound and outbound doorbell controllers. The MSC8144E RapidIO interconnect doorbell unit supports both inbound and outbound doorbells. The doorbell entry size is fixed at 64 bits because doorbell packets only pass a small amount of information, making the enqueue and dequeue pointers double-word addresses.

Inbound doorbells are handled by the doorbell controller similar to the way the message controller handles inbound data messages. The doorbell controller receives the doorbells and places them in a circular queue located in local memory. Additional doorbells can be received and forwarded to memory before the previous doorbell memory write completes as long as the RapidIO priority is the same or lower than the previous doorbell. The doorbell is retried if the RapidIO priority is higher than the previous doorbell.

Table 16-12 depicts an example of the structure of the inbound doorbell queue and pointers. The doorbell queue has eight entries, three of which are currently valid. The doorbell controller enqueues doorbells and the local processor dequeues doorbells.Each inbound doorbell controller has a dedicated interrupt to notify software that there are incoming doorbells.

Outbound doorbells are generated through a memory-mapped write port rather than a queue. An outbound doorbell must complete before another outbound doorbell can be generated. The outbound doorbell controller has a dedicated interrupt used to notify software that a doorbell was sent.

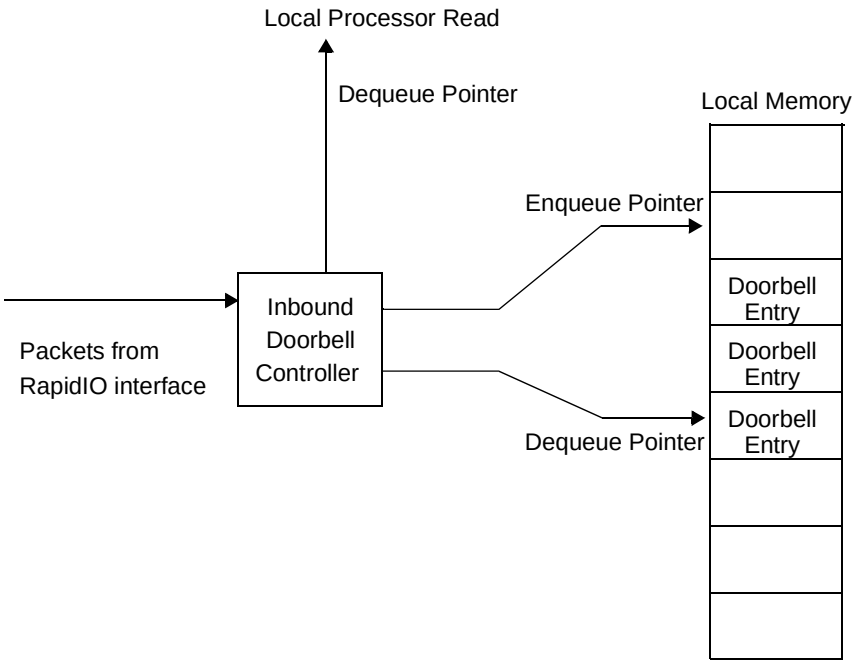


Figure 16-12. Inbound Doorbell Queue and Pointer Structure

16.4.3 Outbound Doorbell Controller

The outbound doorbell controller generates doorbells. Software initializes all parameters in the necessary registers to start the doorbell transmission. The doorbell transfer starts when the doorbell start bit, DUS, in the Outbound Doorbell Mode Register (ODMR) transitions from 0 to 1 (see **Table 16-119,** *ODMR Field Descriptions*, on page 16-187) and the doorbell controller is not busy. Software should program all appropriate registers before setting ODMR[DUS].

Software can interact with the doorbell controller in many ways. One example method for generating a doorbell is as follows:

1. Poll the status register doorbell unit busy bit, ODSR[DUB], to ensure that the outbox is not busy (see **Table 16-120,** *ODSR Field Descriptions*, on page 16-188).
2. Clear the following ODSR status bits:
   - MER
   - RETE
   - PRT
   - EODI]
3. Initialize the following registers:
   - Destination port (ODDPR; see **page 16-189**)
   - Destination attributes (ODDATR; see **page 16-190**)
   - Retry error threshold (ODRETCR; see **page 16-191**)
4. To start the doorbell transfer, clear and then set the doorbell start bit, ODMR[DUS].
5. ODSR[DUB] is set when ODMR[DUS] transitions from 0 to 1 to indicate that the doorbell transfer is in progress.
6. The outbound doorbell controller sends the doorbell.
7. The outbound doorbell controller clears the ODSR[DUB] bit after the doorbell operation completes. A doorbell completes when one of the following events occurs:
   - Done response received.
   - Error response received.
   - Packet response time-out.
   - Retry limit exceeded.
8. The outbound doorbell interrupt is generated if the end of doorbell outbound doorbell interrupt event is enabled (ODDATR[EODIE]).

16.4.3.1 Interrupts

The outbound doorbell controller interrupt is generated after the completion of a doorbell (done, error, packet response time-out, or retry limit exceeded) if this interrupt event is enabled (ODDATR[EODIE] =1). The event causing this interrupt is indicated by ODSR[EODI]. The interrupt is held until the ODSR[EODI] bit is cleared by writing a 1 to it.

The error/port-write interrupt is generated for the following errors under the specified conditions:

- *Error response received*. An interrupt is generated if the interrupt event is enabled (ODMR[EIE]).
- *Packet response timeout*. An interrupt is generated after a packet response time-out occurs and this interrupt event is enabled (ODMR[EIE]).
- *Retry error threshold exceeded*. An interrupt is generated after a retry threshold exceeded error occurs and this interrupt event is enabled (ODMR[EIE]).

Table 16-34 describes each of these error types.

Table 16-34. Error Types In the Outbound Doorbell Controller

Error Type	Doorbell Controller Response to Error
Error Response Error	• Sets the message error response status bit (ODSR[MER]). • Generates the Serial RapidIO error/write-port if ODMR[EIE] is set. • Stops after the doorbell operation completes (indicated by ODSR[DUB]).
Packet Response Time-Out Error	• Sets the packet response time-out status bit (ODSR[PRT]). • Generates the Serial RapidIO error/write-port interrupt if ODMR[EIE] is set. • Stops after the doorbell operation completes (indicated by ODSR[DUB]).
Retry Error Threshold Exceeded Error	• Sets the retry threshold exceed status bit (ODSR[RETE]). • Generates the Serial RapidIO error/write-port interrupt if ODMR[EIE] is set. • Stops after the doorbell operation completes (indicated by ODSR[DUB]).

16.4.3.2 Software Error Handling

When an error occurs and the Serial RapidIO error/write-port interrupt is generated, software takes the following actions:

1. Determines the cause of the interrupt and processes the error.
2. Verifies that the doorbell controller has stopped operation by polling ODSR[DUB].
3. Disables the doorbell controller by clearing ODMR[DUS].
4. Clears the error by writing a 1 to the corresponding ODSR status bit (see **Table 16-120,** *ODSR Field Descriptions,* on page 16-188):
   - MER
   - PRT

- RETE

When an error occurs and the Serial RapidIO error/write-port interrupt is not enabled, software takes the following actions:

1. Determines that an error has occurred by polling the ODSR status bits (MER, PRT, and/or RETE).
2. Verifies that the doorbell controller has stopped operation by polling ODSR[DUB].
3. Disables the doorbell controller by clearing ODMR[DUS].
4. Clears the error by writing a 1 to the corresponding ODSR status bit (listed in step 1).

Note: Once the doorbell controller starts, it cannot be stopped.

16.4.3.3 Hardware Error Handling

Table 16-35 lists possible error conditions for outbound doorbells and how they are handled. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. When an error is detected, no additional error checking beyond the current level is performed. Note that outbound doorbell responses are processed in a pipeline. The first error detected in the processing pipeline updates the error management extensions registers.

These error condition checks are provided by the messaging unit. These check are in addition to the error condition checks provided by the RapidIO port as discussed in **Section 16.2.10**, *Errors and Error Handling*, on page 16-25.

Table 16-35. Outbound Doorbell Hardware Errors

Transaction	Error	Description
Undefined packet	Reserved ftype encoding[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT] (see **page 16-131**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	Reserved tt encoding[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set. **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE] (see **page 16-131**). **Doorbell sent:** Yes. **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.

Table 16-35. Outbound Doorbell Hardware Errors (Continued)

Transaction	Error	Description
Doorbell response	Large transport size in small transport mode or small transport size in large transport mode.[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] set. **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE] (see **page 16-131**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Doorbell response	Illegal Destination ID[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITTE] is set. **Status bit set**: Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE] (see **page 16-131**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	Doorbell not outstanding[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UR] is set. **Status bit set**: Unsolicited response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UR] (see **page 16-131**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	ttype (transaction field) is not doorbell response[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set. **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD] (see **page 16-131**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	RapidIO priority is less than or equal to outbound request[1]	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set (see **page 16-132**). **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD] (see **page 16-131**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	Incorrect Source ID[1]	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] set (see **page 16-132**). **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD] (see **page 16-131**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	Reserved response status[1]	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set (see **page 16-132**). **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD] (see **page 16-131**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.

Table 16-35. Outbound Doorbell Hardware Errors (Continued)

Transaction	Error	Description
Doorbell response	Doorbell response packet size is incorrect[1]	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set (see **page 16-132**). **Status bit set**: Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE] (see **page 16-131**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	Error response	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MER] is set (see **page 16-132**). Serial RapidIO error/write-port if ODMR[EIE] is set (see **page 16-187**). **Status bit set**: Message error response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MER] (see **page 16-131**). ODSR[MER] bit set (see **page 16-187**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the corresponding doorbell request packet.[2] **Comments**: Doorbell transfer complete.
Doorbell response	Number of retries exceeds limit	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[RETE] is set (see **page 16-132**). Serial RapidIO error/write-port if ODMR[EIE] is set (see **page 16-187**). **Status bit set**: Retry limit exceeded in the Logical/Transport Layer Error Detect CSR LTLEDCSR[RETE] (see **page 16-131**). ODSR[RETE] bit is set (see **page 16-188**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the corresponding doorbell request packet.[2] **Comments**: Doorbell transfer complete.
Doorbell response	Packet response time-out[1]	**Error checking level:** Unrelated **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[PRT] is set (see **page 16-132**). Serial RapidIO error/write-port if ODMR[EIE] is set (see **page 16-187**). **Status bit set**: Packet response time-out in the Logical/Transport Layer Error Detect CSR LTLEDCSR[PRT] in RapidIO endpoint (see **page 16-131**). ODSR[PRT] bit is set (see **page 16-188**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the doorbell request packet in RapidIO endpoint.[2] **Comments**: Doorbell transfer complete. Note that RapidIO endpoint sends a special priority 3 packet indicating a doorbell time-out.

Table 16-35. Outbound Doorbell Hardware Errors (Continued)

Transaction	Error	Description
Notes: 1.	**Notes: 1.** These error types are actually detected in the RapidIO port, not in the doorbell controller. **2.** In small transport size configuration using the packet, the following allocations are made: • LTLACCSR[XA] gets the extended address (packet bits 78–79). • LTLACCSR[A] gets the address (packet bits 48–76) • LTLDIDCCSR[MDID] gets 0. • LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23). • LTLDIDCCSR[MSID] gets 0. • LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31). • LTLCCCSR[FT] gets the ftype (packet bits 12–15). • LTLCCCSR[TT] gets the ttype (packet bits 32–35). • LTLCCCSR[MI] gets 0 In large transport size configuration using the packet, the following allocations are made: • LTLACCSR[XA] gets the extended address (packet bits 94–95). • LTLACCSR[A] gets the address (packet bits 64–92). • LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23). • LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31). • LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32–39). • LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47). • LTLCCCSR[FT] gets the ftype (packet bits 12–15). • LTLCCCSR[TT] gets the ttype (packet bits 48–51). • LTLCCCSR[MI] gets 0.	

16.4.3.4 Programming Errors

Table 16-36 lists programming errors that result in undefined or undesired hardware operation.

Table 16-36. Outbound Doorbell Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Transaction flow level set to 3	No	None	Undefined operation.
Target interface set to an invalid RapidIO port	No	None	Undefined operation.
Register values changed during operation	No	No	Undefined operation.

16.4.4 Inbound Doorbell Controller

The inbound doorbell controller receives doorbells and places them in a circular doorbell queue in local memory. The inbound controller controls the enqueue pointer and software controls the dequeue pointer. After a configured number of doorbells are received, an interrupt is generated to the processor. After processing a received doorbell, the local processor can either write the doorbell mode register increment bit (IDMR[DI]]) causing the dequeue pointer to point to the next doorbell in the queue or wait until all the received doorbells are processed and write the dequeue pointer.

There are many ways in which software can interact with the doorbell controllers. One method of initializing the inbound doorbell controller follows:

1. Initialize the doorbell queue dequeue pointer address registers IDQDPAR (see **page 16-195**) and IDQEPAR (see **page 16-196**) and the doorbell queue enqueue pointer address register (DQEPAR). These registers must be initialized to the same value for proper operation. They also must be queue size aligned.
2. Clear the status register (IDSR; see **page 16-194**).
3. Set the doorbell enable bit IDMR[DE] along with the other control parameters (doorbell queue size, doorbell-in-queue threshold, and various interrupt enables) in the doorbell mode register (IDMR; see **Table 16-124,** *IDMR Field Descriptions*, on page 16-192).

Another method follows:

1. The doorbell controller receives a doorbell. If the inbound doorbell controller is enabled (IDMR[DE] =1) and the doorbell queue is not full, then the doorbell is accepted.
2. The doorbell controller stores the 16-bit information field and the RapidIO source and destination IDs in local memory using the value of the doorbell queue enqueue pointer address register (DQEPAR).
3. When the memory write completes, the enqueue pointer is incremented to point to the next doorbell queue entry in local memory.
4. If another doorbell arrives before all previous doorbell memory writes complete and the RapidIO priority of the doorbell is equal to or lower than the priority of all previous doorbells, the doorbell controller processes the doorbell and generates a memory write to the appropriate doorbell queue entry. If the priority of the new doorbell is higher than that of the previous doorbell memory writes that have not completed, the doorbell controller generates a retry.
5. An inbound doorbell interrupt is generated to the local processor because the number of doorbells in the queue is greater than or equal to the configured doorbell-in-queue threshold (IDMR[DIQ_THRESH) and this event is enabled to generate the interrupt (IDMR[DIQIE]).
6. Software determines that the doorbell-in-queue event caused the interrupt by detecting that the doorbell-in-queue interrupt bit is set in the doorbell status register (IDSR[DIQI]).
7. Software processes the doorbell queue entry to which the doorbell queue dequeue pointer address register (DQDPAR) is pointing.
8. Software increments the dequeue pointer address register (DQDPAR) by setting the doorbell increment bit (IDMR[DI]).
9. Software determines whether there are more doorbells to process by reading the queue empty bit (IDSR[QE]). If the queue is not empty, the previous two steps are repeated.

10. Software clears the doorbell-in-queue interrupt bit (IDSR[DIQI]) by writing a 1 to it.

16.4.4.1 Doorbell Queue Entry Format

Each doorbell entry in the queue has two 32-bit words, one for target information (see **Table 16-37**) and one for source information (see **Table 16-38**). The target information is stored because a RapidIO port can be configured to accept packets from any destination. When there are multiple RapidIO ports on one device, each port can be configured with a different destination ID. For the MSC8144E, the target information is identical for all received doorbell entries.

Table 16-37. Inbound Doorbell Target Info Definition

Bit	Name	Description
31–16	—	Reserved.
15–8	ETID	Extended target ID in Large Transport mode. Reserved for Small Transport mode.
7–0	TID	Target ID field from the received doorbell packet.

Table 16-38. Source Info Definition

Bit	Name	Description
31–24	ESID	Extended source ID in Large Transport mode. Reserved fir Small Transport mode.
23–16	SID	Source ID field from the received doorbell packet.
15–8	INFO MSB	Most significant byte of the info field from the received doorbell packet.
7–0	INFO LSB	Least significant byte of the info field from the received doorbell packet.

Figure 16-13 depicts the doorbell queue entry fields and their related offsets.

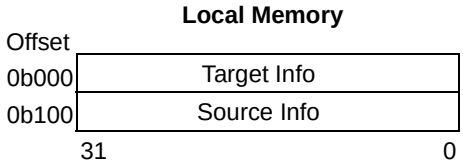


Figure 16-13. Doorbell Entry Format

16.4.4.2 Retry Response Conditions

There are two conditions in which a doorbell is retried at the logical layer (Response Retry):

- A doorbell is received and there are no entries in the doorbell queue.
- A doorbell is received with a higher priority than all previous doorbells being written to memory. If all inbound doorbells have the same priority, this condition does not occur.

16.4.4.3 Doorbell Controller Interrupts

There is one doorbell controller interrupt per inbound doorbell controller. The following events can generate the interrupt:

- *Doorbell-in-queue*:
  - The circular queue has accumulated the number of doorbells specified by the doorbell-in-queue threshold (IDMR[DIQ_THRESH) and this interrupt event is enabled (IDMR[DIQIE]). The event causing this interrupt is indicated by IDSR[DIQI]. The interrupt is held until the dequeue and enqueue pointers indicate that the specified number of doorbells is not in the doorbell queue and the IDSR[DIQI] bit is cleared by writing a 1 to it.
  - The circular queue contains one or more doorbells, the specified number of doorbells has not accumulated, a doorbell has not been dequeued for the maximum interrupt report interval, and this interrupt event is enabled (IDMR[DIQIE]). The event causing this interrupt is indicated by IDSR[DIQI]. The interrupt is held until either IDMR[DI] is set or DQDPAR[DQDPA] is written and IDSR[DIQI] is cleared by writing a 1 to it.
- *Queue Full*. An interrupt is generated each time the circular queue becomes full and this interrupt event is enabled (IDSR[QFIE]). The event causing this interrupt is indicated by IDSR[QFI]. The interrupt is held until the queue is not full and the IDSR[QFI] bit is cleared by writing a 1 to it.

The error/port-write interrupt can be generated after an internal error response is received and this interrupt event is enabled (IDMR[EIE]).

16.4.4.4 Transaction Errors

When an internal error occurs while the doorbell controller is writing to local memory the doorbell controller responds as follows:

1. Sets the transaction error bit (IDSR[TE]) and enters the error state.
2. Returns an error response.
3. Generates the Serial RapidIO error/write-port interrupt if IDMR[EIE] is set.

16.4.4.5 Software Error Handling

When an error occurs and the Serial RapidIO error/write-port interrupt is generated, software performs the following actions.

1. Determines the cause of the interrupt and processes the error.
2. Verifies that the doorbell controller has stopped operation by polling IDSR[DB].
3. Clears the error by writing a 1 to the corresponding status bit (IDSR[TE]).
4. Disables, reinitializes, and reenables the doorbell unit before another doorbell can be received.

16.4.4.6 Hardware Error Handling

Table 16-39 describes the hardware error conditions and how they are handled. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. When an error is detected, no additional error checking beyond the current level is performed. Doorbells are processed in a pipeline. The first error detected in the processing pipeline updates the error management extensions registers. These error condition checks are provided by the messaging unit. These checks are in addition to the error condition checks provided by the RapidIO port as discussed in **Section 16.2.10**, *Errors and Error Handling*, on page 16-25.

Table 16-39. Inbound Doorbell Hardware Errors

Error	Description
Reserved ftype[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Reserved tt encoding[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set. **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.

Table 16-39. Inbound Doorbell Hardware Errors

Error	Description
Large transport size in small transport size or small transport size in large transport size[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set. **Status bit set:** Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Illegal Destination ID[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITTE] is set. **Status bit set**: Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]. **Queue Entry Written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Inbound doorbell received and inbound doorbells are not supported as indicated by DOCAR[D][1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]. **Queue Entry Written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Inbound doorbell packet with a RapidIO priority of 3	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set. **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Incorrect doorbell packet size (not one datum in small transport mode or not two datums in large transport mode)	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set. **Status bit set**: Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue Entry Written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Doorbell controller disabled and doorbell received	**Error checking level:** 3 **Interrupt generated:** No **Status bit set**: None **Queue Entry Written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register** **Comments:** Packet is ignored and discarded.

Table 16-39. Inbound Doorbell Hardware Errors

Error	Description
Doorbell controller enabled but in the error state and doorbell received	**Error checking level:** 3 **Interrupt generated:** No **Status bit set**: None **Queue Entry Written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register** **Comments:** Packet is ignored and discarded.
Internal error during the write of the doorbell queue entry to memory	**Error checking level:** 4 **Interrupt generated:** Serial RapidIO error/write-port if OMMR[EIE] is set. **Status bit set**: Transaction error in the Doorbell status register (ISSR[TE]). Doorbell Failed in the Port-write and Doorbell CSR (PWDCSR[FA]). **Queue Entry Written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register** **Comments:** Doorbell controller stops after the current doorbell operation completes. The enqueue pointer is not incremented.
An internal error occurred for an earlier posted write of a doorbell queue entry to memory and a subsequent write of a doorbell queue entry to memory is posted before the internal error is detected.	**Error checking level:** 5 **Interrupt generated:** No **Status bit set**: None **Queue Entry Written in local memory:** Yes **Response status:** Error **Logical/Transport Layer Capture Register** **Comments:** An internal error may or may not occur during the subsequent write of a doorbell queue entry to memory.

Notes:

1. These error types are actually detected in the RapidIO port, not in the doorbell controller.

2. In small transport size configuration using the packet, the following allocations are made:

- LTLACCSR[XA] gets the extended address (packet bits 78–79).
- LTLACCSR[A] gets the address (packet bits 48–76).
- LTLDIDCCSR[MDID] gets 0.
- LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23).
- LTLDIDCCSR[MSID] gets 0.
- LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31).
- LTLCCCSR[FT] gets the ftype (packet bits 12–15).
- LTLCCCSR[TT] gets the ttype (packet bits 32–35).
- LTLCCCSR[MI] gets 0.

In large transport size configuration using the packet, the following allocations are made:

- LTLACCSR[XA] gets the extended address (packet bits 94–95).
- LTLACCSR[A] gets the address (packet bits 64–92).
- LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23).
- LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31).
- LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32–39).
- LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47).
- LTLCCCSR[FT] gets the ftype (packet bits 12–15).
- LTLCCCSR[TT] gets the ttype (packet bits 48–51).
- LTLCCCSR[MI] gets 0.

16.4.4.7 Programming Errors

Table 16-40 lists the programming errors that result in undefined or undesired hardware operation.

Table 16-40. Inbound Doorbell Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Reserved value of the doorbell in queue threshold (IDxMR[DIQ_THRESH]) or reserved value of the circular doorbell queue size (IDxMR[CIRQ_SIZ]); see **Table 16-124,** *IDMR Field Descriptions,* on page 16-192.	No	No	Undefined operation results
The doorbell in-queue threshold is equal to the doorbell queue size.	No	No	Doorbell in queue interrupt when queue is full
The doorbell in-queue threshold is greater than the doorbell queue size.	No	No	Doorbell in queue interrupt never occurs
Doorbell queue entry written to non-existent memory.	No	No	Memory controller causes the interrupt and updates the capture registers
Doorbell enqueue and dequeue pointers are not initialized to the same value.	No	No	Undefined operation
The dequeue pointer register is set incorrectly.	No	No	Undefined operation

16.4.4.8 Disabling and Enabling the Doorbell Controller

When the doorbell controller is disabled by clearing IDMR[DE] the following occurs in the IDSR (see **Table 16-125,** *IDSR Field Descriptions*, on page 16-194):

1. Queue full clears (QF).
2. Doorbell-in-queue clears (DIQ).
3. Queue empty is set (QE)
4. Doorbell busy clears (DUB) after all pending doorbell queue entry writes to local memory complete.

Before the doorbell controller is reenabled, the doorbell busy bit must be clear (IDSR[DB) and the doorbell dequeue pointer address register (DQDPAR) and the doorbell queue enqueue pointer address register (DQEPAR) must be initialized to the same value for proper doorbell controller operation.

16.4.4.9 RapidIO Message Passing Logical Specification Registers

The port-write and doorbell command and status register (PWDCSR) includes several doorbell controller status bits. See **Section 16.6.9**, *Port Write and Doorbell Command and Status Register (PWDCSR)*, on page 16-112 for details.

These read-only status bits indicate the state of doorbell controller 0.

- *Available (PWDCSR[A])*. Indicates that the inbound doorbell controller is enabled (IDMR[DE]) and the doorbell controller is not in the internal error state (IDnSR[TE] = 0)
- *Full (PWDCSR[FU])*. This bit reflects the inbound doorbell controller queue full status bit (IDSR[QF])
- *Empty (PWDCSR[EM])*. This bit reflects the inverted state of the outbound doorbell busy bit (ODSR[DUB] = 0)
- *Busy (PWDCSR[B])*. This bit reflects the state of the inbound doorbell controller busy bit IDSR[DUB]
- *Failed (PWDCSR[FA])*. This bit reflects the state of the transaction error status bit IDSR[TE]
- *Error (PWDCSR[ERR])*. This bit is always a 0

16.5 Port-Write Controller

The implementation of the port-write controller is very similar to the inbound message and doorbell controllers except that only one queue entry is supported. The port write is an error reporting mechanism for a device that has no end-point to communicate with a control processor or other system host. **Figure 16-14** shows the structure of the inbound queue and pointer. The port-write queue only contains one entry with a fixed size of 64 bytes and is aligned to a cache line boundary. The port-write controller uses the error/port-write interrupt for the RapidIO error/write-port to indicate incoming port-writes.

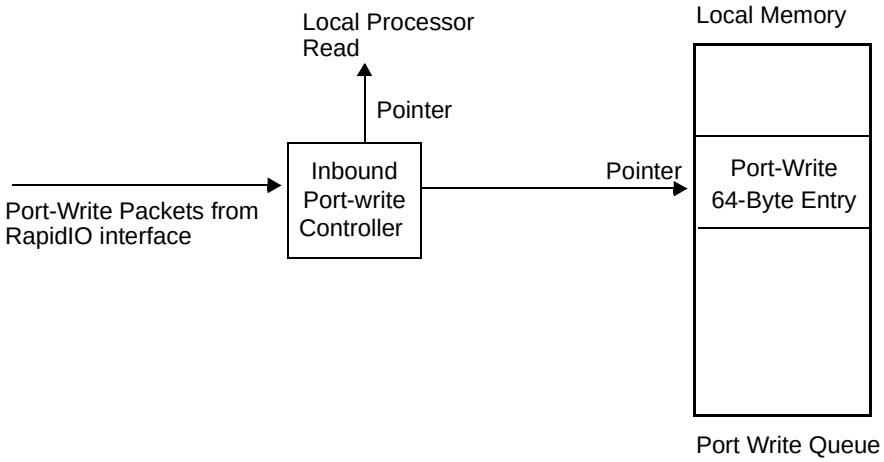


Figure 16-14. Inbound Port-Write Structure

16.5.1 Port-Write Controller Initialization

There are many ways in which software can interact with the port-write controller. One method to initialize the port-write controller is as follows:

1. Initialize the port-write queue base address registers (IPWQBAR; see **Table 16-131,** *IPWQBAR Field Descriptions*, on page 16-200).
2. Clear the status register (IPWSR; see **Table 16-130,** *IPWSR Field Descriptions*, on page 16-199).
3. Set the port-write enable bit IPWMR[PWE] along with the interrupt enable) in the inbound port-write mode register (IPWMR; see **Table 16-129,** *IPWMR Field Descriptions*, on page 16-198).

16.5.2 Port-Write Controller Operation

There are several ways in which software can interact with the port-write controller. One method is as follows:

1. The port-write controller receives a port-write from the RapidIO port. If the inbound port-write controller is enabled (IPWMR[PWE] =1) and the port-write queue is not full, the port-write is accepted.
2. The port-write controller stores 64 bytes of payload in local memory using the value of the port-write queue base address registers (IPWQBAR). Valid payload sizes include 4, 8, 16, 24, 32, 40, 48, 56, or 64 bytes. Note that 64 bytes are always written to memory. If the actual payload size is less than 64 bytes, the non payload data is undefined.
3. If the queue full interrupt enable bit is set (IPWMR[QFIE]) after the memory write completes, the port-write controller generates the error/port-write interrupt.
4. An inbound error/port-write interrupt is generated to the local processor because a port-write is received and this event is enabled to generate the interrupt (IPWMR[QFIE]). Note that the RMU actually generates the Serial RapidIO error/write-port output, which is combined with the error interrupt to generate the error/port-write interrupt.
5. Software reads the queue full bit (IPWSR[QFI]) and determines that the queue full event caused the interrupt. Many events that can generate this interrupt. Software must read several registers to determine that the interrupt is due to a port-write.
6. Software processes the port-write queue entry to which the port-write base address registers (IPWQBAR) are pointing.
7. Software sets the clear queue bit (IPWMR[CQ]) to reenable the hardware to receive another port-write.
8. Software clears the queue full interrupt bit (IPWSR[QFI]) by setting IPWSR[QFI].

16.5.3 Port-Write Controller Interrupts

The error/port-write interrupt is used by the port-write controller. This interrupt is used to notify the processor that some type of error event has occurred in a RapidIO port, message controller, doorbell controller or port-write controller. There are many events that can generate this interrupt. For example, the error management extensions use this interrupt to notify that error events have occurred. In the port-write controller the following event can generate this interrupt.

- *Queue Full*. This interrupt event is enabled (IPWMR[QFIE]) and a port-write is received and written to memory. The event causing this interrupt is indicated by IPWSR[QFI]. The interrupt is held until the queue is not full and the IPWSR[QFI] bit is cleared by writing a value of 1 to it.
- *Transaction Error*. An internal error response is received and this interrupt event is enabled (IPWMR[EIE]).

16.5.4 Discarding Port-Writes

While the queue full bit is set or a port-write is being written to memory but has not completed, all received port-writes are discarded. When a port-write is discarded for one of these reasons, the controller sets the port write discarded bit (IPWSR[PWD]). Note that the port-write busy bit (IPWSR[PWB]) indicates that a port-write is being written to memory but has not completed.

16.5.5 Transaction Errors

When an internal error occurs while the port-write controller is writing data to local memory, the controller takes the following actions:

1. Sets the transaction error bit (IPWSR[TE]) and enters the error state.
2. Generates the Serial RapidIO error/write-port interrupt if IPWMR[EIE] is set.

16.5.6 Software Error Handling

When an error occurs and the Serial RapidIO error/write-port interrupt is generated, software takes the following actions:

1. Determines the cause of the interrupt and processes the error.
2. Polls IPWSR[PWB] to verify that the port-write controller has stopped operation.
3. Disables the port-write controller by clearing IPWMR[PWE].
4. Clears the error by writing a 1 to the corresponding status bit (IPWSR[TE]).
5. Disables, reinitializes, and reenables the port-write unit before another maintenance port-write can be received.

When an error occurs and the Serial RapidIO error/write-port interrupt is not enabled, software takes the following actions:

1. Polls the status bits (IPWSR[TE]) to determines that an error has occurred.
2. Polls IPWSR[PWB] to verify that the port-write controller has stopped operation.
3. Clears IPWMR[PWE] to disable the port-write controller.
4. Clears the error by writing a 1 to the corresponding status bit (IPWSR[TE]).

16.5.7 Hardware Error Handling

Table 16-41 describes the possible hardware error conditions and what occurs when they are detected. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. When an error is detected, no additional error checking beyond the current level is performed. Port-writes are processed in a pipeline. The first error detected in the processing pipeline updates the error management extension registers. These error condition checks are provided by the messaging unit. These check are in addition to the error condition checks provided by the RapidIO port, as described in **Section 16.2.10**, *Errors and Error Handling*, on page 16-25.

Table 16-41. Inbound Port-Write Hardware Errors

Error	Description
Reserved ftype[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]. **Queue Entry Written in local memory:** No **Response status:** No response. **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Reserved tt encoding[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set. **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Large transport size in small transport mode or small transport size in large transport mode[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set. **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Queue Entry Written in local memory:** No **Response status:** No response. **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded. An error or illegal transaction target error response is not generated.

Table 16-41. Inbound Port-Write Hardware Errors

Error	Description
Illegal destination ID[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITTE] is set. **Status bit set**: Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
An incorrect wr_size encoding (not 4, 8, 16, 24, 32, 40, 48, 56, or 64 bytes), payload size greater than the size defined by wr_size, or not 64-bit aligned when the size is not 4 bytes.[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] set. **Status bit set**: Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
wr_size value reserved[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] set. **Status bit set**: Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Inbound maintenance port-write received and inbound maintenance port-writes are not supported as indicated by DOCAR[PW][1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]. **Queue Entry Written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Not an error. An inbound port-write packet with a RapidIO priority of 3	**Error checking level:** 2 **Interrupt generated:** **Status bit set**: **Queue Entry Written in local memory:** Yes **Response status:** No response **Logical/Transport Layer Capture Register:** Inbound port write considered priority 2 by the inbound port write controller since response from memory is required at priority 3. **Comments:**
Port-write controller disabled and port-write received	**Error checking level:** 2 **Interrupt generated:** No **Status bit set**: None **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** **Comments:** Packet is ignored and discarded.

Table 16-41. Inbound Port-Write Hardware Errors

Error	Description
Port-write controller enabled but in the error state and port-write received	**Error checking level:** 2 **Interrupt generated:** No **Status bit set**: None **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** **Comments:** Packet is ignored and discarded.
Internal error during the write of the port-write queue entry to memory	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if IPWMR[EIE] is set. **Status bit set**: Transaction error in the Port-write status register (IPWSR[TE]). Port-write Failed in the Port-write and Doorbell CSR (PWDCSR[PFA]). **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** **Comments:** Port-write controller stops after the current port-write operation completes.

Notes: **1.** These error types are actually detected in the RapidIO port, not in the port-write controller.

2. In small transport size configuration using the packet, the following allocations are made:

- LTLACCSR[XA] gets the extended address (packet bits 78–79).
- LTLACCSR[A] gets the address (packet bits 48–76).
- LTLDIDCCSR[MDID] gets 0.
- LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23).
- LTLDIDCCSR[MSID] gets 0.
- LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31).
- LTLCCCSR[FT] gets the ftype (packet bits 12–15).
- LTLCCCSR[TT] gets the ttype (packet bits 32–35).
- LTLCCCSR[MI] gets 0.

In large transport size configuration using the packet, the following allocations are made:

- LTLACCSR[XA] gets the extended address (packet bits 94–95).
- LTLACCSR[A] gets the address (packet bits 64– 92).
- LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23).
- LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31).
- LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32–39).
- LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47).
- LTLCCCSR[FT] gets the ftype (packet bits 12–15).
- LTLCCCSR[TT] gets the ttype (packet bits 48 –51).
- LTLCCCSR[MI] gets 0.

Table 16-42 lists the port-write programming errors.

Table 16-42. Inbound Port-Write Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Port-write queue entry written to non-existent memory	No	No	When a write to memory occurs, the memory controller causes its own interrupt and update its own capture registers. An internal error response is returned. When the port-write controller receives the error response it sets the transaction error bit (IPWSR[TE]) and enters the error state.

16.5.8 Disabling and Enabling the Port-Write Controller

When the port-write controller is disabled by clearing IPWMR[PWE], the following actions are taken:

1. Queue full clears (IPWSR[QF]).
2. Port-write busy (IPWSR[PWB]) clears after a pending port-write queue entry write completes.

Before the port-write controller is reenabled (IPWMR[PWE]), the port-write busy bit (IPWSR[PWB]) must be cleared.

16.5.9 RapidIO Message Passing Logical Specification Registers

The port-write and doorbell command and status register (PWDCSR) includes several port-write controller status bits. These read-only status bits only indicate the state of the port-write controller. See **Section 16.6.9**, *Port Write and Doorbell Command and Status Register (PWDCSR)*, on page 16-112 for details.

- *Available (PWDCSR[PA])*. Indicates that the port-write controller is enabled (IPWnMR[PWE]), the only port-write queue entry is available to be written (IPWnSR[QF]) = 0 and the port-write controller is not in the internal error state (IPWnSR[TE] = 0)
- *Full (PWDCSR[PFU])*. This bit reflects the state of the queue full status bit (IPWnSR[QF])
- *Empty (PWDCSR[PEM])*. This bit always a 1 since a port-write cannot be generated
- *Busy (PWDCSR[PB])*. This bit reflects the state of the busy bit IPWnSR[PWB]
- *Failed (PWDCSR[PFA])*. This bit reflects the state of the transaction error status bit IPWnSR[TE]
- *Error (PWDCSR[PE])*. This bit is always a 0

16.6 RapidIO Programming Model

The RapidIO registers include:

- Architectural Registers:
  - Device Identity Capability Register (DIDCAR), **page 16-104**
  - Device Information Capability Register (DICAR), **page 16-105**
  - Assembly Identity Capability Register (AIDCAR), **page 16-105**
  - Assembly Information Capability Register (AICAR), **page 16-106**
  - Processing Element Features Capability Register (PEFCAR), **page 16-106**
  - Source Operations Capability Register (SOCAR), **page 16-108**
  - Destination Operations Capability Register (DOCAR), **page 16-109**
  - Mailbox Command and Status Register (MCSR), **page 16-111**
  - Port -Write and Doorbell Command and Status Register (PWDCSR), **page 16-112**
  - Processing Element Logical Layer Control Command and Status Register (PELLCCSR), **page 16-114**
  - Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR), **page 16-115**
  - Base Device ID Command and Status Register (BDIDCSR), **page 16-116**
  - Host Base Device ID Lock Command and Status Register (HBDIDLCSR), **page 16-117**
  - Component Tag Command and Status Register (CTCSR), **page 16-118**
- Extended Features Space: 1x/4x LP-Serial
  - Port Maintenance Block Header 0 (PMBH0), **page 16-119**
  - Port Link Time-Out Control Command and Status Register (PLTOCCSR), **page 16-120**
  - Port Response Time-out Control Command and Status Register (PRTOCCSR), **page 16-121**
  - Port General Control Command and Status Register (PGCCSR), **page 16-122**
  - Port 0 Local ackID Status Command and Status Register (P0LASCR), **page 16-125**
  - Port 0 Error and Status Command and Status Register (P0ESCSR), **page 16-126**
  - Port 0 Control Command and Status Register (P0CCSR), **page 16-128**
- Extended Features Space: Error Reporting, Logical
  - Error Reporting Block Header (ERBH), **page 16-130**
  - Logical/Transport Layer Error Detect Command and Status Register (LTLEDCSR), **page 16-130**
  - Logical/Transport Layer Error Enable Command and Status Register (LTLEECSR), **page 16-132**
  - Logical/Transport Layer Address Capture Command and Status Register (LTLACCSR), **page 16-133**

  - Logical/Transport Layer Device ID Capture Command and Status Register (LTLDIDCCSR), **page 16-134**
  - Logical/Transport Layer Control Capture Command and Status Register (LTLCCCSR), **page 16-135**
- Extended Features Space: Error Reporting, Physical
  - Port 0 Error Detect Command and Status Register (P0EDCSR), **page 16-136**
  - Port 0 Error Rate Enable Command and Status Register (P0ERECSR), **page 16-137**
  - Port 0 Error Capture Attributes Command and Status Register (P0ECACSR), **page 16-138**
  - Port 0 Packet/Control Symbol Error Capture Command and Status Register (P0PCSECCSR0), **page 16-140**
  - Port 0 Packet Error Capture Command and Status Register 1 (P0PECCSR1), **page 16-141**
  - Port 0 Packet Error Capture Command and Status Register 2 (P0PECCSR2), **page 16-142**
  - Port 0 Packet Error Capture Command and Status Register 3 (P0PECCSR3), **page 16-143**
  - Port 0 Error Rate Command and Status Register (P0ERCSR), **page 16-144**
  - Port 0 Error Rate Threshold Command and Status Register (P0ERTCSR), **page 16-145**
- Implementation Space: General
  - Logical Layer Configuration Register (LLCR); **page 16-146**
  - Error /Port-write Interrupt Status Register (EPWISR), **page 16-147**
  - Logical Retry Error Threshold Configuration Register (LRETCR), **page 16-148**
  - Physical Retry Error Threshold Configuration Register (PRETCR), **page 16-149**
  - Port 0 Alternate Device ID Command and Status Register (P0ADIDCSR), **page 16-150**
  - Port 0 Accept-All Configuration Register (P0AACR), **page 16-151**
  - Port 0 Implementation Error Command and Status Register (P0IECSR), **page 16-153**
  - Port 0 Physical Configuration Register (P0PCR), **page 16-154**
  - Port 0 Serial Link Command and Status Register (P0SLCSR), **page 16-155**
  - Port 0 Serial Link Error Injection Configuration Register (P0SLEICR), **page 16-156**
- Implementation Space: Revision Control
  - IP Block Revision Register 1 (IPBRR1); **page 16-157**
  - IP Block Revision Register 2 (IPBRR2), **page 16-157**
- Outbound ATMU
  - Port 0 RapidIO Outbound Window Translation Address Register (P0ROWTARx), **page 16-158**
  - Port 0 RapidIO Outbound Window Translation Extended Address Register (P0ROWTEARx), **page 16-159**
  - Port 0 RapidIO Outbound Window Attributes Register (P0ROWARx), **page 16-160**

- Port 0 RapidIO Outbound Window Base Address Register (P0ROWBARx), **page 16-162**
- Port 0 RapidIO Outbound Window Segment 1–3 Registers (P0ROWS1Rx), **page 16-163**

- Inbound ATMU
  - Port 0 RapidIO Inbound Window Translation Address Registers (P0RIWTARx); **page 16-164**
  - Port 0 RapidIO Inbound Window Base Address Registers (P0RIWBARx), **page 16-165**
  - Port 0 RapidIO Inbound Window Attributes Registers (P0RIWARx), **page 16-166**
- Outbound Message Unit Registers
  - Outbound Message x Mode Registers (OMxMR); **page 16-167**
  - Outbound Message x Status Registers (OMxSR), **page 16-169**
  - Outbound Message x Descriptor Queue Dequeue Pointer Address Registers (OMxDQDPAR), **page 16-171**
  - Outbound Message x Descriptor Queue Enqueue Pointer Address Registers (OMxDQEPAR), **page 16-176**
  - Outbound Message x Source Address Registers (OMxSAR), **page 16-172**
  - Outbound Message x Destination Port Registers (OMxDPR), **page 16-173**
  - Outbound Message x Destination Attributes Registers (OMxDATR), **page 16-174**
  - Outbound Message x Double-Word Count Registers (OMxDCR), **page 16-175**
  - Outbound Message x Retry Error Threshold Configuration Registers (OMxRETCR), **page 16-177**
  - Outbound Message x Multicast Group Registers (OMxMGR), **page 16-178**
  - Outbound Message x Multicast List Registers (OMxMLR), **page 16-179**
- Inbound Message Unit Registers
  - Inbound Message x Mode Registers (IMxMR), **page 16-180**
  - Inbound Message x Status Registers (IMxSR), **page 16-182**
  - Inbound Message x Frame Queue Dequeue Pointer Address Registers (IMxFQDPAR), **page 16-184**
  - Inbound Message x Frame Queue Enqueue Pointer Address Registers (IMxFQEPAR), **page 16-185**
  - Inbound Message x Maximum Interrupt Report Interval Registers (IMxMIRIR), **page 16-186**
- Outbound Doorbell Unit Registers
  - Outbound Doorbell Mode Register (ODMR), **page 16-187**
  - Outbound Doorbell Status Register (ODSR), **page 16-188**
  - Outbound Doorbell Destination Port Register (ODDPR), **page 16-189**
  - Outbound Doorbell Destination Attributes Register (ODDATR), **page 16-190**

— Outbound Doorbell Retry Error Threshold Configuration Register (ODRETCR), **page 16-191**

- Inbound Doorbell Unit Registers
  - — Inbound Doorbell Mode Register (IDMR), **page 16-192**
  - — Inbound Doorbell Status Register (IDSR), **page 16-194**
  - — Inbound Doorbell Queue Dequeue Pointer Address Register (IDQDPAR), **page 16-195**
  - — Inbound Doorbell Queue Enqueue Pointer Address Register (IDQEPAR), **page 16-196**
  - — Inbound Doorbell Maximum Interrupt Report Interval Register (IDMIRIR), **page 16-197**
- Port-Write Registers
  - — Inbound Port-Write Mode Register (IPWMR); **page 16-198**
  - — Inbound Port-Write Status Register (IPWSR), **page 16-199**
  - — Inbound Port-Write Queue Base Address Register (IPWQBAR), **page 16-200**

Note: The base address for the RapidIO registers is: 0xFFF80000.

16.6.1 Device Identity Capability Register (DIDCAR)

DIDCAR Device Identity Capability Register Offset 0x00000

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

Table 16-43. DIDCAR Field Descriptions

Bits	Reset	Description
DI 31–16	0x1807	**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 MSC8144E = 0x1807.
DVI 15–0	0x0002	**Device Vendor Identity** Identifies the vendor that manufactures the device containing the processing element. A value for DVI is uniquely assigned to a device vendor by the registration authority of the RapidIO Trade Association. The value for Freescale Semiconductor, Inc. is 0x0002.

16.6.2 Device Information Capability Register (DICAR)

DICAR Device Information Capability Register Offset 0x00004

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

Table 16-44. DICAR Field Descriptions

Bit	Reset	Description
DR 31–0	0x00000000	**Device Revision** Identifies the revision level of the device. The value for DR is assigned and managed by the vendor specified by DIDCAR[DVI].

16.6.3 Assembly Identity Capability Register (AIDCAR)

AIDCAR Assembly Identity Capability Register Offset 0x00008

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

Table 16-45. AIDCAR Field Descriptions

Bit	Name	Description
AI 31–16	0x0000	**Assembly Identity** Uniquely identifies the type of assembly from the vendor specified by AVI. The values for AI are assigned and managed by the respective vendor.
AVI 15–0	0x0000	**Assembly Vendor Identity** Identifies the vendor that manufactures the assembly or subsystem containing the device. A value for AVI is uniquely assigned to an assembly vendor by the registration authority of the RapidIO Trade Association.

16.6.4 Assembly Information Capability Register (AICAR)

AICAR Assembly Information Capability Register Offset 0x0000C

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

AICAR contains additional information on the assembly and the pointer to the first entry in the extended features list.

Table 16-46. AICAR Field Descriptions

Bit	Reset	Description
AR 31–16	0x0000	**Assembly Revision**
EFP 15–0	0x0100	**Extended Features Pointer**

16.6.5 Processing Element Features Capability Register (PEFCAR)

PEFCAR Processing Element Features Capability Register Offset 0x00010

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	BR	MEM	PROC	SW	—				MB				DB	—		
TYPE	R															
RESET	0	1	1	0	0	0	0	0	1	1	1	1	1	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—											CTLS	EF	EAS		
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1

PEFCAR identifies the major functionality provided by the processing element.

Table 16-47. PEFCAR Field Descriptions

Bit	Reset	Description	Settings
BR 31	0	**Bridge** Specifies whether the MSC8144E can bridge to another interface.	
MEM 30	1	**Memory** Specifies whether the MSC8144E has physically addressable local address space and can be accessed as an endpoint through non-maintenance operations.	
PROC 29	1	**Processor** Specifies whether the MSC8144E contains a local processor that executes code.	
SW 28	0	**Switch** The MSC8144E does not bridge to another external RapidIO interface. (SW=0)	
— 27–24	0	Reserved. Write to zero for future compatibility.	
MB 23–20	0b1111	**Mailbox** Specifies the inbound mailboxes supported by the MSC8144E.	1000 Mailbox 0, supported by MSC8144E. 0100 Mailbox 1, supported by MSC8144E. 0010 Mailbox 2, supported by MSC8144E. 0001 Mailbox 3, supported by MSC8144E. 1111 Mailboxes 0–3.
DB 19	1	**Doorbell** Specifies whether the RapidIO Controller supports inbound Doorbells.	
— 18–5	0	Reserved. Write to zero for future compatibility.	
CTLS 4	1	**Common Transport Large Systems** Specifies whether the RapidIO Controller supports large systems. This is configured at power-up through the reset configuration word.	0 Only common transport small systems (up to 256 devices). 1 Large systems up to 65,536 devices.
EF 3	1	**Extended Features** Specifies whether the extended features pointer is valid.	
EAS 2–0	0b001	**Extended Address Support** Indicates the number of address bits supported by the RapidIO controller both as a source and target of an operation.	000 Reserved. 010 Reserved. 001 34-bit addresses.

16.6.6 Source Operations Capability Register (SOCAR)

SOCAR Source Operations Capability Register 0x00018

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	R	IR	RO	DCI	C	F	IOR	ICI	TIE	TIES	—					
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	NR	NW	SW	NWR	M	D	—	ATS	AI	AD	AS	AC	—	PW	—	
TYPE	R															
RESET	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

SOCAR defines the set of RapidIO logical operations that can be issued by the MSC8144E.

Table 16-48. SOCAR Field Descriptions

Bit	Reset	Description
R 31	0	**Read Operation** MSC8144E does not support.
IR 30	0	**IRead Operation** MSC8144E does not support.
RO 29	0	**Read-to-Own Operation** MSC8144E does not support.
DCI 28	0	**Data Cache Invalidate Operation** MSC8144E does not support.
C 27	0	**Castout Operation** MSC8144E does not support.
F 26	0	**Flush Operation** MSC8144E does not support.
IOR 25	0	**I/O-Read Operation** MSC8144E does not support.
ICI 24	0	**Instruction Cache Invalidate Operation** MSC8144E does not support.
TIE 23	0	**TLBIE Operation** MSC8144E does not support.
TIES 22	0	**TLBSYNC Operation** MSC8144E does not support.
— 21–16	0	Reserved. Write to zero for future compatibility.
NR 15	1	**NREAD Operation** MSC8144E supports operation.
NW 14	1	**NWRITE Operation** MSC8144E supports operation.
SW 13	1	**SWRITE Operation** MSC8144E supports operation.
NWR 12	1	**NWRITE_R Operation** MSC8144E supports operation.
M 11	1	**Message Operation**
D 10	1	**Doorbell Operation**

Table 16-48. SOCAR Field Descriptions (Continued)

Bit	Reset	Description
— 9	—	Reserved. Write to zero for future compatibility.
ATS 8	0	**Atomic-Test-and-Swap Operation** MSC8144E does not support.
AI 7	0	**Atomic-Inc Operation** MSC8144E does not support.
AD 6	0	**Atomic-Dec Operation** MSC8144E does not support.
AS 5	0	**Atomic-Set Operation** MSC8144E does not support.
AC 4	0	**Atomic-Clr Operation** MSC8144E does not support.
— 3	0	Reserved. Write to zero for future compatibility.
PW 2	0	**Port-Write Operation** MSC8144E does not support.
— 1–0	0	Reserved. Write to zero for future compatibility.

16.6.7 Destination Operations Capability Register (DOCAR)

DOCAR Destination Operations Capability Register 0x0001C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	R	IR	RO	DCI	C	F	IOR	ICI	TIE	TIES	—					
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	NR	NW	SW	NWR	M	D	—	ATS	AI	AD	AS	AC	—	PW	—	
TYPE	R															
RESET	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0

DOCAR defines the set of RapidIO I/O operations that the MSC8144E can support as a target.

Table 16-49. DOCAR Field Descriptions

Bit	Reset	Description
R 31	0	**Read Operation** MSC8144E does not support.
IR 30	0	**IRead Operation** MSC8144E does not support.
RO 29	0	**Read-to-Own Operation** MSC8144E does not support.
DCI 28	0	**Data Cache Invalidate Operation** MSC8144E does not support.
C 27	0	**Castout Operation** MSC8144E does not support.
F 26	0	**Flush Operation** MSC8144E does not support.

Table 16-49. DOCAR Field Descriptions (Continued)

Bit	Reset	Description
IOR 25	0	**I/O-Read Operation** MSC8144E does not support.
ICI 24	0	**Instruction Cache Invalidate Operation** MSC8144E does not support.
TIE 23	0	**TLBIE Operation** MSC8144E does not support.
TIES 22	0	**TLBSYNC Operation** MSC8144E does not support.
— 21–16	0	Reserved. Write to zero for future compatibility.
NR 15	1	**Nread Operation** Supported.
NW 14	1	**Nwrite Operation** Supported.
SW 13	1	**Swrite Operation** Supported.
NWR 12	1	**Nwrite_R Operation** Supported.
M 11	1	**Message Operation** Supported.
D 10	1	**Doorbell Operation** Supported.
— 9	—	Reserved. Write to zero for future compatibility.
ATS 8	0	**Atomic-Test-and-Swap Operation** MSC8144E does not support.
AI 7	0	**Atomic-Inc Operation** MSC8144E does not support.
AD 6	0	**Atomic-Dec Operation** MSC8144E does not support.
AS 5	0	**Atomic-Set Operation** MSC8144E does not support.
AC 4	0	**Atomic-Clr Operation** MSC8144E does not support.
— 3	0	Reserved. Write to zero for future compatibility.
PW 2	1	**Port-Write Operation** Supported.
— 1–0	0	Reserved. Write to zero for future compatibility.

16.6.8 Mailbox Command and Status Register (MCSR)

MCSR Mailbox Command and Status Register 0x00040

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	A0	FU0	EM0	B0	FA0	ERR0	—		A1	FU1	EM1	B1	FA1	ERR1	—	
TYPE	R															
RESET	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MCSR reflects the status of the RapidIO message controllers.

Table 16-50. MCSR Field Definitions

Bits	Reset	Description	Settings
A0 31	0	**Available** Specifies whether message controller 0 is initialized and ready to accept messages. When this bit is cleared, all incoming message transactions to message controller 0 return error responses.	0 Not ready. 1 Ready.
FU0 30	0	**Full** Specifies whether message controller 0 is full. When FU0 is set, new messages to message controller 0 are retried.	0 Not full. 1 Full.
EM0 29	1	**Empty** Specifies whether message controller 0 contains outstanding messages.	0 Contains outstanding messages. 1 Contains no outstanding messages.
B0 28	0	**Busy** Specifies whether message controller 0 is busy processing a message. When this bit is set, new message operations to message controller 0 return retry responses.	0 Not busy. 1 Busy.
FA0 27	0	**Failure** Specifies whether message controller 0 has encountered an internal error and is awaiting assistance. When this bit is set, all incoming message transactions to message controller 0 return error responses.	0 No internal error. 1 Internal fault or error condition encountered.
ERR0 26	0	**Error** This field always returns a value of 0.	
— 25–24	0	Reserved. Write to zero for future compatibility.	
A1 23	0	**Available** Specifies whether message controller 1 is initialized and ready to accept messages. When this bit is cleared, all incoming message transactions to message controller 1 return error responses.	0 Not ready. 1 Ready.
FU1 22	0	**Full** Specifies whether message controller 1 is full. When FU0 is set, new messages to message controller 1 are retried.	0 Not full. 1 Full.

Table 16-50. MCSR Field Definitions (Continued)

Bits	Reset	Description	Settings
EM1 21	1	**Empty** Specifies whether message controller 1 contains outstanding messages.	0 Contains outstanding messages. 1 Contains no outstanding messages.
B1 20	0	**Busy** Specifies whether message controller 1 is busy processing a message. When this bit is set, new message operations to message controller 1 return retry responses.	0 Not busy. 1 Busy.
FA1 19	0	**Failure** Specifies whether message controller 1 has encountered an internal error and is awaiting assistance. When this bit is set, all incoming message transactions to message controller 1 return error responses.	0 No internal error. 1 Internal fault or error condition encountered.
ERR1 18	0	**Error** This field always returns a value of 0.	
— 17–0	0	Reserved. Write to zero for future compatibility.	

16.6.9 Port Write and Doorbell Command and Status Register (PWDCSR)

PWDCSR Port-Write and Doorbell Command and Status Register Offset 0x00044

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	A	FU	EM	B	FA	ERR	—									
TYPE	R															
RESET	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—								PA	PFU	PEM	PB	PFA	PE	—	
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

PWDCSR reflects the status of the RapidIO doorbell and port-write hardware. For details, refer to the *RapidIO Interconnect Specification, Revision 1.2* in sections “Doorbell CSR” and “Write Port CSR.”:

Table 16-51. PWDCSR Field Descriptions

Bits	Reset	Description	Settings
A 31	0	**Available** Specifies whether the doorbell unit is available and ready to accept doorbell messages. When this bit is cleared, all incoming doorbell transactions return error responses.	0 Not ready. 1 Ready.
FU 30	0	**Full** Specifies whether the doorbell unit is full. When this bit is set, new doorbell messages are retried.	0 Not full. 1 Full.
EM 29	1	**Empty** Specifies whether the doorbell unit has outstanding doorbell messages.	0 Outstanding doorbell messages. 1 No outstanding doorbell messages.

Table 16-51. PWDCSR Field Descriptions (Continued)

Bits	Reset	Description	Settings
B 28	0	**Busy** Specifies whether the doorbell unit is busy processing a doorbell message. When this bit is set, incoming transactions are not retried.	0 Not busy. 1 Busy.
FA 27	0	**Failure** Specifies whether the doorbell unit has encountered an internal fault or error condition and is awaiting assistance. When this bit is set, all incoming doorbell transactions return error responses.	0 No internal error. 1 Internal fault or error condition encountered.
ERR 26	0	**Error** This field always returns a value of 0.	
— 25–8	0	Reserved. Write to zero for future compatibility.	
PA 7	0	**Port-Write Unit Available** Specifies whether the port-write unit is available and ready to accept a port-write packet. When this bit is cleared, all incoming port-write packets are discarded.	0 Not available. 1 Ready.
PFU 6	0	**Port-Write Unit Full** Specifies whether the port-write unit is full. When this bit is set, all incoming port-write packets are discarded.	0 Not full. 1 Full.
PEM 5	1	**Port-Write Unit Empty** This field always returns a value of 1.	
PB 4	0	**Port-write Unit Busy** Specifies whether the port-write unit is busy processing a port-write packet. When this bit is set, incoming port-write packets are discarded.	0 Not busy. 1 Busy.
PFA 3	0	**Failure** Specifies whether the port-write unit has encountered an internal fault or error condition and is awaiting assistance. When this bit is set, all incoming port-write transactions are discarded.	0 No internal error. 1 Internal fault or error condition encountered.
PE 22	0	**Error** This field always returns a value of 0.	
— 1–0	0	Reserved. Write to zero for future compatibility.	

16.6.10 Processing Element Logical Layer Control Command and Status Register (PELLCCSR)

PELLCCSR Processing Element Logical Layer Control Command and Status Register — Offset 0x0004C

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

PELLCCSR controls the extended addressing abilities. RapidIO endpoint supports only 34-bit addressing.

Table 16-52. PELLCCSR Field Descriptions

Bit	Reset	Description
— 31–3	0	Reserved. Write to zero for future compatibility.
EAC 2–0	0b001	**Extended Addressing Control** The read-only value of 0b001 specifies 34-bit addresses.

16.6.11 Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)

LCSBA1CSR Local Configuration Space Base Address 1 Command and Status Register — Offset 0x0005C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—	LCSBA														—
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	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 most significant bits of the local physical address double-word offset for the MSC8144E configuration register space allowing the configuration register space to be physically mapped in the processing element. This register allows configuration and maintenance of an MSC8144E 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 16-53. LCSBA1CSR Field Descriptions

Bit	Reset	Description
— 31	0	Reserved. Write to zero for future compatibility.
LCSBA 30–17	0x0	**Local Configuration Space Base Address** These bits correspond to the highest 14 bits of the 34-bit RapidIO address space. This register is only used to access the device CCSR memory space using RapidIO read and write accesses. To access the CCSR space in the MSC8144E, use 0x3FFC0000.
— 16–0	0	Reserved. Write to zero for future compatibility.

16.6.12 Base Device ID Command and Status Register (BDIDCSR)

BDIDCSR Base Device ID Command and Status Register Offset 0x00060

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

BDIDCSR contains the base device ID values for the processing element.

Table 16-54. 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, BDID = 0x00. 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, LBDID = 0x0000. If the RapidIO controller is configured as an agent, LBDID = 0xFFFF.

16.6.13 Host Base Device ID Lock Command and Status Register (HBDIDLCSR)

HBDIDLCSR Host Base Device ID Lock Command and Status Register — Offset 0x00068

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

HBDIDLCSR contains the base device ID value for the processing element in the system that initializes this device. The HBDID field is a write-once/resettable field with a lock function. When HBDID is written, all subsequent writes to the field are ignored, except when the value written matches the value in the field. Then, the register is reinitialized to 0xFFFF. After HBDID is written, the processing element must read the host base device ID lock CSR to verify that it owns the lock before it attempts to initialize the device.

Table 16-55. HBDIDLCSR Field Descriptions

Bit	Reset	Description
— 31–16	0	Reserved. Write to zero for future compatibility.
HBDID 15–0	0xFFFF	**Host Base Device ID** The host base device ID for the processing element that initializes this device. Only the first write to this field is accepted; all other writes are ignored, except when the value written matches the value contained in the field. In this case, the register is reinitialized to 0xFFFF.

16.6.14 Component Tag Command and Status Register (CTCSR)

CTCSR Component Tag Command and Status Register Offset 0x0006C

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

CTCSR contains a component tag value for the processing element that software can assign when the device is initialized. It is unused internally in RapidIO Endpoint.

Table 16-56. CTCSR Field Descriptions

Bit	Reset	Description
CT 31–0	0	**Component Tag**

16.6.15 Port Maintenance Block Header 0 (PMBH0)

PMBH0 Port Maintenance Block Header 0 Offset 0x00100

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

PMBH0 contains a pointer (EFPTR) to the next extended features space, error management block, extended features block (EFBLK), and the EFID that identifies it as the generic endpoint port maintenance block header. While registers defined by software-assisted error recovery are supported, software-assisted error recovery is not. Therefore, the RapidIO Endpoint is defined here as not supporting software-assisted error recovery.

Table 16-57. PMBH0 Field Descriptions

Bit	Reset	Description
EFPTR 31–15	0x0600	**Extended Features Pointer**
EFID 16–31	0x0001	**Extended Features ID**

16.6.16 Port Link Time-Out Control Command and Status Register (PLTOCCSR)

PLTOCCSR Port Link Time-Out Control Command and Status Register — Offset 0x00120

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

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 16-58. 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	0x00	Reserved. Write to 0x01 for future compatibility.

16.6.17 Port Response Time-Out Control Command and Status Register (PRTOCCSR)

PRTOCCSR Port Response Time-Out Control Command and Status Register — Offset 0x00124

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

PRTOCCSR contains the time-out timer count for all ports. This time-out is for sending a request packet to receive the corresponding response packet.The reset value is the maximum time-out interval and represents between 3 and 5 seconds. This applies to RapidIO Endpoint and the messaging unit.

Table 16-59. PRTOCCSR Field Descriptions

Bit	Reset	Description
TV 31–8	0xFFFFFF	**Time-Out Value** Clearing this field to all zeros disables the response time-out timer. This value is loaded each time the response time-out timer starts.
— 7–0	0	Reserved. Write to zero for future compatibility.

16.6.18 Port General Control Command and Status Register (PGCCSR)

PGCCSR Port General Control Command and Status Register Offset 0x0013C

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

PGCCSR contains control register bits for the RapidIO interface.

Note: The user must initialize the value of M to 1. Otherwise, no outbound transactions are initiated by the MSC8144E, including messages and doorbells.

Table 16-60. PGCCSR Field Descriptions

Bit	Reset	Description	Settings
H 31	0	**Host** Determines the host/agent configuration for the device. Notice that by default H = 0.	0 Agent device. 1 Host device.
M 30	1	**Initiator** The value of this bit is identical to that of GCCSR[H], which is assigned by power-on reset configuration signals. Setting 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	0	**Discovered** The value of this bit is identical to that of GCCSR[H], which is assigned by power-on reset configuration signals.	0 Device not discovered by system host. 1 Device is discovered by system host.
— 28–0	0	Reserved. Write to zero for future compatibility.	

16.6.19 Port 0 Link Maintenance Request Command and Status Register (P0LMREQCSR)

P0LMREQCSR Port 0 Link Maintenance Request Command and Status Register — Offset 0x00140

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

The port 0 link maintenance request command and status register (P0LMREQCSR), 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 16-61** lists P0LMREQCSR fields.

Table 16-61. P0LMREQCSR Field Descriptions

Bit	Reset	Description
— 31–3	0	Reserved. Write as zero for future compatibility.
C 2–0	0	**Link Request Command** Contains the link request command to send. If read, this field returns the last written value. If written with a value other than 0x011 (reset device) or 0b100 (input status), the resulting operation is undefined. All other values are reserved in the RapidIO specification.

16.6.20 Port 0 Link Maintenance Response Command and Status Register (P0LMRESPCSR)

P0LMRESPCSR Port 0 Link Maintenance Response Command and Status Register — Offset 0x00144

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

The port 0 link maintenance response command and status register (P0LMRESPCSR), 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 16-62** lists P0LMRESPCSR fields.

Table 16-62. P0LMRESPCSR Field Descriptions

Bit	Reset	Description	Settings
RV 31	0	**Response Valid** This bit indicates one of two conditions: • If the link-request causes a link-response, a set bit indicates that the link-response was received and the status fields are valid. • If the link-request did not cause a link-response, a set bit indicates that the link-request was transmitted. **Note:** This bit clears on read.	0 Link response not received, link response not valid, or link request was not transmitted. 1 Link response received with valid status bits or link request was transmitted.
— 30–10	0	Reserved. Write as zero for future compatibility.	
AS 9–5	0	**AckID_Status** This field holds the AckID status field from the link response.	
LS 4–0	0	**Link Status** This field holds the link status field from the link response.	

16.6.21 Port 0 Local ackID Command and Status Register (P0LASCR)

P0LASCSR Port 0 Local ackID Command and Status Register Offset 0x00148

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

The port *0* local ackID status command and status register (P*0*LASCSR) 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 16-63** lists P*0*LASCSR fields.

Table 16-63. P0LASCSR Field Descriptions

Bit	Reset	Description
— 31–29	0	Reserved. Write as zero for future compatibility.
IA 28–24	0	**Input ackID** The value of the next expected Input port ackID.
23–13	0	Reserved. Write as zero for future compatibility.
OUTA 12–8	0	**Outstanding Port Unacknowledged ackID Status** Next expected acknowledge control symbol ackID field that indicates the ackID value expected in the next received acknowledge control symbol. Note that this value is read only even though RapidIO specification allows for it to be writable.
— 7–5	0	Reserved. Write as zero for future compatibility.
OBA 4–0	0	**Outbound ackID Output Port Next Transmitted ackID Value** This can be written by software but only if there are no outstanding unacknowledged packets. If there are, a newly-written value will be ignored

16.6.22 Port 0 Error and Status Command and Status Register (P0ESCSR)

P0ESCSR Port 0 Error Command and Status Register Offset 0x00158

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

P0ESCSR is accessed when a local master or an external device needs to examine the port error and status information.

Table 16-64. P0ESCSR 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 P0CCSR[DPE] (drop packet enable) is set and P0CCSR[SPF] (stop on port failed) is set. • It is received while P0PCR[OBDEN] (output buffer drain enable) is set. • The link-partner does not accept it while the P0ERCSR[ERFTT] (error rate failed threshold trigger) is met or exceeded, P0CCSR[DPE] is set, and P0CCSR[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 16-64. P0ESCSR 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 CCSR[SPF] is set. That is, reaching the failed threshold has caused the output port to stop transmitting packets.
PO 1	0	**Ports Operating** The input and output ports are initialized and the port is exchanging error-free control symbols with the attached device. Read only.
PU 0	1	**Ports Uninitialized** Input and output ports are not initialized. This bit and PO are mutually exclusive. Read only.

16.6.23 Port 0 Control Command and Status Register (P0CCSR)

P0CCSR Port 0 Control Command and Status Register Offset 0x0015C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PW		IPW			PWO			PD	OPE	IPE	ECD	MEP	—		
TYPE	R					R/W						R				
RESET	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—												SPF	DPE	PL	PT
TYPE	R												R/W			R
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

P0CCSR contains control register bits for the RapidIO port. This register is for serial implementation only.

Table 16-65. P0CCSR Field Descriptions

Bit	Reset	Description	Settings
PW 31–30	0b01	**Port Width** Specifies the hardware width of the port. Read only.	00 Single-lane port. 01 Four-lane port (the default). 10–11 Reserved.
IPW 29–27	0b010	Initialized Port Width Specifies the width of the ports after they are initialized. Read only. If the port degrades to 1x, the value changes to 0b000.	000 Single-lane port, lane 0. 001 Reserved. 010 Four-lane port (the default). 011–111 Reserved.
PWO 26–24	0b000	**Port Width Override** Soft port configuration to override the hardware size. Change PWO only when the port is uninitialized. First disable the RapidIO port. Then change PWO to any valid value. Finally, re-enable the RapidIO port.	000 No override (the default). 001 Reserved. 010 Force single lane, lane 0. 011–111 Reserved.
PD 23	0	**Port Disable** When this bit is set, the port does not accept or transmit any transaction. The output will congest is packets are sent to a disabled port.	0 Port receiver/drivers are enabled. 1 Port receiver/drivers are disabled and are unable to receive or transmit.
OPE 22	1	**Output Port Transmit Enable** Specifies whether the port is enabled to issue packets. When OPE is cleared, the port routes or responds to I/O logical maintenance packets. Control symbols are not affected and are sent normally. The initial value of OPE is read from configuration pins.	0 Port is stopped and not enabled to issue packets. 1 Port is enabled to issue packets.

Table 16-65. P0CCSR Field Descriptions (Continued)

Bit	Reset	Description	Settings
IPE 21	1	**Input port Receive Enable** Specifies whether the port is enable to respond to packets. When IPE is cleared, the port can only route or respond to I/O logical maintenance packets. Other packets generate packet-not-accepted control symbols to force an error condition to be signaled by the sending device. Control symbols are not affected and are received and handled normally. The value of IPE must equal the value of OPE for the RapidIO controller to function properly. The initial value of IPE is read from configuration pins.	0 Port is stopped and not enabled to respond to packets. 1 Port is enabled to respond to packets.
ECD 20	0	**Error Checking Disable** Disables all RapidIO transmission error checking. This bit is hard-wired to 0.	0 Error checking and recovery is enabled. 1 Error checking and recovery is disabled.
MEP 19	0	**Multicast Event Participant** This bit is hard-wired to 0. The MSC8144E does not participate in multicast events.	
— 18–4	0	Reserved. Write to zero for future compatibility.	
SPF 3	0	**Stop on Port Failed Encounter Enable** Used with the DPE bit to force certain behavior when the error rate failed threshold is met or exceeded.	0 Stop on port failed disabled. 1 Stop on port failed enabled.
DPE 2	0	**Drop Packet Enable** Used with the SPF bit to force certain behavior when the error rate failed threshold is met or exceeded.	
PL 1	0	**Port Lockout** Stops the port so that is cannot issue or receive packets. The input port can still follow the training procedure and can still send and respond to link requests. All received packets return packet-not-accepted control symbols to force the sending device to signal an error condition.	0 Packets that can be received and issued are controlled by the state of the OPE and IPE bits. 1 The port is stopped and not enabled to issue or receive packets.
PT 0	1	**Port Type** Hard-wired to 1.	0 Reserved. 1 Serial port.

16.6.24 Error Reporting Block Header (ERBH)

ERBH Error Reporting Block Header Offset 0x00600

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

ERBH contains the EFPTR to the next EFBLK. The next EFPTR is 0x0000 since this is the last set of registers in the extended features space. ERBH also contains EFID that identifies this as the error reporting block header.

Table 16-66. ERBH Field Descriptions

Bit	Reset	Description
EFPTR 31–16	0	**Extended Features Pointer** Points to the next EFBLK.
EFID 15–0	0x0007	**Extended Features ID** Identifies this as the error reporting block header.

16.6.25 Logical/Transport Layer Error Detect Command and Status Register (LTLEDCSR)

LTLEDCSR Logical/Transport Layer Error Detect Command and Status Register Offset 0x00608

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IER	MER	—	MFE	ITD	ITTE	MRT	PRT	UR	UT	—					
TYPE	R/W		R	R/W							R					
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—								IACB	OACB	—	RETE	TSE	PTTL	—	
TYPE	R								R/W		R	R/W			R	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LTLEDCSR indicates the error that was detected by the logical or transport logic layer. Software writes this register with all 0s to clear the detected error and unlock the capture registers. Error information that corresponds to two or more different error events is not captured on the same clock cycle. However, one error event such as a corrupted inbound packet can cause multiple bits to be set. The priority of errors is PRT and all other errors. An error that is not enabled sets the detect bit in this register as long as a capture has not yet occurred. You can program fields in this

register to emulate a hardware error during software development. Undefined results occur if fields in this register are set while an actual Logical/Transport Layer error is detected. Note that this register can be written with an invalid combination of bits set, and care should be taken to avoid this.

Table 16-67. LTLEDCSR Field Descriptions

Bit	Name	Description
IER 31	0	**I/O Error Response** Indicates an error response for an I/O logical layer request.
MER 30	0	**Message Error Response** Indicates an error response for an MSG logical layer request. The error is detected and captured in the message unit.
— 29	0	Reserved. Write to zero for future compatibility.
MFE 28	0	**Message Format Error** Indicates a MESSAGE packet data payload with an invalid size or segment. The error is detected and captured in the message unit.
ITD 27	0	**Illegal Transaction Decode** Indicates received illegal fields in the request/response packet for a supported transaction (IO/MSG logical)
ITTE 26	0	**Illegal Transaction Target Error** Indicates a packet containing a destination ID that is not defined for this end point when accept-all is not enabled.
MRT 25	0	**Message Request Time-Out** Indicates that a required message request has not been received within the specified time-out interval. The error is detected and captured in message unit.
PRT 24	0	**Packet Response Time-Out** Indicates that a required response was not received within the specified time-out interval (IO/MSG logical)
UR 23	0	**Unsolicited Response** Indicates that an unsolicited/unexpected response packet was received (IO/MSG logical).
UT 22	0	**Unsupported Transaction** Indicates a received transaction that is not supported in the destination operations CAR.
— 21–8	0	Reserved. Write to zero for future compatibility.
IACB 7	0	**Inbound ATMU Crossed Boundary** Indicates a received transaction that crosses an Inbound ATMU boundary.
OACB 6	0	**Outbound ATMU Crossed Boundary** Indicates a transmitted transaction that crosses an outbound ATMU boundary, a segment boundary, or a subsequent boundary.
— 5	0	Reserved. Write to zero for future compatibility.
RETE 4	0	**Retry Error Threshold Exceeded** Indicates that the allowed number of logical retries (given by LRETCR[RET]) has been exceeded. The message unit also drives RETE when the allowed number of message retries is exceeded.
TSE 3	0	**Transport Size Error** The tt field is not consistent with bit 27 of the processing element features CAR (that is, the tt value is reserved or indicates a common transport system unsupported by this device).
PTTL 2	0	**Packet Time-to-Live Error** Indicates that a packet time-to-live error occurred (that is, a packet could not be successfully transmitted before the packet time-to-live counter expired).
— 1–0	0	Reserved. Write to zero for future compatibility.

16.6.26 Logical/Transport Layer Error Enable Command and Status Register (LTLEECSR)

LTLEECSR Logical/Transport Layer Error Enable Command and Status Register — Offset 0x0060C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ER	MER	—	MFE	ITD	ITTE	MRT	PRT	UR	UT	—					
TYPE	R/W		R	R/W							R					
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—								IACB	OACB	—	RETE	TSE	PTTL	—	
TYPE	R								R/W		R	R/W		R/W	R	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LTLEECSR contains the bits that control whether an error condition locks the logical/transport layer error detect and capture registers and is reported to the system host. LTLEECSR is stored in all ports and the message unit.

Table 16-68. LTLEECSR Field Descriptions

Bit	Reset	Description
ER 31	0	**I/O Error Response Enable** Enables reporting of an I/O error response. It captures and locks the error.
MER 30	0	**Message Error Response Enable** Enables reporting of a message error response. It captures and locks the error. Capture done in message unit.
— 29	0	Reserved. Write to zero for future compatibility.
MFE 28	0	**Message Format Error Enable** Enables reporting of a message format error. It captures and locks the error. Capture done in messaging unit.
ITD 27	0	**Illegal Transaction Decode Error Enable** Enables reporting of an illegal transaction decode error. It captures and locks the error.
ITTE 26	0	**Illegal Transaction Target Error Enable** Enables reporting of an illegal transaction target error. It captures and locks the error.
MRT 25	0	**Message Request Time-Out Error Enable** Enables reporting of a message request time-out error. It captures and locks the error. Capture done in messaging unit.
PRT 24	0	**Packet Response Time-Out Error Enable** Enables reporting of a packet response time-out error. It captures and locks the error.
UR 23	0	**Unsolicited Response Error Enable** Enables reporting of an unsolicited response error. It captures and locks the error.
UT 22	0	**Unsupported Transaction Error Enable** Enables reporting of an unsupported transaction error. It captures and locks the error.
— 21–8	0	Reserved. Write to zero for future compatibility.
IACB 7	0	**Inbound ATMU Crossed Boundary Error Enable** Enables reporting of a received transaction that crosses an inbound ATMU boundary. It captures and locks the error.
OACB 6	0	**Outbound ATMU Crossed Boundary** Indicates a received transaction that crosses an outbound ATMU boundary, a segment boundary, or a subsegmented boundary.

Table 16-68. LTLEECSR Field Descriptions (Continued)

Bit	Reset	Description
— 5	0	Reserved. Write to zero for future compatibility.
RETE 4	0	**Retry Error Threshold Exceeded Enable** Enables reporting when the allowed number of logical retries is exceeded.
TSE 3	0	**Transport Size Error Enable** Enables error reporting when the field is not consistent with the CTLS bit of the processing element features CAR (that is, the tt value is reserved or indicates a common transport system unsupported by this device).
PTTL 2	0	**Packet Time-to-Live Error Enable** Enables reporting of a packet time-to-live error. Capture and lock the result.
— 2–0	0	Reserved. Write to zero for future compatibility.

16.6.27 Logical/Transport Layer Address Capture Command and Status Register (LTLACCSR)

LTLACCSR Logical/Transport Layer Address Capture Command and Status Register — Offset 0x00614

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

LTLACCSR contains error information. It is locked when a logical/transport error is detected and the corresponding enable bit is set. LTLACCSR is stored in the port and in the message unit, although the values in this register can differ between the port and the message unit. The message unit LTLACCSR cannot lock if the port is locked; the port LTLACCSR cannot lock if the message unit is locked.

Note: Fields in this register can be set by writing to the register. You can use this to emulate a hardware error during software development. Undefined results occur if fields in the register are changed while an actual Logical/Transport error is being detected.

Table 16-69. LTLACCSR Field Descriptions

Bit	Reset	Description
A 31–3	0	**Address** Normally, the least significant 29 bits of the address associated with the error (for requests, for responses, if available). For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors*, on page 16-30.

Table 16-69. LTLACCSR Field Descriptions

Bit	Reset	Description
— 2	0	Reserved. Write to zero for future compatibility.
XA 1–0	0	**Extended Address MSBs** Normally the extended address bits of the address associated with the error (for requests, responses, if available). For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-30.

16.6.28 Logical/Transport Layer Device ID Capture Command and Status Register (LTLDIDCCSR)

LTLDIDCCSR Logical/Transport Layer Device ID Capture Command and Status Register — Offset 0x00618

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DIDMSB								DID							
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	SIDMSB								SID							
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LTLDIDCCSR contains error information. It is locked when a logical/transport error is detected and the corresponding enable bit is set. LTLDIDCCSR is stored in the port and in the message unit, although the values in this register can differ between the port and the message unit. The message unit LTLDIDCCSR cannot lock if the port is locked; the port LTLDIDCCSR cannot lock if the message unit is locked.

Note: Fields in this register can be set by writing to the register. You can use this to emulate a hardware error during software development. Undefined results occur if fields in the register are changed while an actual Logical/Transport error is being detected.

Table 16-70. LTLDIDCCSR Field Descriptions

Bit	Reset	Description
DIDMSB 31–24	0	**Destination ID MSB** Normally, the most significant byte of the destination ID associated with the error. This field is valid only if the CTLS bit of the Processing Element Features CAR is set (large transport systems only). For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-30.
DID 23–16	0	**Destination ID** Normally, the destination ID (or least significant byte of the destination ID if large transport system) associated with the error. For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-30.

Table 16-70. LTLDIDCCSR Field Descriptions

Bit	Reset	Description
SIDMSB 15–8	0	**Source ID MSB** Normally, the most significant byte of the source ID associated with the error. This field is valid only if the CTLS bit of the Processing Element Features CAR is set (large transport systems only). For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-30.
SID 7–0	0	**Source ID** Normally, the source ID (or least significant byte of the source ID if large transport system) associated with the error. For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-30.

16.6.29 Logical/Transport Layer Control Capture Command and Status Register (LTLCCCSR)

LTLCCCSR Logical/Transport Layer Control Capture Command and Status Register — Offset 0x0061C

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

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 16-71. LTLCCCSR Field Descriptions

Bit	Reset	Description
FT 31–28	0	**Format Type** Normally, the format type associated with the error. For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-30.
TT 27–24	0	**Transaction Type** Normally, the transaction type associated with the error. For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-30.
MI 23–16	0	**Message Information** Normally, the message information: letter, mbox, and msgseg for the last message request received for the mailbox with an error (message errors only). For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-30.
— 15–0	0	Reserved. Write to zero for future compatibility.

16.6.30 Port 0 Error Detect Command and Status Register (P0EDCSR)

P0EDCSR Port 0 Error Detect Command and Status Register Offset 0x00640

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

P0EDCSR 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 16-72. P0EDCSR Field Descriptions

Bit	Reset	Description
— 31–23	0	Reserved. Write to zero for future compatibility.
CCS 22	0	**CRC Control Symbol** Received a control symbol with a bad CRC value.
AUA 21	0	**Acknowledge Control With Unexpected Acknowledge ID** Received an acknowledge control symbol with an unexpected acknowledge ID (packet-accepted or packet-retry).
PNA 20	0	**Packet Not Accepted** Received packet-not-accepted acknowledge control symbol.
UA 19	0	**Unexpected Acknowledge ID** Received packet with unexpected ackID value.
CRC 18	0	**Bad CRC** Received a packet with a bad CRC value.
EM 17	0	**Exceed Maximum** Received packet which exceed the maximum allowed size (276 bytes).
— 16–6	0	Reserved. Write to zero for future compatibility.
NOA 5	0	**Not Outstanding Acknowledge** Link-response received with an ackID that is not outstanding.
PE 4	0	**Protocol Error** An unexpected packet or control symbol was received.
— 3	0	Reserved. Write to zero for future compatibility.
DE 2	0	**Delineation Error** Received unaligned /SC/ or /PD/ or undefined code-group.
UCS 1	0	**Unsolicited Control Symbol** An unsolicited acknowledge control symbol was received.
LTO 0	0	**Link Time-Out** An acknowledge or link-response control symbol is not received within the specified time-out interval.

16.6.31 Port 0 Error Rate Enable Command and Status Register (P0ERECSR)

P0ERECSR Port 0 Error Rate Enable Command and Status Register Offset 0x00644

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

P0ERECSR contains the bits that control when an error condition is allowed to increment the error rate counter in the port 0 error rate threshold register and lock the port 0 error capture registers.

Table 16-73. P0ERECSR 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.

16.6.32 Port 0 Error Capture Attributes Command and Status Register (P0ECACSR)

P0ECACSR Port 0 Error Capture Attributes Command and Status Register Offset 0x00648

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IT		—	ET					ECI15	ECI14	ECI13	ECI12	ECI11	ECI10	ECI9	ECI8
TYPE	R/W		R	R/W												
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	ECI7	ECI6	ECI5	ECI4	ECI3	ECI2	ECI1	ECI0	—							CVI
TYPE	R/W								R							R/W
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

P0ECACSR indicates the type of information contained in the port 0 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 P0ECACSR[CVI] bit is set to ensure that the error is properly captured.

Table 16-74. P0ECACSR 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 0 error detect CSR that describes the error captured in the port 0 error capture CSRs.	

Table 16-74. P0ECACSR 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	0	**Contain Valid Information** Hardware sets CVI to indicate that the packet/control symbol capture registers contain valid information. For control symbols, only capture register 0 contains meaningful information.	

16.6.33 Port 0 Packet/Control Symbol Error Capture Command and Status Register (P0PCSECCSR0)

P0PCSECCSR0 Port 0 Packet/Control Symbol Error Capture Command and Status Register — Offset 0x0064C

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

P0PCSECCSR0 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 P0ECACSR[CVI] bit is set before reading the capture registers to ensure that the error is properly captured.

Table 16-75. P0PCSECCSR0 Field Descriptions

Bit	Reset	Description
C0 31–0	0	**Capture 0** Control character and control symbol or bytes 0–3 of the packet header.

16.6.34 Port 0 Packet Error Capture Command and Status Register 1 (P0PECCSR1)

P0PECCSR1 Port 0 Packet Error Capture Command and Status Register 1 Offset 0x00650

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

P0PECCSR1 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 P0ECACSR[CVI] bit is set before reading the capture registers to ensure that the error is properly captured.

Table 16-76. P0PECCSR1 Field Descriptions

Bit	Reset	Description
C1 31–0	0	**Capture 1** Contains bytes 4–7 of the packet header.

16.6.35 Port 0 Packet Error Capture Command and Status Register 2 (P0PECCSR2)

P0PECCSR2 Port 0 Packet Error Capture Command and Status Register 2 Offset 0x00654

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

P0PECCSR2 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 P0ECACSR[CVI] bit is set before reading the capture registers to ensure that the error is properly captured.

Table 16-77. PECCSR2 Field Descriptions

Bit	Reset	Description
C2 31–0	0	**Capture 2** Bytes 8 to 11 of the packet header

16.6.36 Port 0 Packet Error Capture Command and Status Register 3 (P0PECCSR3)

P0PECCSR3 Port 0 Packet Error Capture Command and Status Register 3 Offset 0x00658

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

P0PECCSR3 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 P0ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Table 16-78. P0PECCSR3 Field Descriptions

Bit	Reset	Description
C3 31–0	0	**Capture 3** Bytes 12–15 of the packet header.

16.6.37 Port 0 Error Rate Command and Status Register (P0ERCSR)

P0ERCSR Port 0 Error Rate Command and Status Register Offset 0x00668

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

P0ERCSR is used with the Port0 Error Rate Threshold Command and Status Register (P0ERTCSR) to monitor and control the reporting of transmission errors.

Table 16-79. P0ERCSR Field Descriptions

Bit	Reset	Description	Setting
ERB 31–24	0b1000_0000	**Error Rate Bias** Provides the error rate bias value.	0x00 Do not decrement the error rate counter. 0x01 Decrement every 1 ms (±34%). 0x02 Decrement every 10 ms (±34%). 0x04 Decrement every 100 ms (±34%). 0x08 Decrement every 1 s (±34%). 0x10 Decrement every 10 s (±34%). 0x20 Decrement every 100 s (±34%). 0x40 Decrement every 1000 s (±34%). 0x80 Decrement every 10000 s (±34%). Other values are reserved and cause undefined operation.
— 23–18	0	Reserved. Write to zero for future compatibility.	
ERR 17–16	0	**Error Rate** Limits increments of the error rate counter above the failed threshold trigger. This counter never increments above 0xFF, even if the combined settings of ERR and the failed threshold trigger imply that it would.	0b00 Count only 2 errors above. 0b01 Count only 4 errors above. 0b10 Count only 16 error above. 0b11 Do not limit increments above the error rate count.
PER 15–8	0	**Peak Error Rate** Contains the peak value attained by the error rate counter.	
ERC 7–0	0	**Error Rate Counter** Counts the number of transmission errors detected by the port, decremented by the error rate bias, to create an indication of the link error rate. Software should not attempt to write to ERC a value higher than the failed threshold trigger plus the number of errors specified in the ERR field (the maximum ERC value).	

16.6.38 Port 0 Error Rate Threshold Command and Status Register (P0ERTCSR)

P0ERTCSR Port 0 Error Rate Threshold Command and Status ‘Register Offset 0x0066C

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

P0ERTCSR controls reporting of the link status to the system host.

Table 16-80. P0ERTCSR 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.	

16.6.39 Logical Layer Configuration Register (LLCR)

LLCR Logical Layer Configuration Register Offset 0x10004

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

LLCR contains general port-common logical layer mode enables.

Table 16-81. LLCR Field Descriptions

Bit	Reset	Description
— 31–30	0	Reserved. Write to zero for future compatibility.
ECRAB 29	0	**External Configuration Register Access Block** Blocks all maintenance requests and accesses to LCSBA1CSR. Reads return all 0s, and writes are ignored (both return a done response). When ECRAB is cleared, any external RapidIO device can access the registers.
— 28–0	0	Reserved. Write to zero for future compatibility.

16.6.40 Error/Port-Write Status Register (EPWISR)

EPWISR Error/Port-Write Interrupt Status Register Offset 0x10010

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

EPWISR contains status bits of the interrupts generated by the port or the message unit for physical or logical/transport layer errors or inbound port-writes. Because errors from the port are reported to the SC3400 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 16-82. EPWISR Field Descriptions

Bit	Reset	Description
PINT 31	0	**Physical or Logical/Transport Error Interrupt** Indicates a physical or logical transport error interrupt was generated by the port.
— 30–2	0	Reserved. Can be used to indicate errors on more ports if they exist.
MU 1	0	**Message Unit Logical/Transport Layer Error Interrupt** Indicates a logical/transport layer error interrupt was generated by the message unit.
PW 0	0	**Port Write** Indicates an inbound port-write was received.

16.6.41 Logical Retry Error Threshold Configuration Register (LRETCR)

LRETCR Logical Retry Error Threshold Configuration Register Offset 0x10020

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	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															
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 16-83. LRETCR Field Descriptions

Bit	Reset	Description	
— 31–8	0	Reserved. Write to zero for future compatibility.	
RET 7–0	0xFF	**Retry Error Threshold** The threshold value for the number of consecutive logical retries received for a given packet that causes the RAPIDIO ENDPOINT to report an error condition.	0x00 Disable the retry threshold. 0x01 Set the error reporting threshold to 1. ... 0xFF Set the error reporting threshold to 255.

16.6.42 Physical Retry Error Threshold Configuration Register (PRETCR)

PRETCR Physical Retry Error Threshold Configuration Register Offset 0x10080

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

PRETCR contains the retry error threshold for the physical layer. When the number of consecutive acknowledge-retries is greater than or equal to this value, an error interrupt is generated.

Table 16-84. PRETCR Field Descriptions

Bit	Reset	Description	Settings
— 31–8	0	Reserved. Write to zero for future compatibility.	
RET 7–0	0xFF	**Retry Error Threshold** The threshold value for the number of consecutive acknowledge-retries received that cause the RAPIDIO ENDPOINT to report an error condition.	0x00 Disable the retry threshold. 0x01 Set the error reporting threshold to 1. ... 0xFF Set the error reporting threshold to 255.

16.6.43 Port 0 Alternate Device ID Command and Status Register (P0ADIDCSR)

P0ADIDCSR Port 0 Alternate Device ID Command and Status Register Offset 0x10100

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

P0ADIDCSR contains an alternate device ID. This register should be enabled before the initiator enabled bit of the PGCCSR is set (see **Table 16-60,** *PGCCSR Field Descriptions,* on page 16-122). 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 P0ADIDCSR 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 P0ADIDCSR 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 P0ADIDCSR. It generates responses with the deviceID in the original request packet (either from P0ADIDCSR 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 16-85. P0ADIDCSR Field Descriptions

Bit	Reset	Description
ADE 31	0	**Alternate Device ID Enable** Causes the port to use the device ID specified in this register instead of the device ID specified in BDIDCSR.
— 30–24	0	Reserved. Write to zero for future compatibility.
ADID 23–16	0	**Alternate Device ID** Alternate device ID in a small transport system.
LADID 15–0	0	**Large Alternate Device ID** Alternate device ID for the device in a large transport system.

16.6.44 Port 0 Accept-All Configuration Register (P0AACR)

P0AACR Port 0 Accept-All Configuration Register Offset 0x10120

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**
	—															AA
TYPE	R															R/W
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

P0AACR contains information on accept-all mode.

Table 16-86. P0AACR Field Descriptions

Bit	Name	Description	Settings
— 31–1	0	Reserved. Write to zero for future compatibility.	
AA 0	0	**Accept All** Specifies whether packet acceptance is based on a target ID. When this bit is set, the tt field value must be consistent with the common transport system specified by the CTLS bit of the processing element features CAR.	0 Normal RapidIO acceptance based on target ID. 1 All packets are accepted without checking the target ID.

16.6.45 Port 0 Logical Outbound Packet Time-to-Live Configuration Register (P0LOPTTLCR)

P0LOPTTLCR Port 0 Logical Outbound Packet Time-to-Live Configuration Register Offset 0x10124

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

The port 0 logical outbound packet time-to-live configuration register (P0LOPTTLCR) 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, P0PCR[OBDEN] is automatically set. P0PCR[OBDEN] must be cleared by software.

Table 16-87. P0LOPTTLCR Field Descriptions

Bit	Reset	Description
TV 31–8	0	**Time-out Value** Setting to all zeros disables the time-to-live time-out timer. This value is loaded each time the time-to-live time-out timer starts.
— 7–0	0	Reserved. Write to zero for future compatibility.

16.6.46 Port 0 Implementation Error Command and Status Register (P0IECSR)

P0IECSR Port 0 Implementation Error Command and Status Register Offset 0x10130

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

P0IECSR contains status bits that are asserted when an implementation-defined error occurs.

Table 16-88. P0IECSR Field Descriptions

Bit	Reset	Description
RETE 31	0	**Retry Error Threshold Exceeded** Set when the number of consecutive retries reaches the retry error threshold in the Physical Retry Error Threshold Configuration Register (PRETCR). RETE is cleared by writing a value of 1 to it. This bit is set again if another retry is received and the number of consecutive retries continues to exceed the retry error threshold.
— 30–0	0	Reserved. Write to zero for future compatibility.

16.6.47 Port 0 Physical Configuration Register (P0PCR)

P0PCR Port 0 Physical Configuration Register Offset 0x10140

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**
	CCC	—		—								CCP	—	OBDEN	—	
TYPE	R/W	R	R/W	R								R/W	R	R/W		R
RESET	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

P0PCR contains general physical layer protocol and link mode enables.

Table 16-90. P0PCR 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.

16.6.48 Port 0 Serial Link Command and Status Register (P0SLCSR)

P0SLCSR Port 0 Serial Link Command and Status Register Offset 0x10158

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	LS0	LS1	LS2	LS3	—				LA	—						
TYPE	W1C	W1C	W1C	W1C	R				W1C	R						
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

P0SLCSR contains status of the of the serial physical link.

Table 16-91. P0SLCSR Field Descriptions

Bit	Reset	Description
LS0 31	0	**Lane Sync Achieved for Lane 0** Write with 1 to clear
LS1 30	0	**Lane Sync Achieved for Lane 1** Write with 1 to clear
LS2 29	0	**Lane Sync Achieved for Lane 2** Write with 1 to clear
LS3 28	0	**Lane Sync Achieved for Lane 3** Write with 1 to clear
— 27–24	0	Reserved. Write to zero for future compatibility.
LA 23	0	**Lane Alignment Achieved** Write with 1 to clear.
— 22–0	0	Reserved. Write to zero for future compatibility.

16.6.49 Port 0 Serial Link Error Injection Configuration Register (P0SLEICR)

P0SLEICR Port 0 Serial Link Error Injection Configuration Register Offset 0x10160

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

P0SLEICR controls the injection of bit errors into the transmit bit stream and is used to generate pseudo-random errors into the outbound serial RapidIO data stream. If the EIC field is non-zero, error injection is enabled and, at pseudo-random intervals, an error is injected by inverting a single bit in the outgoing data stream. The range of the pseudo-random value (delay between injected errors) is controlled by the EIR field. That is, the value of EIR, multiplied by 32, determines the maximum number of character times between injected errors.

Table 16-92. P0SLEICR Field Descriptions

Bit	Reset	Description	Settings
EIC 31–27	0	**Error Injection Control** Enables and controls serial link error injection as follows.	00000 Error injection is disabled. 10000 Error injection, lane 0 only. 01000 Error injection, lane 1 only 00100 Error injection, lane 2 only 00010 Error injection, lane 3 only 11110 Error injection, all lanes simultaneously 11111 Error injection, randomly distributed over all 4 lanes All other values reserved.
— 26–20	0	Reserved. Write to zero for future compatibility.	
EIR 19–0	0	**Error Injection Range** The value of EIR × 32 determines the maximum value of the pseudo-random delay between errors. For example, a value of 0x1 indicates a maximum delay of 32 character times. The value within this register should be right-justified.	

16.6.50 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	0	0

IPBRR1 tracks changes and revisions of the RapidIO endpoint.

Table 16-93. IPBRR1 Field Descriptions

Bit	Reset	Description
IPID 31–16	0x01C0	IP block ID = 0x01C0
IPMJ 15–8	0x01	Major revision of the IP block = 0x01
IPMN 7–0	0x00	Minor revision of the IP block = 0x0

16.6.51 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 16-94. IPBRR2 Field Descriptions

Bit	Reset	Description
— 31–24	0	Reserved. Write to zero for future compatibility.
IPINT 8–15	0	IP block Integration options = 0x0.
— 16–23	0	Reserved. Write to zero for future compatibility.
IPCFG 7–0	0	IP Block Configuration Options = 0x0.

16.6.52 Port 0 RapidIO Outbound Window Translation Address Registers x (P0ROWTARx)

P0ROWTAR[0–8] Port 0 RapidIO Outbound Window Translation Address Registers 0–8 Offset 0x10C00 + x*0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reg. Trans	LTGTID		TREXAD										TRAD			
Maint. Trans.	—		TARGETID								—		HOPCOUNT			
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**
Reg. Trans	TRAD															
Maint. Trans.	HOPCOUNT				MAINT_OFFSET											
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

P0ROWTARx points to the starting addresses in the RapidIO address space for window hits within the outbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Table 16-95. P0ROWTAR*x* Field Descriptions

Bit	Reset	Description
Regular Transaction		
LTGTID 31–30	0	**Large Transport System Target Address** LTGTID corresponds to bits 6–7 of the targetID for a large transport system. This field is valid only if the PEFCAR[CTLS] bit is set (see **Section 16.6.5**, *Processing Element Features Capability Register (PEFCAR),* on page 16-106). Bits 0–5 of the target ID are specified in the P0ROWTEARX (see **Section 16.6.53**, *Port 0 RapidIO Outbound Window Translation Extended Address Registers x (P0ROWTEARx),* on page 16-159
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 fro 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 size field. This corresponds to bits 2–21 of the 34-bit RapidIO translation address (defined as [0–33]). This field is reserved for default window 0.
Maintenance Transaction		
— 31–30	0	Reserved. Write to zero for future compatibility.
TARGETID 29–22	0	**Target ID** Used to identify the target of the maintenance transaction.
— 21–20	0	Reserved. Write to zero for future compatibility.

Table 16-95. P0ROWTAR*x* Field Descriptions

Bit	Reset	Description
HOPCOUNT 19–12	0	**Hop Count** This field contains the hop count value. This field is reserved for default window 0.
MAINT_ OFFSET 11–0	0	**Maintenance Offset** Contains the upper 12 bits of the maintenance offset value. This field is reserved for default window 0.

16.6.53 Port 0 RapidIO Outbound Window Translation Extended Address Registers x (P0ROWTEARx)

P0ROWTEAR[0–8] Port 0 RapidIO Outbound Window Translation Extended Address Registers 0–8 Offset 0x10C04 + x*0x20

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**
	—										LTGTID					
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

P0RIWTARx 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 16-96. P0RIWTAR*x* Field Descriptions

Bit	Reset	Description
— 31–6	0	Reserved. Write to zero for future compatibility.
LTGID 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 16.6.5**, *Processing Element Features Capability Register (PEFCAR),* on page 16-106). Bits 6–7 of the target ID are specified in the corresponding P0ROWTARx.

16.6.54 Port 0 RapidIO Outbound Window Attributes Registers x (P0ROWARx)

P0ROWARx Port 0 RapidIO Outbound Window Attributes Registers 0–8 Offset 0x10C10 + x*0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EN	—			TFLOWLV		PCI	—	NSEG		NSSEG		RDTYP			
TYPE	R/W	R			R/W			R	R/W							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	WRTYP				—						SIZE					
TYPE	R/W				R						R/W (R for default window 0)					
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The port0 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 16-97. P0ROWAR*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.
— 30–28	0	Reserved. Write to zero for future compatibility.	
TFLOWLY 27–26		**Transaction Flow Level** Selects the transaction flow priority level. This field must be set to 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 Next highest priority transaction request flow. 10 Highest priority transaction request flow. 11 Reserved.
PCI 25		**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	0	**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 16-97. P0ROWAR*x* Field Descriptions (Continued)

Bit	Reset	Description	Settings
NSSEG 21–20	0	**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		**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 15–12		**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		Reserved. Write to zero for future compatibility.	
SIZE 5–0		**Window Size** Outbound translation window size N, which is the encoded 2^(N+1) byte window size. The smallest window size is 4 Kbyte. **Note:** This field is read-only for default window 0.	000000 Reserved. ... 001011 4 KB window size. 001100 8 KB window size. ... 011111 4 GB window size. 100000 8 GB window size. 100001 16 GB window size. 100010 32 GB window size. 100011 64 GB window size. 100100 Reserved ... 111111 Reserved.

16.6.55 Port 0 RapidIO Outbound Window Base Address Registers x (P0ROWBARx)

P0ROWBAR[1–8] Port 0 RapidIO Outbound Window Base Address Registers 1–8 — Offset 0x10C08 + 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

P0ROWBARx 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 out of the device using the default window. For information on transactions that cross more than one window, see **Section 16.2.5.4.2**, *Window Boundary Crossing Errors*, on page 16-22.

Table 16-98. P0ROWBAR*x* Field Descriptions

Bit	Reset	Description
— 31–24	0	Reserved. Write to zero for future compatibility.
BEXAD 23–20	0	**Base Extended Address** Bits 0–3 of the 36-bit RapidIO base address. **Note:** Bit 0 is the most significant bit.
BADD 19–0	0	**Base Address** A system address that is the starting-point for the outbound translation window. The window must be aligned on the basis of the size selected in the window size bits. This corresponds to bits 4–23 of the 36-bit RapidIO base address. **Note:** Bit 0 is the most significant bit.

16.6.56 Port 0 RapidIO Outbound Window Segment 1–3 Registers 1–8 (P0ROWSxRn)

Port 0 RapidIO Outbound Window Segment 1–3 Registers 1–8

P0ROWS[1–3]R[1–8] Offset 0x10C34 + (x–1)*0x4 + (n –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

P0ROWSxRn define the attributes and target device ID to use for a transaction that hits in the segment rather than the primary attributes and target deviceID.There is a segment register for each segment except segment 0.

Table 16-99. P0ROWSxRn Field Descriptions

Bit	Description	Settings
— 31–28	Reserved. Write to zero for future compatibility.	
TFLOWLY 27–26	**Transaction Flow Level** Selects the transaction flow priority level. **Note:** This field must be set to 00 if the PCI bit is set.	00 Lowest priority transaction request flow. 01 Next highest priority transaction request flow. 10 Highest priority transaction request flow. 11 Reserved.
— 25–24	Reserved. Write to zero for future compatibility.	
RDTYP 23–20	**Read Type** Transaction type to run on the RapidIO interface if the access is a read.	0100 NREAD. 0111 Maintenance read. All other values are reserved.
WRTYP 19–16	**Write Type** Transaction type to run on the RapidIO interface if access is a write.	0011 SWRITE. 0100 NWRITE. 0101 NWRITE_R. 0111 Maintenance Write. All other values are reserved.
— 15–8	Reserved. Write to zero for future compatibility.	
SGTGTDID 7–0	**Segment Target DeviceID** Stores the Target DeviceID as follows: • SGTGTDID[7–3]: Bits 0–4 for small transport or bits 8–12 for large transport. • SGTGTDID2: Bit 5 for small transport or bit 13 for large transport; reserved for 8 target subsegments. • SBTGTDID1: Bit 6 for small transport or bit 14 for large transport; reserved for 8 or 4 target subsegments. • SBTGTDID0: Bit 7 for small transport or bit 15 for large transport; reserved for 8, 4, or 2 target subsegments.	

16.6.57 Port 0 RapidIO Inbound Window Translation Address Registers x (P0RIWTARx)

P0RIWTAR[0–4] Port 0 RapidIO Inbound Window Translation Address Registers 0–4 — Offset 0x10D60 + (4 –x)*0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—												TRAD			
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**
	TRAD															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

P0RIWTARx points to the starting addresses in the RapidIO address space for window hits within the inbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Table 16-100. P0RIWTAR*x* Field Descriptions

Bit	Reset	Description
— 31–20	0	Reserved. Write to zero for future compatibility.
TRAD 19–0	0	**Translation Address** Target address to indicate the starting point of the inbound translated address. The translation address must be aligned on the basis of the size field. TRAD is reserved for default window 0.

16.6.58 Port 0 RapidIO Inbound Window Base Address Registers x (P0RIWBARx)

P0RIWBAR[1–4] Port 0 RapidIO Inbound Window Base Address Registers 1–4 Offset 0x10D68 + (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

P0RIWBARx selects the base address for the windows that are translated to an alternate target address space. Addresses for inbound transactions are compared with the addresses of these windows. If such a transaction does not fall within one of these spaces, it is forwarded to the interior of the device using the default window. For information on transactions that cross more than one window, see **Section 16.2.5.4.2**, *Window Boundary Crossing Errors*, on page 16-22.

Table 16-101. P0RIWBAR*x* Field Descriptions

Bit	Reset	Description
— 31–22	0	Reserved. Write to zero for future compatibility.
BEXAD 21–20	0	**Base Extended Address** Bits 0–1 of the 34-bit RapidIO base address. **Note:** Bit 0 is the most significant bit.
BADD 19–0	0	**Base Address** A system address that is the starting-point for the inbound translation window. The window must be aligned on the basis of the size selected in the window size bits. This corresponds to bits 2–21 of the 34-bit RapidIO base address. **Note:** Bit 0 is the most significant bit.

16.6.59 Port 0 RapidIO Inbound Window Attributes Registers x (P0RIWARx)

P0RIWARx Port 0 RapidIO Inbound Window Attributes Registers 0–4 — Offset 0x10D70 + (4 –x)*0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EN	PW	—						TGINT				RDTYP			
TYPE	R/W		R						R/W							
RESET	0000_0000_0000_0100 (P0RIWAR 1–4) 1000_0000_0000_0100 (P0RIWAR 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 (P0RIWAR 0–4)															

The port0 RapidIO inbound window attributes registers define the window sizes to translate and other attributes of the translations. The largest window size allowed is 16 GB. The RDTYP and WRTYP fields are used for attributes on the request, but they do not change the type of the transaction. In other words, P0RIWAR*x* does not modify whether the request requires a response or not. This type of attribute remains unchanged on the request as it is translated through the ATMU.

Table 16-102. P0RIWAR*x* Field Descriptions

Bit	Description	Settings
EN 31	**Enable Address Translation** Set to 1 and read-only for default window 0.	
PW 30	**Protected Window** Indicates that this window is protected. Writes requiring a response and reads to this window generate an error response. Writes not requiring a response are silently discarded.	
— 29–24	Reserved. Write to zero for future compatibility.	
TGINT 23–20	**Target Interface** If this field is set to anything other than local address space, the attributes for the transaction must be assigned in a corresponding outbound window at the target.	0000–1110 Reserved. 1111 Local memory.
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.

Table 16-102. P0RIWAR*x* Field Descriptions (Continued)

Bit	Description	Settings
WRTYP 15–12	Transaction type to run on local memory if access is a write.	0000 Reserved. ... 0011 Reserved. 0100 Write. 0101 Reserved. ... 1111 Reserved.
— 11–6	Reserved. Write to zero for future compatibility.	
SIZE 5–0	**Window Size** Inbound translation window size N, which is the encoded 2^(N+1) byte window size. The smallest window size is 4 KB. This field is read-only for default window 0.	000000 Reserved. ... 001011 4 KB window size. 001100 8 KB window size. ... 011111 4 GB window size. 100000 8 GB window size. 100001 16 GB window size. 100010 Reserved. ... 111111 Reserved.

16.6.60 Outbound Message x Mode Registers (OMxMR)

OM[0–1]MR Outbound Message 0–1 Mode Registers Offset 0x13000 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—				SCTL			—								
TYPE	R				R/W			R								
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	CIRQ_SIZ				—		QOIE	QFIE	—	QEIE	EIE	—		MUTM	MUI	MUS
TYPE	R/W				R		R/W		R	R/W		R		R/W		
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OMxMR allows software to start a message operation and to control various message operation characteristics.

Table 16-103. OM*x*MR Field Descriptions

Bits	Reset	Description	
— 31–28	0	Reserved. Write to zero for future compatibility.	
SCTL 27–25	0	**Service Control** Determines the number of descriptors to process before the next queue is serviced. Note that if one queue has SCTL set to fixed priority, all SCTL values in other queues are ignored. For example, of the two outbound message units and the service control value for unit 1 is set to 0b000, a fixed priority is set and unit 0 handles the highest priority results. If a queue is in direct mode, only one message operation is serviced before the next queue is serviced. For proper operation, this field should be modified only when the outbound message controller is not enabled. The value of this field cannot be changed unless both units are disabled.	000 Fixed priority based on outbound message unit number. 001 1 descriptor. 010 2 descriptors. 011 4 descriptors. 100 8 descriptors. 101 16 descriptors. 110 32 descriptors. 111 64 descriptors.
— 24–16	0	Reserved. Write to zero for future compatibility.	
CIRQ_SIZ 15–12	0	**Circular Descriptor Queue Size** Determines the number of descriptors that can be placed on the circular queue without overflow. For proper operation, this field should be modified only when the outbound message controller is not enabled	0000 2. 0001 4. 0010 8. 0011 16. 0100 32. 0101 64. 0110 128. 0111 256. 1000 512. 1001 1024. 1010 2048. 1011— 1111 Reserved.
— 11–10	0	Reserved. Write to zero for future compatibility.	
QOIE 9	0	**Queue Overflow Interrupt Enable** Enables an interrupt when a queue overflow is detected. That is, the enqueue and dequeue pointers are no longer equal after the processor increments the enqueue pointer while the queue is full. This bit is applicable only in chaining mode. No queue overflow interrupt is generated if this bit is cleared. If this bit is not set and the queue overflows, the result is undefined.	
QFIE 8	0	**Queue Full Interrupt Enable** Enables an interrupt when the queue transitions to full. That is, the enqueue and dequeue pointers are equal after the processor increments the enqueue pointer. No QF interrupt is generated if this bit is cleared. If this bit is set and OM*x*SR[QF] is set, OM*x*SR[QFI] becomes set.	
— 7	0	Reserved. Write to zero for future compatibility.	

Table 16-103. OM*x*MR Field Descriptions (Continued)

Bits	Reset	Description	
QEIE 6	0	**Queue Empty Interrupt Enable** Enables an interrupt at the completion of all outstanding message operations. That is, the enqueue and dequeue pointers are equal after an increment by the message unit controller. No Queue Empty interrupt is generated if this bit is cleared. For proper operation, this field should be modified only when the outbound message controller is not enabled	0 No interrupt. 1 Queue empty interrupt.
EIE 5	0	**Error Interrupt Enable** Enables a port-write/error interrupt when a transfer error (OM*x*SR[TE]), a message error response (OM*x*SR[MER]), a packet response time-out (OM*x*SR[PRT]), or a retry threshold event exceeded (OM*x*SR[RETE]) event occurs. No port-write/error interrupt is generated if this bit is cleared.	0 No port-write/error interrupt. 1 Generate a port-write/error interrupt.
— 4–3	0	Reserved. Write to zero for future compatibility.	
MUTM 2	0	**Message unit Transfer Mode** Puts the message unit into direct mode so that software is responsible for placing all the required parameters into registers to start the message transmission. Clearing this bit configures the message unit in chaining mode.	0 Chaining mode. 1 Direct mode.
MUI 1	0	**Message Unit Increment** Software sets this bit after writing a descriptor to memory. Hardware then increments the OM*x*DQEPAR and clears this bit. MUI always reads as 0 when MUS is set.	
MUS 0	0	**Message Unit Start** In Direct mode, a 0 to 1 transition when the message unit is not busy (MUB bit is 0) starts the message unit. A 1 to 0 transition has no effect. If this bit is set in Chaining mode, the message unit starts when the enqueue and dequeue pointers are not equal.	

16.6.61 Outbound Message x Status Registers (OMxSR)

OM[0–1]SR Outbound Message 0–1 Status Registers Offset 0x13004 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—											QF	—			
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—			MER	RETE	PRT	—		TE	—	QOI	QFI	—	MUB	EOMI	QEI
TYPE	R			W1C	W1C	W1C	R		W1C	R	W1C	W1C	R		W1C	W1C
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OMxSR reports various message unit conditions during and after a message operation. Writing a 1 to the corresponding set bit clears the bit.

Table 16-104. OM*x*SR Field Descriptions

Bits	Reset	Description
— 31–21	0	Reserved. Write to zero for future compatibility.
QF 20	0	**Queue Full** If the queue becomes full, this bit is set. Read only.
— 19–13	0	Reserved. Write to zero for future compatibility.
MER 12	0	**Message Error Response** Set when an ERROR response is received from the message target. The error response received field indicates the value of the error response status bits when an error response is received. This bit is cleared by writing a value of 1 to it.
RETE 11	0	**Retry Error Threshold Exceeded** Set when the message unit is unable to complete a message operation because the retry error threshold value is exceeded due to a RapidIO retry response. This bit is cleared by writing a value of 1 to it.
PRT 10	0	**Packet Response Time-Out** Set when the message unit has been unable to complete a message operation and a packet response time-out occurred. This bit is cleared by writing a 1.
— 9–8	0	Reserved. Write to zero for future compatibility.
TE 7	0	**Transaction Error** Set when an internal error condition occurs during the message operation. This bit is cleared by writing a value of 1 to it. For proper operation, this field should be modified only when the outbound message controller is not enabled
— 6	0	Reserved. Write to zero for future compatibility.
QOI 5	0	**Queue Overflow Interrupt** Set when a queue overflow condition is detected. This bit is cleared by writing a value of 1 to it. QOI is applicable only to chaining mode.
QFI 4	0	**Queue Full Interrupt** If the queue becomes full and the QFIE bit in the Mode Register is set, this bit is set and an interrupt is generated. This bit is cleared by writing a value of 1 to it.
— 3	0	Reserved. Write to zero for future compatibility.
MUB 2	0	**Message Unit Busy** Indicates that a message operation is currently in progress. This bit is cleared when an error occurs or the message operation completes. Read only.
EOMI 1	0	**End-Of-Message Interrupt** When the message operation completes and the EOMIE bit in the Destination Attributes Register is set, EOMI is set and an interrupt is generated. This bit is cleared by writing a value of 1 to it.
QEI 0	0	**Queue Empty Interrupt** When the last message operation in the outbound descriptor queue is finished and the QEIE bit in the Mode Register is set, this bit is set and an interrupt is generated. Otherwise, no interrupt is generated. This bit is cleared by writing a value of 1 to it.

16.6.62 Outbound Message x Descriptor Queue Dequeue Pointer Address Registers (OMxDQDPAR)

OM[0–1]DQDPAR Outbound Message 0–1 Descriptor Queue Dequeue Pointer Address Registers — Offset 0x1300C + x*0x100

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

OMxDQDPAR contain the address of the first descriptor in memory to be processed. Software must initialize this register to point to the first descriptor in memory. After the descriptor is processed, the message unit controller increments the outbound message descriptor queue dequeue pointer address in OM*x*DQDPAR to point to the next descriptor. If the outbound message descriptor queue enqueue pointer and the outbound message descriptor queue dequeue pointer are not equal (indicating that the queue is not empty), the message unit controller reads the next descriptor from memory for processing. If the enqueue and dequeue pointers are equal after the message unit controller increments the dequeue pointer, the queue is empty and the message unit halts until the processor increments the enqueue pointer. Incrementing the pointer indicates that a new descriptor was added to the queue and is ready for processing. If the queue becomes empty and OM*x*MR[QEIE] is set, OM*x*SR[QEI] is set and an interrupt is generated.

When software initializes these registers, they must be aligned on a boundary equal to the number of queue entries × 32 bytes (the size of each queue descriptor).For example, if there are eight entries in the queue, the register must be 256-byte aligned. The number of queue entries is set in OMnMR[CIRQ_SIZ].

Table 16-105. OM*x*DQDPAR Field Descriptions

Bits	Reset	Description
DQDPA 31–5	0	**Descriptor Dequeue Pointer Address** Contains the address of the first descriptor in memory to process. The descriptor must be aligned to a 32-byte boundary. For proper operation, this field should be modified only when the outbound message controller is not enabled.
— 4–0	0	Reserved. Write to zero for future compatibility.

16.6.63 Outbound Message x Source Address Registers (OMxSAR)

OM[0–1]SAR Outbound Message 0–1 Source Address Registers — Offset 0x13014 + x*0x100

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

OMxSAR indicates the address from which the message unit controller is to read data. Software must ensure that this is a valid local memory address. The source address must be aligned to a double-word boundary, so the least significant three bits are reserved.

Table 16-106. OM*x*SAR Field Descriptions

Bits	Reset	Description
SAD 31–3	0	**Source Address** The source address of the message operation. The contents are updated after every memory read operation. For proper operation, this field should be modified only when the outbound message controller is not enabled
— 2–0	0	Reserved. Write to zero for future compatibility.

16.6.64 Outbound Message x Destination Port Register (OMxDPR)

OM[0–1]DPR Outbound Message 0–1 Destination Port Registers — Offset 0x13018 + x*0x100

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

OMxDPR indicates the RapidIO destination ID and mailbox to which the message unit controller is to send data. The software must ensure that this is a valid port in the receiving device.

Table 16-107. OM*x*DPR Field Descriptions

Bits	Reset	Description
EDGTROUTE 31–24	0	**Extended Destination Target Route** Most significant byte of a 16-bit target route when activated in Large Transport mode. Reserved when operated in Small Transport mode. For proper operation, this field should be modified only when the outbound message controller is not enabled
DTGTROUTE 23–16	0	**Destination Target Route** Contains the target route field of the transaction (device ID of the target). This value is overridden by the multicast group and list if Multicast mode is enabled. When an error occurs while Multicast mode is enabled, this field is loaded with the destination of the failed operation. For proper operation, this field should be modified only when the outbound message controller is not enabled
— 15–2	0	Reserved. Write to zero for future compatibility.
MAILB 1–0	0	**Value for MBOX Field in MESSAGE Packet** For proper operation, this field should be modified only when the outbound message controller is not enabled

16.6.65 Outbound Message x Destination Attributes Register (OMxDATR)

OM[0–1]DATR Outbound Message 0–1 Destination Attributes Registers — Offset 0x1301C + x*0x100

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

OMxDATR contains the transaction attributes to be used for the message operation.

Table 16-108. OM*x*DATR Field Descriptions

Bits	Reset	Description	
MM 31	0	**Multicast Mode** When set, sends the message operation to all targets indicated by the multicast group and list. Messages are limited to one segment and 256 bytes or less.	0 Normal operation. 1 Multicast mode.
— 30	0	Reserved. Write to zero for future compatibility.	
EOMIE 29	0	**End-of-Message Interrupt Enable** When set, generates an interrupt when the current message operation finishes. For proper operation, this field should be modified only when the outbound message controller is not enabled.	
— 28	0	Reserved. Write to zero for future compatibility.	
DTFLOWLVL 27–26	0	**Transaction Flow Level** For proper operation, this field should be modified only when the outbound message controller is not enabled	00 Lowest priority transaction request flow. 01 Next highest priority transaction request flow. 10 Highest priority transaction request flow. 11 Reserved.
— 25–24	0	Reserved. Write to zero for future compatibility.	
DTGTINT 23–20	0	**Target Interface** The value of this field should always be set to RapidIO (0x0).	
— 19–0	0	Reserved. Write to zero for future compatibility.	

16.6.66 Outbound Message x Double-Word Count Register (OMxDCR)

OM[0–1]DCR Outbound Message 0–1 Double-Word Count Registers — Offset 0x13020 + x*0x100

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

OMxDCR contains the number of double-words for the message operation. The maximum message operation size is 4 KB and the minimum is 8 bytes.

Table 16-109. OMxDCR Field Descriptions

Bits	Reset	Description	Settings
— 31–13	0	Reserved. Write to zero for future compatibility.	
DCR 12–3	0	**Transfer Count Register** Contains the number of bytes for the message operation.For proper operation, this field should be modified only when the outbound message controller is not enabled. The values with an asterisk (*) apply only in Multi-Segment mode. **Note:** The value in this register represents the number of double words to transfer and not the byte count; that is, the DCR value = the number of bytes/8.	00 0000 0000 Reserved. 00 0000 0001 8 bytes. 00 0000 0010 16 bytes. 00 0000 0100 32 bytes. 00 0000 1000 64 bytes. 00 0001 0000 128 bytes. 00 0010 0000 256 bytes. 00 0100 0000 512 bytes*. 00 1000 0000 1024 bytes*. 01 0000 0000 2048 bytes*. 10 0000 0000 4096 bytes*. All other values yield undefined behavior.
— 2–0	0	Reserved. Write to zero for future compatibility.	

16.6.67 Outbound Message x Descriptor Queue Enqueue Pointer Address Registers (OMxDQEPAR)

OM[0–1]DQEPAR Outbound Message x Descriptor Queue Enqueue Pointer Address Registers — Offset 0x13028 + x*0x100

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

OMxDQEPAR contains the address for the next descriptor in memory to be added to the queue. Software must initialize this register to match the outbound message descriptor queue dequeue pointer address. When a message is ready to be sent, the processor writes a descriptor to the next location in the queue (indicated by the address in OM*x*DQEPAR), and then either writes the OM*x*DQEPAR to point to the next descriptor location in memory or sets OM*x*MR[MUI]. This can result in a number of actions:

- If the enqueue and dequeue pointers match, the queue is now full. If the OM*x*MR[QFIE] bit is set, then the OM*x*SR[QFI] bit is set, and an interrupt is generated.
- If the enqueue and dequeue pointers no longer match after the enqueue pointer is incremented and the queue is full, then the queue overflows, and the message unit stops. If OM*x*MR[QOIE] is set, then the controller sets OM*x*SR[QOI] and generates an interrupt. OM*x*MR[MUS] must change from a 1 to a 0 to clear this error condition. If the enqueue pointer is written directly, the queue overflow condition is not detected.
- If the enqueue and dequeue pointers were the same before the register is incremented, the message unit controller will start, if enabled.

Note: When software initializes these registers, they must be aligned on a boundary equal to the number of queue entries × 32 bytes (the size of each queue descriptor). For example, if there are eight entries in the queue, the register must be 256-byte aligned. The number of queue entries is set in OMxMR[CIRQ_SIZ].

Table 16-110. OM*x*DQEPAR Field Descriptions

Bits	Reset	Description
DQEPA 31–5	0	**Descriptor Enqueue Pointer Address** Contains the address of the next free descriptor location. The descriptor must be aligned to a 32-byte boundary and a descriptor queue boundary.
— 4–0	0	Reserved. Write to zero for future compatibility.

16.6.68 Outbound Message x Retry Error Threshold Configuration Register (OMxRETCR)

OM[0–1]RETCR Outbound Message 0–1 Retry Error Threshold Configuration Registers — Offset 0x1302C + x*0x100

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

OMxRETCR controls the number of times the message unit can attempt to transfer a message segment to a specific destination before reporting an error. A message segment is retransmitted if a RETRY response is received from the target.

Table 16-111. OM*x*RETCR Field Descriptions

Bits	Reset	Description	Settings
— 31–8	0	Reserved. Write to zero for future compatibility.	
RET 7–0	0	**Retry Error Threshold** The number of times the message unit can attempt to transmit a message segment to a specific target. For proper operation, this field should be modified only when the outbound message controller is not enabled	0x00 Disabled. 0x01 Message segment transmitted only 1 time. 0x02 Message segment transmitted up to 2 times. ... 0xFF Message segment transmitted up to 255 times.

16.6.69 Outbound Message x Multicast Group Registers (OMxMGR)

OM[0–1]MGR Outbound Message 0–1 Multicast Group Registers — Offset 0x13030 + x*0x100

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

OMxMGR contains a multicast group (MG) value and extended multicast group number (EMG) which, in combination with the multicast list register and the multicast enable (OM*x*MR[MM] described in **Table 16-103,** *OMxMR Field Descriptions*, on page 16-168), indicates which device IDs are targets of a multicast operation. The multicast group represents the most significant three bits (bits[7–5]) of the RapidIO device IDs that are destinations of the message operation. The multicast list indicates a list of targets within that group. Each individual bit, when encoded, determines the least significant five bits of the RapidIO device IDs (bits[4–0]) that are targets of the message operation. Therefore, multicast group 0 (MG = 0) contains target device IDs (0, 1, ..., 31), multicast group 1 (MG = 1) contains target device IDs (32, 33, ... 63), and so on.

In large transport mode, the extended multicast group represents the eight most significant bits (bits[15–8]), the multicast group represents the next three bits (bits[7–5]) and the multicast list indicates a list of targets within that group.

If multicast is enabled, this information in the multicast group and mask register is used to determine the target of the message operation instead of the DTGTROUTE and EDTGTROUTE fields in the Outbound Message Destination Port Register (see **Table 16-107,** *OMxDPR Field Descriptions*, on page 16-173).

Table 16-112. OM*x*MGR Field Descriptions

Bits	Name	Description
— 31–11	0	Reserved. Write to zero for future compatibility.

Table 16-112. OM*x*MGR Field Descriptions (Continued)

Bits	Name	Description
EMG 10–3	0	**Extended Multi-Cast Group** The most significant eight bits of the target device IDs for the multicast operation in Large Transport mode. For proper operation, this field should be modified only when the outbound message controller is not enabled
MG 2–0	0	**Multi-Cast Group** The most significant three bits of the target device IDs for the multicast operation. In Large Transport mode, these are the most significant bits of the least significant byte of the Target Device ID. For proper operation, this field should be modified only when the outbound message controller is not enabled

16.6.70 Outbound Message x Multicast List Registers (OMxMLR)

OM[0–1]MLR Outbound Message 0–1 Multicast List Registers — Offset 0x13034 + x*0x100

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

See **Section 16.6.69**, *Outbound Message x Multicast Group Registers (OMxMGR)*, on page 16-178 for more information.

Table 16-113. OM*x*MLR Field Descriptions

Bits	Reset	Description
ML 31–0	0	**Multicast List** The group target list for the message operation. Depending upon the value of the multicast group value, bit 31 corresponds to device ID 0, 32, 64, 96, and so on. Bit 30 corresponds to device ID 1, 33, 65, 97, and so on. If none of the bits are set, bit 31 is assumed to be set. For proper operation, this field should be modified only when the outbound message controller is not enabled.

16.6.71 Inbound Message x Mode Registers (IMxMR)

IM[0–1]MR Inbound Message 0–1 Mode Registers Offset 0x13060 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MIQ_THRESH				—								FRM_SIZ			
TYPE	R/W				R								R/W			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	CIRQ_SIZ				—			QFIE	—	MIQIE	EIE	—			MI	ME
TYPE	R/W				R			R/W	R	R/W		R			R/W	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMxMR allows software to enable the mailbox controller and to control various characteristics of the message operation.

Table 16-114. IM*x*MR Field Descriptions

Bits	Reset	Description	Settings
MIQ_THRESH 31–28	0	**Message in Queue Threshold** Determines the number of message frames to be accumulated in the frame queue before Message In Queue is signaled. Results are undefined if the message in queue threshold is greater than or equal to the message queue size (IMxMR[CIRQ_SIZ]). For proper operation, this field should be modified only when the inbound message controller is not enabled.	0000 1. 0001 2. 0010 4. 0011 8. 0100 16. 0101 32. 0110 64. 0111 128. 1000 256. 1001 512. 1010 1024. 1011–1111 Reserved.
— 27–20	0	Reserved. Write to zero for future compatibility.	
FRM_SIZ 19–16	0	**Message Frame Size** Determines the maximum message size the mailbox can accept without error. Combined with IMxMR[CIRQ_SIZ], this parameter determines the maximum contiguous memory space allocated to the mailbox. For proper operation, this field should be modified only when the inbound message controller is not enabled.	0000–0001 Reserved. 0010 8 bytes. 0011 16 bytes. 0100 32 bytes. 0101 64 bytes. 0110 128 bytes. 0111 256 bytes. 1000 512 bytes. 1001 1024 bytes. 1010 2048 bytes. 1011 4096 bytes. 1100–1111 Reserved.

Table 16-114. IM*x*MR Field Descriptions (Continued)

Bits	Reset	Description	Settings
CIRQ_SIZ 15–12	0	**Circular Frame Queue Size** Determines the number of message frames that can be placed on the circular queue without overflow. Combined with IMxMR[FRM_SIZ], this parameter determines the maximum contiguous memory space allocated to the mailbox. This field should be modified only when the inbound message controller is not enabled.	0000 2. 0001 4. 0010 8. 0011 16. 0100 32. 0101 64. 0110 128. 0111 256. 1000 512. 1001 1024. 1010 2048. 1011–1111 Reserved.
— 11–9	0	Reserved. Write to zero for future compatibility.	
QFIE 8	0	**Queue Full Interrupt Enable** When set, the controller generates an interrupt when the queue is full (that is, the enqueue and dequeue pointers are equal after the mailbox controller increments the dequeue pointer). No QFI interrupt is generated if this if this bit is cleared. If this bit is set and IM*x*SR[QF] = 1, IM*x*SR[QFI] is set.	0 No interrupt is generated. 1 Interrupt generaated on queue full.
— 7	0	Reserved. Write to zero for future compatibility.	
MIQIE 6	0	**Message in Queue Interrupt Enable** When set, the controller enerates an interrupt when the queue has accumulated the number of messages specified by the IM*x*MR[MIQ_THRESH]. No MIQ interrupt is generated if this bit is cleared. If this bit is set and IM*x*SR[MIQ] = 1, IM*x*SR[MIQI] is set. If this bit is set and IM*x*MR[MI] is also set simultaneously, IM*x*SR[MIQI] reflects the value of MIQ after the increment.	0 No interrupt is generated. 1 Interrupt generaated on message in queue event.
EIE 5	0	**Error Interrupt Enable** When set, the controller generates a port-write/error interrupt when a transfer error (IM*x*SR[TE]) or a message request time-out (IM*x*SR[MRT]) event occurs. No port-write/error interrupt is generated if this bit is cleared.	0 No interrupt is generated. 1 Interrupt generaated on error.
— 4–2	0	Reserved. Write to zero for future compatibility.	
MI 1	0	**Mailbox Increment** Software sets this bit after processing an inbound message. Hardware increments the IM*x*FQDPAR and clears this bit. MI always reads as 0.	
ME 0	0	**Mailbox Enable** Set when the mailbox is initialized and can service incoming message operations. If this bit is cleared after the first segment of a multi-segment message arrives, a message request time-out results (IM*x*SR[MRT]). The busy bit (IM*x*SR[MB]) clears if the port response timer value (PRTOCCSR[TV]) is not set to the disabled value. If it is set to the disabled value, the busy bit does not clear.	

16.6.72 Inbound Message x Status Registers (IMxSR)

IM[0–1]SR Inbound Message 0–1 Status Registers Offset 0x13064 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—											QF	—			MIQ
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—					MRT	—		TE	—		QFI	—	MB	QE	MIQI
TYPE	R					W1C	R		W1C	R		W1C	R			W1C
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

IMxSR reports various mailbox conditions during and after a message operation. Writing a value of 1 to the corresponding set bit clears the bit.

Table 16-115. IM*x*SR Field Descriptions

Bits	Reset	Description
— 31–21	0	Reserved. Write to zero for future compatibility.
QF 20	0	**Queue Full** If the queue becomes full, this bit is set. QF is cleared when the queue is not full and if the message controller is disabled. Read only.
— 19–17	0	Reserved. Write to zero for future compatibility.
MIQ 16	0	**Message-In-Queue** If the queue has accumulated the number of messages specified by the IM*x*MR[MIQTH], this bit is set. MIQ is cleared when the number of message in the queue is less than the number specified by IMxMR[MIQ_THRESH] and if the message controller is disabled. Read only.
— 15–11	0	Reserved. Write to zero for future compatibility.
MRT 10	0	**Message Request Time-Out** Set when the message unit has not received another message segment for a multi-segment message and a time-out occurs. This bit is cleared by writing a 1 to it. For proper operation, this bit should be modified only when the inbound message controller is not enabled.
— 9–8	0	Reserved. Write to zero for future compatibility.
TE 7	0	**Transaction Error** Set when an internal error condition occurs during the message operation. TE is cleared by writing a 1 to it. For proper operation, this bit should be modified only when the inbound message controller is not enabled.
— 6–5	0	Reserved. Write to zero for future compatibility.

Table 16-115. IM*x*SR Field Descriptions (Continued)

Bits	Reset	Description
QFI 4	0	**Queue Full Interrupt** If the queue is full and the IMxMR[QFIE] bit is set, QFI is set and an interrupt is generated. QFI s cleared by writing a 1 to it.
— 3	0	Reserved. Write to zero for future compatibility.
MB 2	0	**Mailbox Busy** Indicates that a message operation is in progress. MB is cleared when an error occurs or the message operation finishes. Read only.
QE 1	1	**Queue Empty** If the queue is empty, this bit is set. QE is also set if the message controller is disabled. Read only.
MIQI 0	0	**Message-In-Queue Interrupt** If the queue has accumulated the number of messages specified by the IM*x*MR[MIQ_THRESH] and the IMxMR[MIQIE] bit is set, this bit is set and an interrupt is generated. This bit is cleared by writing a 1 to it.

16.6.73 Inbound Message x Frame Queue Dequeue Pointer Address Registers (IMxFQDPAR)

IM[0–1]FQDPAR Inbound Message 0–1 Frame Queue Dequeue Pointer Address Registers — Offset 0x1306C + x*0x100

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

IMxFQDPAR contains the address for the first message in memory to be processed. Software must initialize this register to the first frame location in memory. When a message is processed, the processor sets IM*x*MR[MI]. The mailbox hardware then increments IM*x*FQDPAR to point to the next frame in memory and clears the IM*x*MR[MI] bit. If the inbound message frame queue enqueue pointer and the inbound message frame queue dequeue pointer are not equal (indicating that the queue is not empty), the processor can read the next message frame from memory for processing. If the enqueue and dequeue pointers are equal after the processor increments them, the queue is empty and all outstanding messages are processed.

When software initializes thesee registers, they must be aligned on a boundary equal to the number of queue entries x frame size. For example, if there are eight entries in the queue and the frame size is 128 bytes, the register must be 1024-byte aligned. The number of queue entries is set in IMxMR[CIRQ_SIZ] and the frame size is set in IMxMR[FRM_SIZ].

Table 16-116. IM*x*FQDPAR Field Descriptions

Bits	Reset	Description
FQDPA 31–3	0	**Frame Dequeue Pointer Address** Contains the address of the first message in memory to process.
— 2–0	0	Reserved. Write to zero for future compatibility.

16.6.74 Inbound Message x Frame Queue Enqueue Pointer Address Registers (IMxFQEPAR)

IM[0–1]FQEPAR Inbound Message x Frame Queue Enqueue Pointer Address Registers — Offset 0x13074 + x*0x100

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

IMxFQEPAR contains the address for the next message frame in memory to be added to the queue. Software must initialize this register to match the frame queue dequeue pointer address. When the mailbox controller receives a message, it writes the message data to the next location in the queue (indicated by the address in IM*x*FQEPAR) and then increments IM*x*FQEPAR to point to the next frame location in memory. This can result in a number of actions:

- If the enqueue and dequeue pointers match, the queue is full and the mailbox controller does not accept any more incoming messages, returning RETRY responses to the sending devices until the queue is no longer full. If the IM*x*MR[QFIE] bit is set, the IM*x*MR[QFI] bit is set and an interrupt is generated.
- If the enqueue and dequeue pointers are the same before the register is incrememted, the queue has changed from empty to not empty. If IM*x*MR[MIQIE] is set, IM*x*SR[MIQI] is set, and an interrupt is generated.

When software initializes these registers, they must be aligned on a boundary equal to the number of queue entries x frame size. For example, if there are eight entries in the queue and the frame size is 128 bytes, the register must be 1024-byte aligned. The number of queues is set in IMxMR[CIRQ_SIZ] and the frame size is set in IMxMR[FRM_SIZ].

Table 16-117. IM*x*FQEPAR Field Descriptions

Bits	Reset	Description
FQEPA 31–3	0	**Frame Queue Enqueue Pointer Address** Contains the address of the next message frame to be added to the queue. For proper operation, this field should be modified only when the inbound message controller is not enabled.
— 2–0	0	Reserved. Write to zero for future compatibility.

16.6.75 Inbound Message x Maximum Interrupt Report Interval Registers (IMxMIRIR)

IM[0–1]MIRIR Inbound Message 0–1 Maximum Interrupt Report Interval Registers — Offsert 0x13078 + x*0x100

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

IMxMIRIR contains a time-out timer value to define the maximum amount of time that the mailbox controller is to wait after transitioning from not empty before signaling an interrupt to the local processor (if enabled) if the IM*x*MR[MIQ_THRESH] limit is not reached. The reset value is the maximum time-out interval, and represents between 3 and 5 seconds.

Table 16-118. IM*x*MIRIR Field Descriptions

Bits	Reset	Description
MIRI 31–8	0xFFFFFF	**Maximum Interrupt Report Interval** Maximum interval between receiving the first message and setting IMxSR[IMIQ]. A value of 0 disables the time-out timer. For proper operation, this field should be modified only when the inbound message controller is not enabled.
— 7–0	0	Reserved. Write to zero for future compatibility.

16.6.76 Outbound Doorbell Mode Register (ODMR)

ODMR Outbound Doorbell Mode Register Offset 0x13400

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

ODMR allows software to start a doorbell operation and to control the characteristics of various doorbell operations.

Table 16-119. ODMR Field Descriptions

Bits	Reset	Description
— 31–1	0	Reserved. Write to zero for future compatibility.
DUS 0	0	**Doorbell Unit Start** A 0-to-1 transition when the doorbell unit is not busy (ODSR[DUB] bit has a value of 0) starts the doorbell unit. A 1-to-0 transition has no effect.

16.6.77 Outbound Doorbell Status Register (ODSR)

ODSR Outbound Doorbell Status Register Offset 0x13404

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—			MER	RETE	PRT	—							DUB	EODI	—
TYPE	R			W1C			R							R	W1C	R
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ODSR reports various doorbell conditions during and after a doorbell operation. Writing a 1 to the corresponding set bit clears the bit.

Table 16-120. ODSR Field Descriptions

Bits	Reset	Description
— 31–13	0	Reserved. Write to zero for future compatibility.
MER 12	0	**Message Error Response** Set when an error response is received from the doorbell target. The error response reserved field indicates the value of the error response status bits when an error response is received. MER is cleared by writing a 1 to it. For proper operation, this bit should be cleared only when a doorbell operation is not in progress.
RETE 11	0	**Retry Error Threshold Exceeded** Set when the doorbell unit cannot complete a doorbell operation because the retry error threshold value has been exceeded due to a RapidIO retry response. RETE is cleared by writing a 1 to it. For proper operation, this bit should be cleared only when a doorbell operation is not in progress.
PRT 10	0	**Packet Response Time-Out** Set when the doorbell unit cannot complete a doorbell operation and a packet response time-out occurs. PRT is cleared by writing a 1 to it. For proper operation, this bit should be cleared only when a doorbell operation is not in progress.
— 9–3	0	Reserved. Write to zero for future compatibility.
DUB 2	0	**Doorbell Unit Busy** Indicates that a doorbell operation is in progress. DUB is cleared when an error occurs or the doorbell operation is finishes. Read only.
EODI 1	0	**End-of-Doorbell Interrupt** When a doorbell operation finishes and the ODDATR[EODIE] bit is set, this bit is set and an interrupt is generated. EODI is cleared by writing a 1 to it. For proper operation, this bit should be cleared only when a doorbell operation is not in progress.
— 0	0	Reserved. Write to zero for future compatibility.

16.6.78 Outbound Doorbell Destination Port Register (ODDPR)

ODDPR Outbound Doorbell Destination Port Register Offset 0x13418

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

ODDPR indicates the RapidIO destination device ID to which the doorbell unit controller is to send data. Software must ensure that this is a valid port in the receiving device.

Table 16-121. ODDPR Field Descriptions

Bits	Reset	Description
EDTROUTE 31–24	0	**Extended Destination Target Route** Most significant byte of a 16-bit target route (device ID of the target) in Large Transport mode. In Small Transport mode, this bit is reserved. For proper operation, this field should be modified only when a doorbell operation is not in progress.
DTROUTE 23–16	0	**Destination Target Route** Contains the target route field of the transaction (device ID of the target). For proper operation, this field should be modified only when a doorbell operation is not in progress.
— 15–0	0	Reserved. Write to zero for future compatibility.

16.6.79 Outbound Doorbell Destination Attributes Register (ODDATR)

ODDATR Outbound Doorbell Destination Attributes Register Offset 0x1341C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		EODIE	—	DTFLOWLVL		—		TGINT				—			
TYPE	R/w															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	INFO_MSB								INFO_LSB							
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ODDATR contains the transaction attributes to be used for the doorbell operation.

Table 16-122. ODDATR Field Descriptions

Bits	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility. Bit 30 is hardwired to 0.	
EODIE 29	0	**End-of-Doorbell Interrupt Enable** Generates an interrupt when the current doorbell operation finishes. This field should be modified only when a doorbell operation is not in progress.	
— 28	0	Reserved. Write to zero for future compatibility.	
DTFLOWLVL 27–26	0	**Transaction Flow Level** Specifies the transaction flow level. This field should be modified only when a doorbell operation is not in progress.	00 Lowest-priority transaction flow. 01 Next highest priority transaction flow. 10 Highest-priority transaction flow. 11 Reserved.
— 25–24	0	Reserved. Write to zero for future compatibility.	
TGINT 23–20	0	**Target Interface** This field should always be cleared to RapidIO (0x0).	
— 19–16	0	Reserved. Write to zero for future compatibility.	
INFO_MSB 15–8	0	**MSB of Doorbell INFO** Most significant byte of the doorbell INFO field. This field should be modified only when a doorbell operation is not in progress.	
INFO_LSB 7–0	0	**LSB of Doorbell INFO** Least significant byte of the doorbell INFO field. This field should be modified onlyy when a doorbell operation is not in progress.	

16.6.80 Outbound Doorbell Retry Error Threshold Configuration Register (ODRETCR)

ODRETCR Outbound Doorbell Retry Error Threshold Configuration Register — Offset 0x1342C

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

ODRETCR controls the number of times the doorbell unit attempts to complete a doorbell operation before reporting an error and moving on to the next task, if available. Failures to complete an operation are indicated when the doorbell unit receives a RapidIO logical layer RETRY response from the target. If the programmed count is exceeded, the ODSR[RETE] bit is set.

Table 16-123. ODRETCR Field Descriptions

Bits	Name	Description	Settings
— 31–8	0	Reserved. Write to zero for future compatibility.	
RET 7–0	0	**Retry Error Threshold** Specifies the number of times the doorbell unit attempts to transmit a doorbell message. This field should be modified only when a doorbell operation is not in progress.	0x00 Disabled. 0x01 Doorbell transmitted only 1 time. 0x02 Doorbell transmitted up to 2 times ... 0xFF - Doorbell transmitted up to 255 times.

16.6.81 Inbound Doorbell Mode Registers (IDMR)

IDMR Inbound Doorbell Mode Registers Offset 0x13460

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DIQ_THRESH				—											
TYPE	R/W				R											
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	CIRQ_SIZ				—			QFIE	—	DIQIE	EIE	—			DI	DE
TYPE	R/W				R			R/W	R	R/W		R			R/W	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IDMR allows software to enable the inbound doorbell controller and to control the characteristics of various doorbell operations.

Table 16-124. IDMR Field Descriptions

Bits	Reset	Description	Settings
DIQ_THRESH 31–28	0	**Doorbell-in-Queue Threshold** Determines the number of doorbells to accumulate in the doorbell queue before the doorbell-in-queue bit is set (IDSR[DIQ]). Undefined operation results if the actual number of entries in the doorbell-in-queue threshold is set as greater than or equal to the actual size of the doorbell queue. For proper operation, this field should be modified only when the doorbell controller is not enabled.	0000 1. 0001 2. 0010 4. 0011 8. 0100 16. 0101 32. 0110 64. 0111 128. 1000 256. 1001 512. 1010 1024. 1011–1111 Reserved.
— 27–16	0	Reserved. Write to zero for future compatibility.	
CIRQ_SIZ 15–12	0	**Circular Doorbell Queue Size** Determines the number of doorbell entries that can be placed in the circular queue. CIRQ_SIZ × 8 bytes determine the maximum contiguous memory space allocated to the doorbell unit. For proper operation, this field should be modified only when the doorbell controller is not enabled.	0000 2. 0001 4. 0010 8. 0011 16. 0100 32. 0101 64. 0110 128. 0111 256. 1000 512 (4 KB page boundary). 1001 1024. 1010 2048. 1011–1111 Reserved.

Table 16-124. IDMR Field Descriptions (Continued)

Bits	Reset	Description	Settings
— 11–9	0	Reserved. Write to zero for future compatibility.	
QFIE 8	0	**Queue Full Interrupt Enable** When set, enables the queue full interrupt. The queue is full when the enqueue and dequeue pointers are equal after the doorbell controller increments the dequeue pointer, the controller genrates the Queue Full interrupt. That is, if this bit is set and IDSR[QF] = 1, IDSR[QFI] is set. For proper operation, this field should only be modified when the inbound doorbell controller is not enabled.	0 No QF interrupt is generated. 1 Generates an interrupt when the queue is full .
— 7	0	Reserved. Write to zero for future compatibility.	
DIQIE 6	0	**Doorbell in Queue Interrupt Enable** When set, enables the doorbell in queue interrupt. The interrupt cannot be asserted if this bit is cleared. If this bit is set and IDSR[DIQ] = 1, IDSR[DIQI] is set. If this bit is set and IDMR[DI] is set simultaneously, IDSR[DIQI] reflects the value of DIQ after the increment. For proper operation, this field should only be modified when the inbound doorbell controller is not enabled.	0 No DIQ interrupt is generated. 1 Generates an interrupt when IDSR[DIQ] = 1.
EIE 5	0	**Error Interrupt Enable** When set, enables the port-write/error interrupt when a transfer error (IDSR[TE]) event occurs. No port-write/error interrupt is generated if this bit is cleared. For proper operation, this field should only be modified when the inbound doorbell controller is not enabled.	
— 4–2	0	Reserved Reserved. Write to zero for future compatibility.	
DI 1	0	**Doorbell Increment** Software sets this bit after an inbound doorbell is processed. Hardware then increments the ODQEPAR and clears this bit. DI always reads as 0.	
DE 0	0	**Doorbell Enable** Enabled/disables the inbound doorbell operations.	0 Inbound doorbell disabled. 1 The inbound doorbell is initialized and can service incoming doorbell operations.

16.6.82 Inbound Doorbell Status Register (IDSR)

IDSR Inbound Doorbell Status Register Offset 0x13464

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
												QF	—			DIQ
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—								TE	—		QFI	—	DB	QE	DIQI
TYPE	R								W1C	R		W1C	R			W1C
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

IDSR reports various doorbell conditions after a doorbell operation. Writing a 1 to the corresponding set bit clears the bit.

Table 16-125. IDSR Field Descriptions

Bits	Reset	Description
— 31–21	0	Reserved. Write to zero for future compatibility.
QF 20	0	**Queue Full** If the queue is full, this bit is set. This bit is cleared when the queue is not full or if the doorbell controller is disabled. Read only.
— 19–17	0	Reserved. Write to zero for future compatibility.
DIQ 16	0	**Doorbell-In-Queue** If the queue has accumulated the number of doorbells specified by DIQ_THRESH, then this bit is set. Also, if a valid entry pointed to by the dequeue address pointers has not been serviced within the configured maximum interval, this bit is set. This bit is cleared if the above conditions are not met or the doorbell controller is disabled. Read-only.
— 15–8	0	Reserved. Write to zero for future compatibility.
TE 7	0	**Transaction Error** Set when an internal error occurs during the doorbell operation. To reset TE, write a value of 1 to clear it. For proper operation, this bit should be modified only when the doorbell controller is not enabled..
— 6–5	0	Reserved. Write to zero for future compatibility.
QFI 4	0	**Queue Full Interrupt** If the queue becomes full and the QFIE bit in the Mode Register is set, this bit is set and an interrupt is generated. This bit is cleared by writing a 1 to it. For proper operation, this bit should be modified only when the doorbell controller is not enabled..
— 3	0	Reserved. Write to zero for future compatibility.
DB 2	0	**Doorbell Busy** Indicates that a doorbell has been received and the doorbell queue is being written. This bit is cleared when the write to memory finishes. Disabling the doorbell controller does not affect this bit. Read only.

Table 16-125. IDSR Field Descriptions (Continued)

Bits	Reset	Description
QE 1	1	**Queue Empty** If the queue is empty, then this bit is set. This bit will also be set if the doorbell controller is disabled. Read only.
DIQI 0	0	**Doorbell-In-Queue Interrupt** If DIQ is set and IDMR[DIQIE} is set, the controller sets this bit and generates an interrupt. This bit is cleared by writing a 1 to it.

16.6.83 Inbound Doorbell Queue Dequeue Pointer Address Register (IDQDPAR)

IDQDPAR Inbound Doorbell Queue Dequeue Pointer Address Registers Offset 0x1346C

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

IDQDPAR contains the double-word address for the first doorbell in memory to be processed. Software must initialize this register to the first doorbell entry location in memory. After a doorbell is processed, the processor sets IDMR[DI]. Then the doorbell queue dequeue pointer address register is incremented by hardware to point to the next doorbell entry in memory and IDMR[DI] is cleared.

When processing multiple doorbells, the processor can write this register directly instead of setting IDMR[DI] for each doorbell. If the enqueue pointer and the dequeue pointer are not equal (indicating that the queue is not empty), the processor can read the next doorbell from memory for processing. If the enqueue and dequeue pointers are equal after the processor increments the IDQDPAR, the queue is empty, and all outstanding doorbells have been processed.

Table 16-126. IDQDPAR Field Descriptions

Bits	Reset	Description
DQDPA 31–3	0	**Doorbell Dequeue Pointer Address** Contains the double-word address of the first doorbell in memory to process.
— 2–0	0	Reserved. Write to zero for future compatibility.

16.6.84 Inbound Doorbell Queue Enqueue Pointer Address Registers (IDQEPAR)

IDQEPAR Inbound Doorbell Queue Enqueue Pointer Address Registers Offset 0x13474

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

IDQEPAR contains the double-word address for the next doorbell entry in memory to be added to the queue. Software must initialize IDQEPAR to match the doorbell queue dequeue pointer address. When a doorbell packet is received by the doorbell controller, it writes the doorbell information to the next location in the queue (indicated by the address in IDQEPAR) and then increments IDQEPAR to point to the next doorbell location in memory. This can result in a number of actions:

- If the enqueue and dequeue pointers match, then the queue is now full and the doorbell controller will not accept any more incoming doorbell packets, returning RETRY responses to the sending devices until the queue is no longer full. If the IDMR[QFIE] bit is set, then the IDSR[QFI] is set and the interrupt is generated.
- If the enqueue and dequeue pointers were the same before receiving the doorbell, the queue has changed from empty to not empty. When the number of doorbells received matches the configured threshold, the IDSR[DIQ] bit is set. If the IDMR[DIQIE] bit is set, then the IDSR[DIQI] bit is also set and the inbound doorbell interrupt is generated.

Table 16-127. IDQEPAR Field Descriptions

Bits	Name	Description
DQEPA 31–3	0	**Doorbell Queue Enqueue Pointer Address** Contains the double-word address of the next doorbell location to be added to the queue. For proper operation, this field should be written only when the doorbell controller is disabled.
— 2–0	0	Reserved. Write to zero for future compatibility.

16.6.85 Inbound Doorbell Maximum Interrupt Report Interval Register (IDMIRIR)

IDMIRIR Inbound Doorbell Maximum Interrupt Report Interval Register — Offset 0x13478

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

IDMIRIR contains a time-out timer value to define the maximum amount of time that a doorbell entry can be at the head of the doorbell queue before generating an interrupt if the IDMR[DIQ_THRESH] limit is not reached. The reset value is the maximum time-out interval and represents 3–6 seconds.

Table 16-128. IDMIRIR Field Descriptions

Bits	Name	Description
MIRI 31–8	0xFFFFFF	**Maximum Interrupt Report Interval** Maximum interval between the time a doorbell message goes into the queue until an interrupt is generated. A value of 0 disables the timer. This field should be written only when the doorbell controller is disabled.
— 7–0	0	Reserved. Write to zero for future compatibility.

16.6.86 Inbound Port-Write Mode Register (IPWMR)

IPWMR Inbound Port-Write Mode Register Offset 0x134E0

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

IPWMR allows software to enable the port-write controller and to control various characteristics of port-write operations.

Table 16-129. IPWMR Field Descriptions

Bits	Reset	Description
— 31–9	0	Reserved. Write to zero for future compatibility.
QFIE 8	0	**Queue Full Interrupt Enable** When set, enable a error/port-write interrupt when the queue is full; that is, the controller has written the port-write data payload into memory. No interrupt is generated if this if this bit is cleared. For proper operation, this field should be modified only when the port-write controller is not enabled.
— 7–6	0	Reserved. Write to zero for future compatibility.
EIE 5	0	**Error Interrupt Enable** When set, enables a port-write/error interrupt when a transfer error (IPW0SR[TE]) event occurs. No interrupt is generated if this bit is cleared. For proper operation, this field should be modified only when the port-write controller is not enabled.
— 4–2	0	Reserved. Write to zero for future compatibility.
CQ 1	0	**Clear Queue** Software sets this bit after processing an inbound port-write operation. Hardware clears the queue full bit (IPWSR[QF]), clears this bit, and allows another port-write to be received. This bit is always read as a 0.
PWE 0	0	**Port Write Enable** If this bit is set the port-write controller is initialized and can service an incoming operation.

16.6.87 Inbound Port-Write Status Register (IPWSR)

IPWSR Inbound Port-Write Status Register Offset 0x134E4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—											QF	—			
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—								TE	—		QFI	PWD	PWB	—	
TYPE	R								W1C	R		W1C		R		
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IPWSR reports various port-write conditions.

Table 16-130. IPWSR Field Descriptions

Bits	Reset	Description	Settings
— 31–21	0	Reserved. Write to zero for future compatibility.	
QF 20	0	**Queue Full** Set when the queue becomes full. QF is cleared when the clear queue bit is set (IPWMR[CQ]) and the queue is not full. This bit is cleared when the queue is not full or if the port-write controller is disabled. Read only.	
— 19–16	0	Reserved. Write to zero for future compatibility.	
TE 15	0	**Transaction Error** Set when an internal error condition occurs during the port-write operation. Write a value of 1 to TE to clear it. This bit should be modified only when the port-write controller is not enabled.	
— 6–5	0	Reserved. Write to zero for future compatibility.	
QFI 4	0	**Queue Full Interrupt** If the queue is full and the IPWMR[QFIE] bit is set, this bit is set and an interrupt generated. This bit is cleared by writing a 1 to it.	
PWD 3	0	**Port-Write Discarded** Set when a port-write is discarded while the port-write controller is enabled but busy. This bit is cleared by writing a 1 to it.	
PWB 2	0	**Port-Write Busy** Indicates a port-write busy condition. Disabling the port-write controller does not affect this bit. Read only.	0 Port-write payload has been written to memory. 1 A port-write has been received and the port-write payload is being written to memory.
— 1–0	0	Reserved. Write to zero for future compatibility.	

16.6.88 Inbound Port-Write Queue Base Address Register (IPWQBAR)

IPWQBAR Inbound Port-Write Queue Base Address Register Offset 134EC

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

IPWQBAR contains the 64-byte cache line address for the port-write data payload. Software must initialize this register to the desired location in memory.

Table 16-131. IPWQBAR Field Descriptions

Bits	Reset	Description
PWQBA 31–6	0	**Port-Write Queue Base Address** Contains the address of the port-write data payload. This field should be written only when the port-write controller is disabled
— 5–0	0	Reserved. Write to zero for future compatibility.

RapidIO Interface Dedicated DMA Controller

17

The MSC8144E includes a dedicated DMA controller that transfers blocks of data between the serial RapidIO controller and the local address space independent from the DSP cores.

Figure 17-1 shows the block diagram of the dedicated DMA controller.

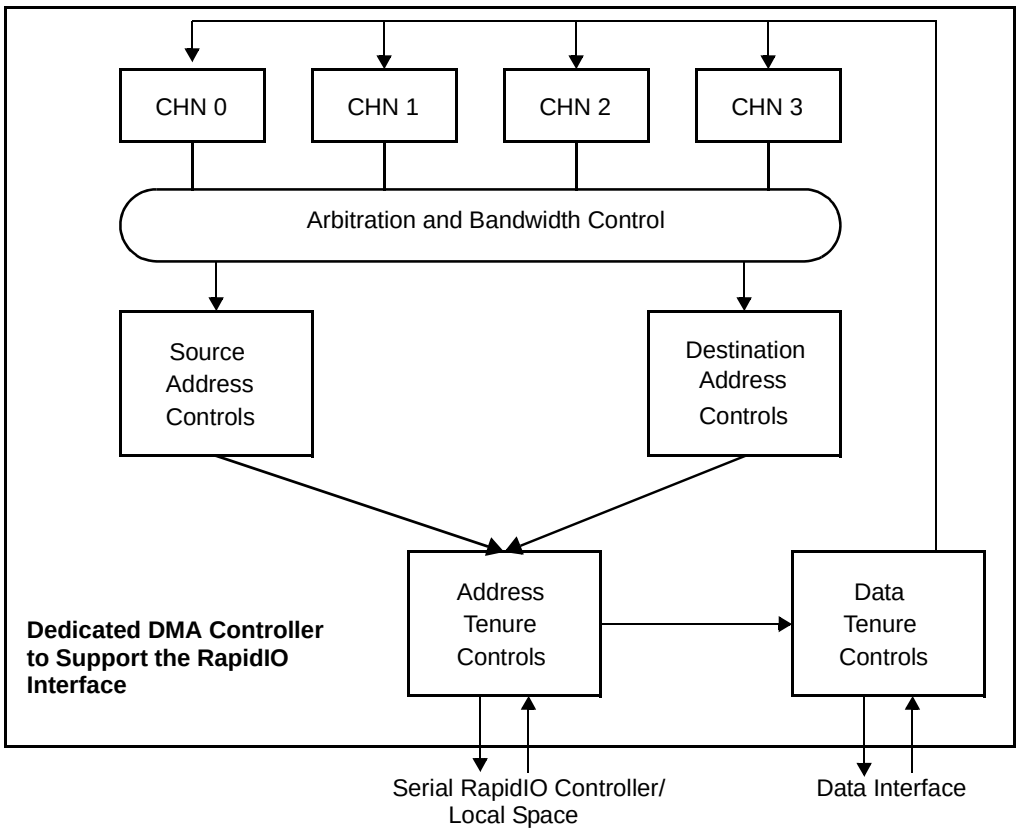


Figure 17-1. RapidIO Interface Dedicated DMA Controller Block Diagram

17.1 Overview

The dedicated DMA controller has four high-speed DMA channels. Each one 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.

17.1.1 Features

The dedicated DMA controller offers the following features:

- Four high-speed/high-bandwidth channels accessible by local and remote masters
- Basic DMA operation modes (direct, simple chaining)
- Extended DMA operation modes (advanced chaining and stride capability)
- Cascading descriptor chains
- Misaligned transfers
- Programmable bandwidth control between channels
- Three priority levels supported for source and destination transactions
- Interrupt on error and completed segment, list, or link
- An Address Translation Management Unit (ATMU) with 10 local access address windows. The ATMU translates a request address into a logical device source/destination.

17.1.2 Modes of Operation

The MPC8144E 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 17-1** summarizes the relationship between the modes and the following features:

- *Direct mode*. No descriptors are involved. Software must initialize the required fields as described in **Table 17-1** before starting a transfer.
- *Chaining mode*. Software must initialize descriptors in memory and the required fields as described in **Table 17-1** before starting a transfer.

- *Single-write start mode*. The DMA process can be started by using a single-write command to either the descriptor address register in one of the chaining modes or the source/destination address registers in one of the direct modes.
- *External control capability*. This allows an external agent to start, pause, and check the status of a DMA transfer that has already been initialized.
- *Channel continue capability*. The channel continue capability allows software the flexibility of having the DMA controller start with descriptors that have already been programmed while software continues to build more descriptors in memory.
- *Channel abort capability*. The software can abort a previously initiated transfer by setting the bit MR*n*[CA]. The DMA controller terminates all outstanding transfers initiated by the channel without generating any errors before entering an idle state.

Table 17-1. Relationship of Modes and Features

Mode	Mode with One Additional Feature	Mode with Two Additional Features
B (Basic)	BD (basic direct)	BDS (BD single-write start)
	BC (basic chaining)	BCS (BC single-write start)
Ext (Extended)	ExtD (extended direct)	ExtDS (ExtD single-write start)
	ExtC (extended chaining)	ExtCS (ExtC single-write start)

Refer to **Section 17.2**, *Functional Description* for details on these modes. **Figure 17-2** shows the general DMA operational flow chart.

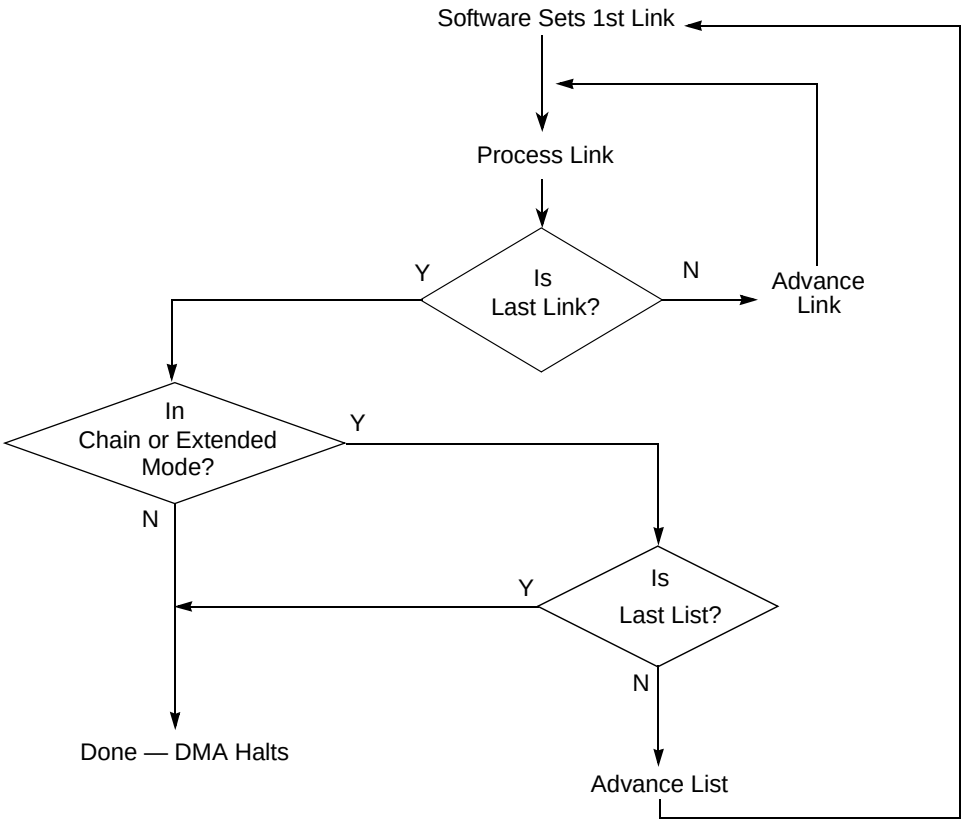


Figure 17-2. DMA Operational Flow Chart

17.2 Functional Description

This section describes the function of the DMA controller.

17.2.1 DMA Channel Operation

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 17-2** for more complete descriptions of the channel states and state transitions.

17.2.1.1 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.

17.2.1.1.1 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 17-2**), to confirm that the specific DMA channel is idle.
2. Initialize SAR*n*, SATR*n*, DAR*n*, DATR*n* and BCR*n*.
3. Set the mode register channel transfer mode bit, MR*n*[CTM], to indicate direct mode. Other control parameters may also be initialized in the mode register.
4. Clear then set the mode register channel start bit, MR*n*[CS], to start the DMA transfer.
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after the transfer is finished, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if a transfer error occurs.
7. End of segment interrupt is generated if MR*n*[EOSIE] is set.

17.2.1.1.2 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 17-2**), to confirm that the specific DMA channel is idle.
2. Initialize the source attributes (SATR*n*), DATR*n*, and BCR*n* registers.

3. Set the mode register channel transfer mode bit, MR*n*[CTM], and the single-write start direct mode bit, MR*n*[SRW]. Other control parameters may also be initialized in the mode register. Set MR*n*[CDSM/SWSM] for transfers started using SAR*n*. Clear MR*n*[CDSM/SWSM] for transfers started using the DAR*n*.
4. A write to the source or destination address register starts the DMA transfer and automatically sets MR*n*[CS].
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after the transfer is finished, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if a transfer error occurs.
7. End of segment interrupt is generated if MR*n*[EOSIE] is set.

17.2.1.1.3 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 17-2**), to confirm that the specific DMA channel is idle.
3. Initialize CLNDAR*n* to point to the first link descriptor in memory.
4. Clear the mode register channel transfer mode bit, MR*n*[CTM], as well as MR*n*[XFE], to indicate basic chaining mode. Other control parameters may also be initialized in the mode register.
5. Clear, then set the mode register channel start bit, MR*n*[CS], to start the DMA transfer.
6. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
7. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

17.2.1.1.4 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, MR*n*[CDSM/SWSM], and the extended features enable bit MR*n*[XFE]. Also, clear the channel transfer mode bit, MR*n*[CTM]. This initialization indicates basic chaining and single-write start mode. Also other control parameters may be initialized in the mode register.
2. Build link descriptor segments in memory.
3. Poll the channel state (see **Table 17-2**), to confirm that the specific DMA channel is idle.
4. Initialize CLNDAR*n* to point to the first descriptor segment in memory. This write automatically causes the DMA controller to begin the link descriptor fetch and set MR*n*[CS].
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

17.2.1.1.5 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].

17.2.1.1.6 Extended Direct Mode

Extended direct mode has the same functionality as basic direct mode with the addition of stride capabilities. The bit settings are the same as in direct mode with the exception of the MR*n*[XFE] being set. Striding on the source address can be accomplished by setting SATR*n*[SSME] and setting the desired stride size and distance in SSR*n*. Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in DSR*n*.

17.2.1.1.7 Extended Direct Single-Write Start Mode

Extended direct single-write start mode has the same functionality as the basic direct single-write start mode with the addition of stride capabilities. The bit settings are also the same with the exception of MR*n*[XFE] being set. Striding on the source address can be accomplished by setting SATR*n*[SSME] and setting the desired stride size and distance in SSR*n*. Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in DSR*n*.

17.2.1.1.8 Extended Chaining Mode

In extended chaining mode, the software must first build list and link descriptor segments in memory. Then CLSDAR*n* must be initialized to point to the first list descriptor in memory. The DMA controller loads list descriptors and link descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the link descriptor information loaded. Once the current link descriptor is finished, the DMA controller reads the next link descriptor from memory and begins another DMA transfer. If the current link descriptor is the last in the list, the DMA controller reads the next list descriptor in memory. The transfer is finished if the current link descriptor is the last one in the last list in memory or if an error condition occurs. The sequence of events to start and complete a transfer in extended chaining mode is as follows:

1. Build link and list descriptor segments in memory.
2. Poll the channel state (see **Table 17-2**), to confirm that the specific DMA channel is idle.
3. Initialize CLSDAR*n* to point to the first list descriptor in memory.
4. Clear the mode register channel transfer mode bit, MR*n*[CTM], to indicate chaining mode. MR*n*[XFE] must be set to indicate extended DMA mode. Other control parameters may also be initialized in the mode register.
5. Clear, then set the mode register channel start bit, MR*n*[CS], to start the DMA transfer.
6. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
7. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

17.2.1.1.9 Extended Chaining Single-Write Start Mode

In the extended mode, the single-write start feature allows a chain to be started by writing the current list descriptor pointer. Setting MR*n*[CDSM/SWSM] causes MR*n*[CS] to be set automatically when CLSDAR*n* is written. The sequence of events to start and complete an extended chain using single-write start mode is as follows:

1. Set MR*n*[CDSM/SWSM], MR*n*[CTM], and MR*n*[XFE] to indicate extended chaining and single-write start mode. Also other control parameters may be initialized in the mode register.
2. Build list and link descriptor segments in local memory.
3. Poll the channel state (see **Table 17-2**), to confirm that the specific DMA channel is idle.
4. Initialize the current list descriptor address register to point to the first list descriptor segment in memory. This write automatically causes the DMA controller to begin the list descriptor fetch and set MR*n*[CS].

5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

17.2.1.2 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.

17.2.1.2.1 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.

17.2.1.2.2 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.

17.2.1.3 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.

17.2.1.4 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.

17.2.1.5 Channel State

Table 17-2 defines the state of a channel based on the values of the channel start (MR*n*[CS]), channel busy (SR*n*[CB]), transfer error (SR*n*[TE]), and channel continue (MR*n*[CC]) bits.

Table 17-2. Channel State Table

MR*n*[CS]	SR*n*[CB]	SR*n*[TE]	MR*n*[CC]	Channel State
0	0	0	0	Idle state. This is the state of the bits out of reset.
0	0	0	1	Channel continue unexpected. Channel remains idle
0	0	1	0	Error occurred after software halted the channel.
0	0	1	1	Channel Continue unexpected. Channel remains in error halt state
0	1	0	0	Software halted channel. The channel was busy and software cleared MR*n*[CS].
0	1	0	1	Channel remains in halt state.
—	1	1	—	The channel has encountered an error condition and it is trying to halt.
1	0	0	0	Ready to start a transfer, or transfer completed
1	0	0	1	Continue transfer (only meaningful in chaining mode, not direct mode). In direct mode, the channel continue has no effect.
1	0	1	0	Error occurred during transfer
1	0	1	1	Channel remains in error halt state

Table 17-2. Channel State Table (Continued)

MR*n*[CS]	SR*n*[CB]	SR*n*[TE]	MR*n*[CC]	Channel 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

17.2.1.6 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 17-3** 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.

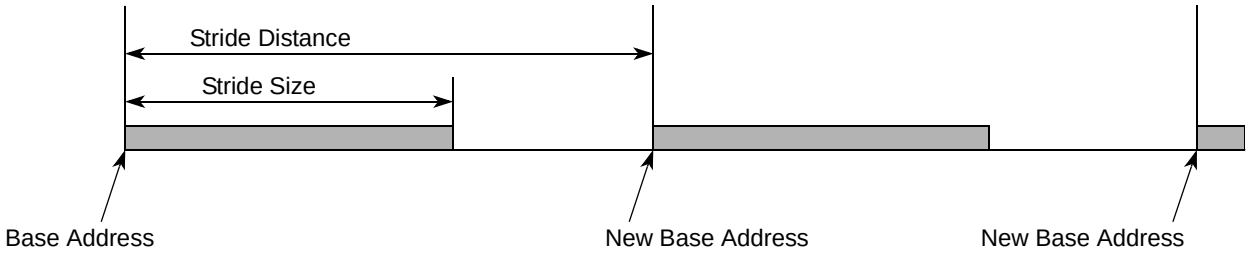


Figure 17-3. Stride Size and Stride Distance

17.2.2 DMA Transfer Interfaces

The DMA can be used to achieve data transfers across the entire memory map.

17.2.3 DMA Errors

On a transfer error (uncorrectable ECC errors on memory accesses, parity errors on local bus or PCI, 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.

17.2.4 DMA Descriptors

The DMA engine recognizes list descriptors and link descriptors. List descriptors connect lists of link descriptors. Link descriptors describe the DMA activity that is to take place. DMA descriptors are built in either local or remote memory and are connected by the next descriptor fields. Only link descriptors contain information for the DMA controller to transfer data. Software must ensure that each descriptor is 32-byte aligned. The last link descriptor in the last list in memory sets the EOLND bit in the next link descriptor; the next list descriptor fields indicating that these are the last descriptors in memory. Software initializes the current list descriptor address register to point to the first list descriptor in memory. The DMA controller traverses through the descriptor lists until the last link descriptor is met. For each link descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by that descriptor. 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 17-3 summarizes the DMA list descriptors.

Table 17-3. List DMA Descriptor Summary

Descriptor Field	Description
Next list descriptor extended address	Points to the next list descriptor in memory. After the DMA controller reads the descriptor from memory, this field is loaded into the next list descriptor extended address registers.
Next list descriptor address	Points to the next list descriptor in memory. After the DMA controller reads the descriptor from memory, this field is loaded into the next list descriptor address registers.
First link descriptor extended address	Points to the first link descriptor in memory for this list. After the DMA controller reads the descriptor from memory, this field is loaded into the current link descriptor extended address registers.
First link descriptor address	Points to the first link descriptor in memory for this list. After the DMA controller reads the descriptor from memory, this field is loaded into the current link descriptor address registers.
Source stride	Contains the stride information used for the data source if striding is enabled for a link in the list
Destination stride	Contains the stride information used for the data destination if striding is enabled for a link in the list

Table 17-4 summarizes the DMA link descriptors.

Table 17-4. 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 17-4 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 17-4. List Descriptor Format (Used for 32-bit devices)

Figure 17-5 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 17-5. List Descriptor Format (Used for 36-bit devices)

)**Figure 17-4** 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 17-6. Link Descriptor Format (Used for 32-bit devices

Figure 17-7 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 17-7. Link Descriptor Format (Used for 36-bit devices)

17.2.5 Local Access ATMU Registers

The local access ATMU has ten address windows that can map a DMA request to a programmable transaction interface (logical device). The user set up the ATMU windows correctly before enabling translation through the ATMU. In a multiple hit scenario, the first matching window is used. If no hit is detected or if mapping is to a reserved transaction interface, the source/target defaults to local address space.

Note: See **Section 17.3.1**, *Local Access Window Base Address Registers 0–9 (LAWBAR[0–9])*, on page 17-17 and **Section 17.3.2**, *Local Access Window Attributes Registers 0–9 (LAWAR[0–9])*, on page 17-18 for details.

17.2.6 Limitations and Restrictions

This section addresses some of the limitations and restrictions of the dedicated DMA controller and is intended to help software maximize the DMA performance and avoid DMA programming errors.

The limitations of the dedicated DMA controller are the following:

- 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.

The DMA controller restrictions are as follows:

- Setting the source or destination priority level (STFLOWLVL or DTFLOWLVL) to a value of three (0b11) is considered a programming error.
- 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.
- If the DMA is programmed to send messages over RapidIO, the programmer must ensure that the message length (BCR*n*[BC]) is 8, 16, 32, 64, 128, or 256 bytes. This can be achieved by setting the byte count register (BCR) to a power of 2 value equal to 8 or greater.
- 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.

17.3 Dedicated DMA Controller Programming Model

The following sections describe the dedicated DMA controller registers. The majority of these registers are channel-specific and can be identified by one of the four offsets that describe the register. These registers include the following:

- Local Access Window Base Address Registers 0–9 (LAWBAR[0–9]), page 17-17
- Local Access Window Attributes Registers 0–9 (LAWAR[0–9]), page 17-18
- DMA 0–3 Mode Registers (MR[0–3]), page 17-20
- DMA 0–3 Status Registers (SR[0–3]), page 17-23
- DMA 0–3 Current Link Descriptor Extended Address Registers (ECLNDAR[0–3], page 17-25
- DMA 0–3 Current Link Descriptor Address Registers (CLNDAR[0–3]), page 17-26
- DMA 0–3 Source Attributes Registers (SATR[0–3]), page 17-27
- DMA 0–3 Source Address Registers (SAR[0–3]), page 17-29
- DMA 0–3 Destination Attributes Registers (DATR[0–3]), page 17-30
- DMA 0–3 Destination Address Registers (DAR[0–3]), page 17-32
- DMA 0–3 Byte Count Registers (BCR[0–3], page 17-33
- DMA 0–3 Next Link Descriptor Extended Address Registers (ENLNDAR[0–3], page 17-34
- DMA 0–3 Next Link Descriptor Address Registers (NLNDAR[0–3]), page 17-35
- DMA 0–3 Current List Descriptor Extended Address Registers (ECLSDAR[0–3], page 17-36
- DMA 0–3 Current List Descriptor Address Registers (CLSDAR[0–3]), page 17-37
- DMA 0–3 Next List Descriptor Extended Address Registers (ENLSDAR[0–3], page 17-38
- DMA 0–3 Next List Descriptor Address Registers (NLSDAR[0–3]), page 17-39
- DMA 0–3 Source Stride Registers (SSR[0–3]), page 17-40
- DMA 0–3 Destination Stride Registers (DSR[0–3]), page 17-41
- DMA General Status Register (DGSR), page 17-42

Note: The dedicated DMA registers use a base address of 0xFFFA2000.

17.3.1 Local Access Window Base Address Registers 0–9 (LAWBAR[0–9])

LAWBAR0	Local Access Window Base Address Registers 0–9	Offset 0x1C08
LAWBAR1		Offset 0x1C28
LAWBAR2		Offset 0x1C48
LAWBAR3		Offset 0x1C68
LAWBAR4		Offset 0x1C88
LAWBAR5		Offset 0x1CA8
LAWBAR6		Offset 0x1CC8
LAWBAR7		Offset 0x1CE8
LAWBAR8		Offset 0x1D08
LAWBAR9		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

LAWBARx holds 24 most significant bits of the window base address. The least significant byte is always 0s. **Table 17-5** describes the LAWBARx fields.

Table 17-5. LAWBARx 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 window base address. **Note:** For local transactions, the most significant 4 bits in this field must be 0s.	

17.3.2 Local Access Window Attributes Registers 0–9 (LAWAR[0–9])

LAWAR0	Local Access Window Attributes Registers 0–9	Offset 0x1C10
LAWAR1		Offset 0x1C30
LAWAR2		Offset 0x1C50
LAWAR3		Offset 0x1C70
LAWAR4		Offset 0x1C90
LAWAR5		Offset 0x1CB0
LAWAR6		Offset 0x1CD0
LAWAR7		Offset 0x1CF0
LAWAR8		Offset 0x1D10
LAWAR9		Offset 0x1D30

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

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

LAWARx contains the transaction interface for a request mapped to this window. **Table 17-6** describes the LAWARx fields.

Table 17-6. LAWARx 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.	1100 RapidIO interface 0. 1111 Local address space. All others reserved.
— 19–6	0	Reserved. Write to zero for future compatibility.	

Table 17-6. LAWARx 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

17.3.3 Mode Registers 0–3 (MR[0–3]).

MR0 Mode Registers 0–3 Offset 0x100
MR1 Offset 0x180
MR2 Offset 0x200
MR3 Offset 0x280

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SAHTS		DAHE	SAHE	—	SRW	EOSIE	EOLNIE	EOLSIE	EIE	XFE	CDSM/ SWSM	CA	CTM	CC	CS
	R/W															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The mode register allows software to start a DMA transfer and to control various DMA transfer characteristics. **Table 17-7** describes the fields of the MR.

Table 17-7. MR Field Descriptions

Bits	Reset	Description	Setting
— 31–28	0	Reserved. Write to zero for future compatibility.	
BWC 27–24	0	**Bandwidth Control** If multiple channels are executing transfers concurrently, this value determines how many bytes a channel can transfer before the DMA controller shifts to the next channel. If a single channel is executing, this value determines how many bytes to transfer before pausing the channel; after pausing a new assertion of DREQ resumes channel operation.	0000 1 byte 0001 2 bytes 0010 4 bytes 0011 8 bytes 0100 16 bytes 0101 32 bytes 0110 64 bytes 0111 128 bytes 1000 256 bytes 1001 512 bytes 1010 1024 bytes 1011–1110 reserved. 1111 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 17-7. MR 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 17-7. MR 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** Halts 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.

17.3.4 Status Registers (SR*n*)

SR0 Status Registers 0–3 Offset 0x104
SR1 Offset 0x184
SR2 Offset 0x204
SR3 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 17-8** describes the fields of the SR.

Table 17-8. SR 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 17.2.3**, *DMA Errors,* on page 17-11 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 17-8. SR 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.

17.3.5 Current Link Descriptor Extended Address Registers (ECLNDAR*n*)

ECLNDAR0 Current Link Descriptor Extended Address Registers 0–3 Offset 0x108
ECLNDAR1 Offset 0x188
ECLNDAR2 Offset 0x208
ECLNDAR3 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 ECLNDAR 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 (CLNDAR) to point to the first link descriptor in memory. After the current descriptor is processed, the ECLNDAR and CLNDAR are loaded from the Next Link Descriptor Extended Address Register (ENLNDAR) and the Next Link Descriptor Address Register (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 Current List Descriptor Extended Address Register (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 17-9** describes the ECLNDAR fields.

Table 17-9. ECLNDAR Field Descriptions

Bits	Reset	Description	Setting
— 31–4	0	Reserved. Write to zero for future compatibility.	
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.	

17.3.6 Current Link Descriptor Address Registers (CLNDAR*n*)

CLNDAR0 Current Link Descriptor Address Registers 0–3 Offset 0x10C
CLNDAR1 Offset 0x18C
CLNDAR2 Offset 0x20C
CLNDAR3 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 CLNDAR 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 NLNDAR is examined. If EOLND is zero, the DMA controller reads in the new current link descriptor for processing. If EOLND is set, the last descriptor of the list was just completed. If extended chaining mode is not enabled, all DMA transfers are complete and the DMA controller halts. If extended chaining mode is enabled, the DMA controller examines the state of NLSDAR*n*[EOLSD] in the NLSDAR. 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 17-10** describes the CLNDAR fields.

Table 17-10. CLNDAR 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.	
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.	

17.3.7 Source Attributes Registers (SATR*n*)

SATR0 Source Attributes Registers 0–3 Offset 0x110
SATR1 Offset 0x190
SATR2 Offset 0x210
SATR3 Offset 0x290

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		SBPATMU	—	STFLOWLVL		SPCIORDER	SSME	STRANSINT				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 SATR contains the transaction attributes to be used for the DMA operation on the specified channel. Stride mode is enabled by setting SATR[SSME]. Source read transaction type is specified using SATR[SREADTTYPE].

ATMU bypass mode is enabled by setting SATR[SBPATMU] and is only applicable for RapidIO type accesses. ATMU bypass is only supported in small systems (up to 256 devices). If SATR[SBPATMU] is set, SATR[STRANSINT] must be set to RapidIO operation and attributes that would otherwise come from the ATMU must be specified in this register. If SATR[SBPATMU] is cleared, attributes are derived from local access windows and outbound ATMU registers according to the transaction address.

Table 17-11 describes the fields of the SATR.

Table 17-11. SATR Field Descriptions

Bits	Reset	Description	Setting
— 31–30	0	Reserved. Write to zero for future compatibility.	
SBPATMU 29	0	**Bypass ATMU for this DMA Operation** Indicates to use the ATMU outbound windows. **Note:** The value of this bit only applies to the external RapidIO interface.	0 Route the transfer through the ATMU outbound windows. SATR[SREADTTYPE] must specify a local address space transaction type. 1 Bypass ATMU. Never generate an address match. Always use the SATR values to route the transaction to the interface specified by the STRANSMIT field.
— 28	0	Reserved. Write to zero for future compatibility.	

Table 17-11. SATR Field Descriptions (Continued)

Bits	Reset	Description	Setting
STFLOWLVL 27–26	0	**RapidIO Transaction Flow Level** Selects a priority level for the transaction flow. **Note:** The value of this bit only applies to the external RapidIO interface in bypass mode (SBPATMU = 1).	00 Lowest priority transaction flow. 01 Medium priority transaction flow. 10 High priority transaction flow. 11 Reserved.
SPCIORDER 24	0	**Select PCI Ordering** When set, selects PCI ordering rules, which elevates write priority one level above reads. **Note:** This bit is ignored unless SBPATMU is set (1).	0 PCI ordering not used. 1 PCI ordering selected.
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.
STRANSINT 23–20	0	**DMA Source Transaction Interface** 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:** This bit is ignored unless SBPATMU is set (1) and the transaction is to the RapidIO interface.	1100 RapidIO interface. 1111 Local access memory. All other values are reserved.
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.	*RapidIO Interface in ATMU Bypass mode:* 0100 NREAD. 0111 Maintenance read. All other values reserved. *Local Address space (even in non-ATMU bypass mode):* 0100 Read, do not snoop local processor. 0101 Read, snoop local processor. 0111 Read, unlock L2 cache line. All other values reserved.
— 15–10	0	Reserved. Write to zero for future compatibility.	
ESAD 9–0	0	**Extended Source Address** The value depends on the mode: • Internal RapidIO transactions: Bits 9–4 are reserved and bits 3–0 represent the most significant 4 bits of the 36-bit source address. • RapidIO interface: Bits 9–2 represent the target ID and bits 1–0 represent the two most significant bits of the RapidIO address (33–32). **Note:** This field must be cleared (0) in non-bypass mode.	

17.3.8 Source Address Registers (SAR*n*)

SAR0 Source Address Registers 0–3 Offset 0x114
SAR1 Offset 0x194
SAR2 Offset 0x214
SAR3 Offset 0x294

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Local	SAD															
RapidIO Source	HOP_COUNT								CONFIG_OFFSET							
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	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															
RapidIO Source	CONFIG_OFFSET															
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 MR[CDSM/SWSM] and MR[SRW] are set, a write to this register simultaneously sets MR[CS], starting a DMA transfer. Software must ensure that this is a valid address.

If the RapidIO interface is the source of a transaction, the SARs are redefined. Several options exist for the transaction type that can be specified. There are a number of noncoherent reads and flush types for address-based read transactions, and message types for port-based read transactions. Maintenance packets use an offset instead of an address.

Table 17-12 describes the SAR fields.

Table 17-12. SAR Field Descriptions

Bits	Reset	Description
Local Source		
SAD 31–0	0	**Source Address** Contains the 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.
Source is RapidIO Interface		
HOP_COUNT 31–24	0	**Maintenance Packet Hop Count** This value is defined by the RapidIO Interconnect Specification 1.2.
CONFIG_ OFFSET 23–-0	0	**Maintenance Packet Word Offset** This value is defined by the RapidIO Interconnect Specification 1.2. Bits 1–0 are always zero because the value is 4-byte aligned.

17.3.9 Destination Attributes Registers (DATR*n*).

DATR0 Destination Attributes Registers 0–3 Offset 0x118
DATR1 Offset 0x198
DATR2 Offset 0x218
DATR3 Offset 0x298

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		DBPATMU	—	DTFLOWLVL		DPCIORDER	DSME	DTRANSINT				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 DATR[DSME] for the specified channel. Destination write transaction type is specified using the DATR[DWRITETTYPE] field.

ATMU bypass mode, which is applicable only for accesses to RapidIO interface, is enabled by setting DATR[DBPATMU]. If DATR[DBPATMU] is set, DATR*n*[DTRANSINT] must be set to RapidIO and attributes that would otherwise come from the ATMU must be specified in this register.

If DATR*n*[DBPATMU] is cleared, the target interface is derived from the local access ATMU mapping and the transaction is obtained from the value specified in DATR[DWRITETTYPE] using the local address space category.

Table 17-13 describes the fields of the DATR.

Table 17-13. DATR Field Descriptions

Bits	Reset	Description	Setting
— 31–30	0	Reserved. Write to zero for future compatibility.	
DBPATMU 29	0	**Bypass ATMU for this DMA Operation** Indicates to use the ATMU outbound windows. **Note:** The value of this bit only applies to the external RapidIO interface.	0 Route the transfer through the ATMU outbound windows. DATR[DWRITETTYPE] must specify a local address space transaction type. 1 Bypass ATMU. Never generate an address match. Always use the SATR values to route the transaction to the interface specified by the DTRANSMIT field.
— 28	0	Reserved. Write to zero for future compatibility.	

Table 17-13. DATR Field Descriptions (Continued)

Bits	Reset	Description	Setting
DTFLOWLVL 27–26	0	**RapidIO Transaction Flow Level** Selects a priority level for the transaction flow. **Note:** The value of this bit only applies to the external RapidIO interface in bypass mode (DBPATMU = 1).	00 Lowest priority transaction flow. 01 Medium priority transaction flow. 10 High priority transaction flow. 11 Reserved.
DPCIORDER 24	0	**Select PCI Ordering** When set, selects PCI ordering rules, which elevates write priority one level above reads. **Note:** This bit is ignored unless DBPATMU is set (1).	0 PCI ordering not used. 1 PCI ordering selected.
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.
DTRANSINT 23–20	0	**DMA Destination Transaction Interface** 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:** This bit is ignored unless DBPATMU is set (1) and the transaction is to the RapidIO interface.	1100 RapidIO interface. 1111 Local access memory. All other values are reserved.
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.	*RapidIO Interface in ATMU Bypass mode:* 0011 SWRITE for all but last transaction; NWRITE_R for last transaction. 0100 NWRITE for all but last transaction; NWRITE_R for last transaction. 0101 NWRITE_R. 0111 Maintenance write. All other values reserved. *Local Address space (even in non-ATMU bypass mode):* 0100 Write. All other values reserved.
— 15–10	0	Reserved. Write to zero for future compatibility.	
EDAD 9–0	0	**Extended Destination Address** The value depends on the mode: • Internal RapidIO transactions: Bits 9–4 are reserved and bits 3–0 represent the most significant 4 bits of the 36-bit destination address. • RapidIO interface: Bits 9–2 represent the target ID and bits 1–0 represent the two most significant bits of the RapidIO address (33–32). **Note:** This field must be cleared (0) in non-bypass mode.	

17.3.10 Destination Address Registers (DAR*n*)

DAR0 Destination Address Registers 0–3 Offset 0x11C
DAR1 Offset 0x19C
DAR2 Offset 0x21C
DAR3 Offset 0x29C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Local	DAD															
RapidIO Dest.	HOP_COUNT								CONFIG_OFFSET							
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	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															
RapidIO Dest.	CONFIG_OFFSET															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DAR contains the address to which the DMA controller writes data for the specified channel. In direct mode, if MR[CDSM/SWSM] is cleared and MR[SRW] is set, a write to this register simultaneously sets MR[CS], starting a DMA transfer. Software must ensure that this is a valid address.

If the RapidIO interface is the destination of a transaction, the DARs are redefined. Several options exist for the transaction type that can be specified. There are a number of noncoherent write and flush types for address-based write transactions, and message types for port-based write transactions. Maintenance packets use an offset instead of an address.

Table 17-14 describes the DAR fields.

Table 17-14. DAR Field Descriptions

Bits	Reset	Description
Local Source		
DAD 31–0	0	**Destination Address** Contains the 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.
Source is RapidIO Interface		
HOP_COUNT 31–24	0	**Maintenance Packet Hop Count** This value is defined by the RapidIO Interconnect Specification 1.2.
CONFIG_ OFFSET 23–-0	0	**Maintenance Packet Word Offset** This value is defined by the RapidIO Interconnect Specification 1.2. Bits 1–0 are always zero because the value is 4-byte aligned.

17.3.11 Byte Count Registers (BCR*n*).

BCR0 Byte Count Registers 0–3 Offset 0x120
BCR1 Offset 0x1A0
BCR2 Offset 0x220
BCR3 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 17-15** describes the fields of the BCR.

Table 17-15. BCR 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.

17.3.12 Extended Next Link Descriptor Address Registers (ENLNDAR*n*)

ENLNDAR0 Extended Next Link Descriptor Address Registers 0–3 Offset 0x124
ENLNDAR1 Offset 0x1A4
ENLNDAR2 Offset 0x224
ENLNDAR3 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 ENLNDAR 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 17-16** describes the ENLNDAR fields.

Table 17-16. ENLNDAR 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.	

17.3.13 Next Link Descriptor Address Registers (NLNDAR*n*)

NLNDAR0 Next Link Descriptor Address Registers 0–3 Offset 0x128
NLNDAR1 Offset 0x1A8
NLNDAR2 Offset 0x228
NLNDAR3 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 NLNDAR 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 NLNDAR and the NLNDAR*n*[EOLND] field in the NLNDAR is examined. If EOLND is zero, the DMA controller reads in the new current link descriptor for processing. If EOLND is set, the last descriptor of the list was just completed. If extended chaining mode is not enabled, all DMA transfers are complete and the DMA controller halts. If extended chaining mode is enabled, the DMA controller examines the state of NLSDAR*n*[EOLSD] in the NLSDAR. 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 17-17** describes the NLNDAR fields.

Table 17-17. NLNDAR 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 17-17. NLNDAR 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.

17.3.14 Extended Current List Descriptor Address Registers (ECLSDAR*n*)

ECLSDAR0 Extended Current List Descriptor Address Registers 0–3 Offset 0x130
ECLSDAR1 Offset 0x1B0
ECLSDAR2 Offset 0x230
ECLSDAR3 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 ECLSDAR 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 CLSDAR 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 ENLNDAR and the NLNDAR. Then the controller evaluates the NLSDAR*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 17-18** describes the ECLSDAR fields.

Table 17-18. ECLSDAR 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.	

17.3.15 Current List Descriptor Address Registers (CLSDAR*n*)

CLSDAR0 Current List Descriptor Address Registers 0–3 Offset 0x134
CLSDAR1 Offset 0x1B4
CLSDAR2 Offset 0x234
CLSDAR3 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 CLSDAR 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 ENLSDAR and the NLSDAR. Then the controller evaluates the NLSDAR*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 17-19** describes the CLSDAR fields.

Table 17-19. CLSDAR 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.	

17.3.16 Extended Next List Descriptor Address Registers (ENLSDAR*n*)

ENLSDAR0 Extended Next List Descriptor Address Registers 0–3 Offset 0x138
ENLSDAR1 Offset 0x1B8
ENLSDAR2 Offset 0x238
ENLSDAR3 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 ENLSDAR 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 17-20 describes the ENLSDAR fields.

Table 17-20. ENLSDAR 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.	

17.3.17 Next List Descriptor Address Registers (NLSDAR*n*)

NLSDAR0 Next List Descriptor Address Registers 0–3 Offset 0x13C
NLSDAR1 Offset 0x1BC
NLSDAR2 Offset 0x23C
NLSDAR3 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 NLSDAR 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 17-21** describes the NLSDAR fields.

Table 17-21. NLSDAR 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.

17.3.18 Source Stride Registers (SSR*n*)

SSR0 Source Stride Registers 0–3 Offset 0x140
SSR1 Offset 0x1C0
SSR2 Offset 0x240
SSR3 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 SSR 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 SSR 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 17-21** describes the SSR fields.

Table 17-22. SSR 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.

17.3.19 Destination Stride Registers (DSR*n*)

DSR0 Destination Stride Registers 0–3 Offset 0x144
DSR1 Offset 0x1C4
DSR2 Offset 0x244
DSR3 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 DSR 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 DSR 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 17-23** describes the DSR fields.

Table 17-23. DSR 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.

17.3.20 DMA General Status Register (DGSR))

DGSR 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 17-24** describes the fields of the DGSR.

Table 17-24. DGSR 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 17-24. DGSR 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.

QUICC Engine™ Subsystem 18

The QUICC Engine™ subsystem is a versatile RISC-based communication processor that supports multiple external interfaces and protocols independently from the core processor(s) in an integrated processing device. This allows the cores to execute the data processing code and be relieved from the data transfer and handling overhead. This chapter provides an overview of the QUICC Engine subsystem components used in the MSC8144E, 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
- Three programmable Unified Communication Controllers (UCCs), two of which provides dedicated support for an Ethernet controller for MII/RMII/SMII/RGMII/SGMII interfaces and one of which provides dedicated support for an ATM controller for UTOPIA and POS interfaces.
- Serial peripheral interface (SPI)

Note: Detailed information about QUICC Engine functionality and programming is provided in the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM). Refer to that manual for all the functional and programming details

Note: If the QUICC Engine subsystem is not used for booting, the user must clear the QUICC Engine DRAM before using it to ensure correct operation.

18.1 Overview

Figure 18-1 shows an architectural diagram of the QUICC Engine subsystem. The UCCs group is controlled by a RISC engine. A common multi-user RAM is used to store parameters for RISC engines. Each RISC has a ROM associated with it that contains the code image. The instruction RAM is used to run RISC code from the RAM. The internal clocks (RCLK/TCLK) for each UCC can be programmed to use either an external or internal source. The rate of these clocks can be up to one-half of the QUICC Engine subsystem clock frequency. However, the ability of an interface to support a sustained bit stream depends on the protocol settings and other factors.

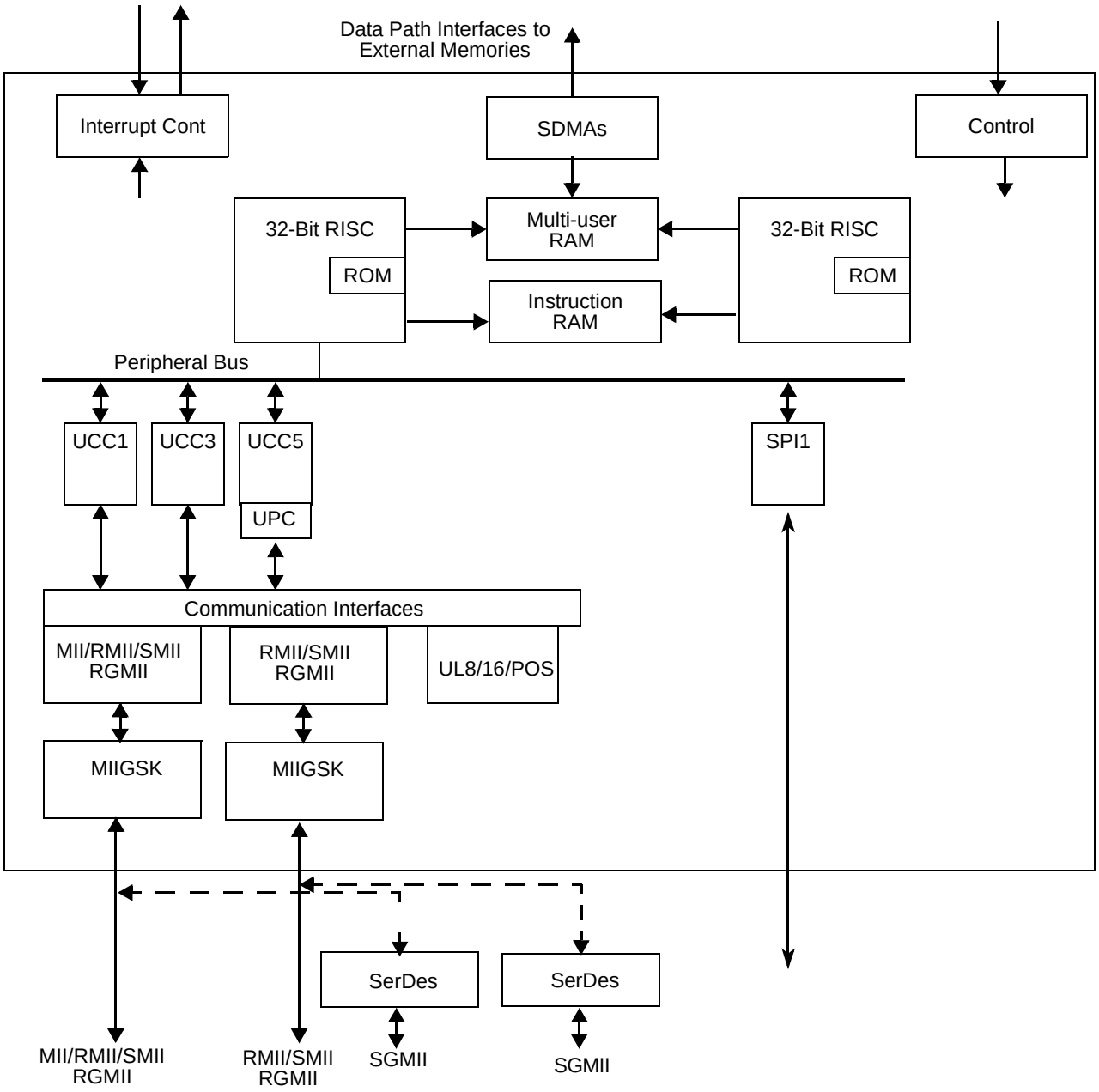


Figure 18-1. QUICC Engine Subsystem Architectural Block Diagram

18.2 RISC Processors

The two 32-bit RISC processors reside on a bus separate from the SC3400 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 SC3400 DSP cores issues commands to the RISC processors by writing to the QUICC Engine Command Register (CECR), which is described in **Section 18.10**. 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 (CLASS128 default, programmable after initialization) per instruction
- 32-bit instruction object code
- Code execution from internal ROM or RAM
- 32-bit ALU data path
- 64-bit multi-port RAM access
- Optimized for communications processing
- DMA bursting of serial data to/from external memory

18.2.1 SC3400 Core Interface

The RISC processors communicate with the SC3400 cores in several ways:

- Many parameters are exchanged through the multi-port RAM.
- The RISC processors can execute special commands issued by the SC3400 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 SC3400 cores can read the QUICC Engine subsystem status/event registers at any time.

18.2.2 Peripheral Interface

The RISC processors use the peripheral bus to communicate with all of its UCCs. Each controller has separate receive and transmit 192-byte FIFOs.

18.2.3 Parameter RAM

The QUICC Engine subsystem maintains a section in the multi-user RAM that contains the associated parameter values for the operation of the UCCs and SPI. Each peripheral has an associated page in the parameter RAM. The exact definition of the parameter RAM for each peripheral is in the protocol descriptions in the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM). The size of the parameter RAM page differs from protocol to protocol. The minimum size of the parameter RAM page assigned to any protocol is 64 bytes. The base address of the parameter RAM page must be aligned to 64 bytes.

Some parameter RAM values must be initialized before the UCC is enabled. The QUICC Engine subsystem initializes or writes other values. Once initialized, most parameter RAM values need not be accessed by user software, because most activity centers around the TxBDs and RxBDs rather than the parameter RAM. However, if the parameter RAM is accessed, note the following:

- You can read the parameter RAM at any time.
- You can write to the Tx parameter RAM only when the transmitter is disabled (that is, after a STOP TRANSMIT command or after the buffer/frame finishes transmitting after a GRACEFUL STOP TRANSMIT command and before a RESTART TRANSMIT command).
- You can write to the Rx parameter RAM only when the receiver is disabled. The CLOSE RX BD command does not stop reception, but it does allow the user to extract data from a partially full Rx buffer.

After reset, the QUICC Engine subsystem initializes the base addresses of the parameter RAM pages for all the UCCs to default values. You can override the default values by issuing the ASSIGN PAGE command to the QUICC Engine subsystem. The advantage of using the ASSIGN PAGE command is that you can arrange the parameter RAM area efficiently (with no unused areas) according to the specific peripherals/protocols used in your system.

Table 18-1 lists the default values of the parameter RAM pages base addresses assigned by the QUICC Engine subsystem to the different peripherals.

Table 18-1. Default Parameter RAM Base Addresses

Address Offset	Peripheral	Size (Bytes)
0x8400	UCC 1 (Rx and Tx)	256
0x8600	UCC 3 (Rx and Tx)	256
0x8000	UCC 5 (Rx and Tx)	256
0x8900	SPI (Rx and Tx)	128

18.2.4 Buffer Descriptors (BDs)

If you are programming the UCCs, you need to know how the controllers use buffer descriptors to define buffer allocation. A buffer descriptor (BD) contains the essential information about each buffer in memory. Each buffer is referenced by a BD that can reside anywhere in system memory. These BDs are split between all UCCs.

Each 64-bit BD has the structure shown in **Figure 18-2**. This structure is common to all UCCs. A receive buffer descriptor (RxBD) table and a transmit buffer descriptor (TxBD) table are associated with each controller. Each table can have multiple BDs.

Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	Status and Control															
0x2	Data Length															
0x4	High-Order Buffer Pointer															
0x6	Low-Order Buffer Pointer															

Figure 18-2. Buffer Descriptor Structure

In this discussion, the BD and field values use the following convention:

```
BD.field
```

Table 18-2 shows the possible BD and field naming conventions. Bit names in `RxBD.bd_cstat` and `TxBD.bd_cstat` use the following convention:

```
BD.bd_cstat.bit
```

Table 18-2. Buffer Descriptor Name Convention

BD	Field	Example
RxBD/TxBD	bd_cstat	`TxBD.bd_cstat`.R refers to the ready bit in the TxBD's status and control field.
	bd_length	`RxBD.bd_length` refers to RxBD data length field.
	bd_addr	`RxBD.bd_addr` refers to RxBD buffer pointer field.

The structural elements of a buffer descriptor are defined as follows:

- *Status and control*. The 16-bit value at `offset+0x0`, which contains status and control bits that control and report status information on the data transfer. The RISC processor updates the status bits after the buffer is sent or received. Only this field differs for each protocol. Refer to the relevant chapter in this manual for each protocol `RxBD.bd_cstat` and `TxBD.bd_cstat` bit description.
- *Data length*. The 16-bit value at `offset+0x2`, which contains the number of bytes sent or received.

- *RxBD data length*. The number of bytes the RISC processor writes into the RxBD buffer once the BD closes. The RISC processor updates this field after the received data is placed into the buffer and the buffer is closed. You do not need to initialize this field. In frame-based protocols, `RxBD.bd_length` contains the total frame length including CRC bytes. If a received frame length, including CRC, is an exact multiple of the parameter RAM maximum receive buffer length MRBLR, the last buffer holds no actual data but the associated BD contains the total frame length.
- *TxBD data length*. The number of data bytes the controller needs to transmit from its buffer. The RISC processor never modifies this field. This field needs to be initialized by the user.
- *Buffer pointer*. The 32-bit data at `offset+0x4`, which points to the beginning of the buffer in internal or external memory.
- *RxBD buffer pointer*. The buffer pointer value must be a multiple of four to be word-aligned.
- *TxBD buffer pointer*. The buffer pointer value can be even or odd.

18.2.5 Multithreading

The Ethernet Controllers are able to processes frames or cells at high bit rates (gigabit Ethernet and above OC-3 nominal rates). In order to achieve these bit rates the UCC receiver and the UCC transmitter are able to process multiple frames/cells simultaneously at any given time. This is implemented with the multithreading mechanism. Each thread processes a different frame/cell. The multithreading processing mechanism comprises three components: Distributor, Threads and in some cases Terminator. **Figure 18-3** shows a high-level diagram of the multithreading architecture.

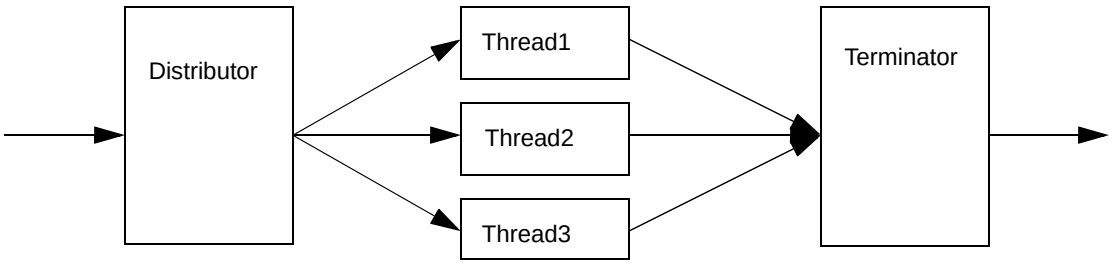


Figure 18-3. Multi-Threading Processing Mechanism

Each one of the components (distributor, threads and terminator) has an ID number associated with it, referred to as its Serial Number (SNUM). The distributor SNUM is always the SNUM of the UCC receiving or transmitting the data. Each one of the transmitter and receiver threads has its own parameter RAM located in the multi-port RAM. The user software initializes the values for the SNUM and the pointer of the parameter RAM base address at initialization time.

Note: See the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM).for specific interface configuration details.

18.2.6 Serial Numbers (SNUMs)

Each peripheral has a unique serial number (SNUM) associated with it. Threads, which are used by the multithreading mechanisms described in **Section 18.2.5**, *Multithreading* also have a unique serial number. There are two cases for which you should specify the SNUM:

1. When issuing the ASSIGN PAGE command.
2. When initializing the multithreading mechanism.

Table 18-3 lists the Serial Numbers (SNUMs) used to program the multithreading options in the UCCs for the Ethernet controllers. The component column indicates the peripheral or thread name for each SNUM.

Table 18-3. SNUM Table

SNUM	Component	SNUM	Component	SNUM	Component	SNUM	Component
0x00	UCC1 TX	0x40	—	0x80		0xC0	Reserved
0x01	UCC1 RX	0x41	—	0x81		0xC1	Reserved
0x04	Thread16	0x44	—	0x84	—	0xC4	—
0x05	Thread17	0x45	—	0x85	—	0xC5	—
0x08	—	0x48		0x88	Thread0	0xC8	Thread8
0x09	—	0x49		0x89	Thread1	0xC9	Thread9
0x0C	Thread18	0x4C	—	0x8C	—	0xCC	—
0x0D	Thread19	0x4D	—	0x8D	—	0xCD	—
0x10	Reserved	0x50	Reserved	0x90	Reserved	0xD0	—
0x11	Reserved	0x51	Reserved	0x91	Reserved	0xD1	—
0x14	Thread20	0x54	—	0x94	—	0xD4	—
0x15	Thread21	0x55	—	0x95	—	0xD5	—
0x18	—	0x58	—	0x98	Thread2	0xD8	Thread10
0x19	—	0x59	—	0x99	Thread3	0xD9	Thread11
0x1C	Thread22	0x5C	—	0x9C	—	0xDC	—
0x1D	Thread23	0x5D	—	0x9D	—	0xDD	—
0x20	UCC3 TX	0x60	Reserved	0xA0	Reserved	0xE0	
0x21	UCC3 RX	0x61	Reserved	0xA1	Reserved	0xE1	
0x24	Thread24	0x64	—	0xA4	—	0xE4	—
0x25	Thread25	0x65	—	0xA5	—	0xE5	—
0x28	—	0x68	—	0xA8	Thread4	0xE8	Thread12
0x29	—	0x69	—	0xA9	Thread5	0xE9	Thread13
0x2C	Thread26	0x6C	—	0xAC	—	0xEC	—
0x2D	Thread27	0x6d	—	0xAD	—	0xED	—
0x30	Reserved	0x70	Reserved	0xB0	Reserved	0xF0	TIMER
0x31	Reserved	0x71	Reserved	0xB1	Reserved	0xF1	Lowest
0x34	Thread28	0x74	—	0xB4	—	0xF4	—
0x35	Thread29	0x75	—	0xB5	—	0xF5	—
0x38	—	0x78	—	0xB8	Thread6	0xF8	Reserved
0x39	—	0x79	—	0xB9	Thread7	0xF9	Reserved
0x3C	Reserved	0x7C	—	0xBC	—	0xFC	—
0x3D	Reserved	0x7D	—	0xBD	—	0xFD	—

Note: See the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM) for details on how to use the SNUM with the ASSIGN PAGE command.

18.2.7 IRAM

The QUICC Engine subsystem includes 48 KB of IRAM that can be used to store processing code for the RISC processors. The IRAM can be split into two 24-KB areas assigned individually to each of the two RISC processors, or it can be shared by both processors as one contiguous 48-KB memory space. Access is through the IRAM address register (IADDR) using the IRAM data register (IDR). See the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM) for details.

18.3 Serial DMA Controller

The QUICC Engine subsystem has one physical serial DMA (SDMA) channel that interfaces with the internal MBus. The QUICC Engine subsystem implements two dedicated virtual SDMA channels for each peripheral, one for the receiver and one for the transmitter. Additional virtual channels are available for general purpose accesses.

18.3.1 Data Paths

Figure 18-4 is a simplified block diagram that shows the data paths in the MSC8144E. The MSC8144E may be implemented with one DDR bus (32 or 64 bit data).

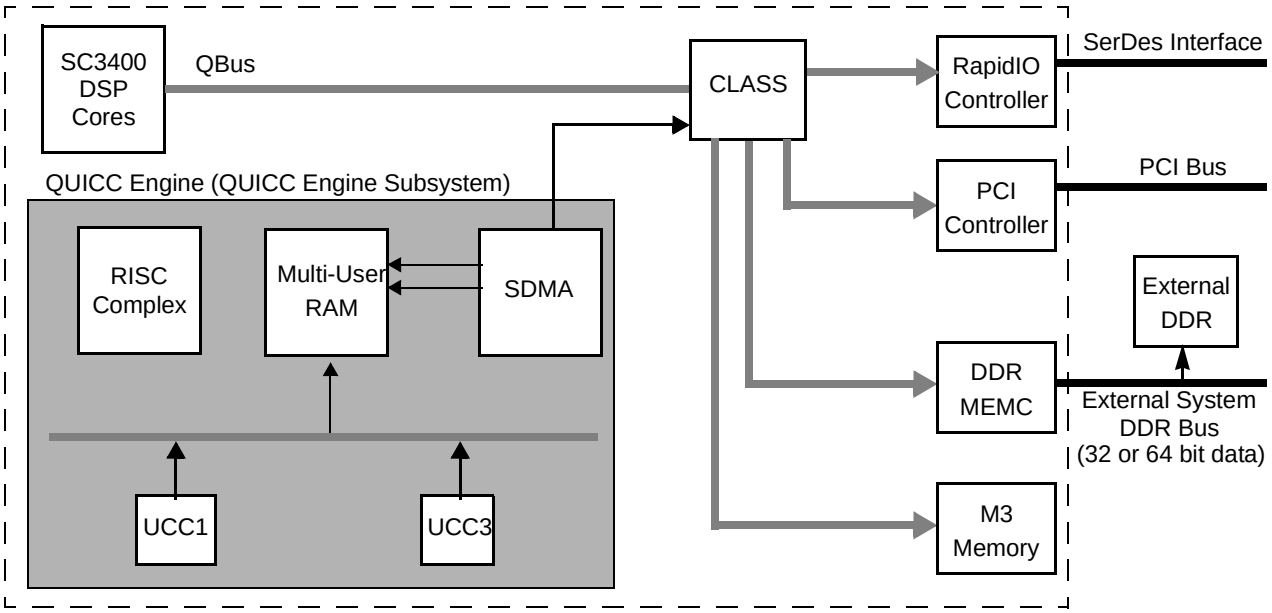


Figure 18-4. MSC8144E Data Paths

The CLASS is responsible for distribution of MBus transactions to the various possible targets (for example, RapidIO or DDR memory controller) based on the transaction address. See **Chapter 4**, *Chip-Level Arbitration and Switching System (CLASS)* for details.

18.3.2 SDMA and Bus Error

If a bus error occurs on an SDMA access from the QUICC Engine subsystem, the following occurs:

1. The QUICC Engine subsystem generates a unique maskable interrupt in the SDMA status register (SDSR),
2. The DSP core uses an interrupt service routine (ISR) to read the SDSR to determine which bus generated the error.
3. System recovery depends on how you configure the SDMR[SBER_1] bit. One of the following occurs:
   - The QUICC Engine subsystem disables the peripheral or thread associated with the bus error and continues to operate as usual on all other peripherals (default). The recovery sequence in this mode is based on re-initialization of the peripheral associated with the error, or
   - The QUICC Engine subsystem stops all activity, and must be reset through the reset command to the QUICC Engine Command Register (CECR).

In general, it should be noted that the DSP core, depending on how it is programmed, may read the SDMA address register (SDTA) to determine the address at which the bus error occurred, and the SDMA SNUM register (SDTM) to determine which peripheral or thread was being serviced by the SDMA virtual channel. See **Table 18-3** for the list of SNUMs.

The SDTA and the SDTM registers store information related to accesses to the MBus. These two registers are not updated with the address and SNUM of subsequent transactions as long as the event bit in the SDSR is set.

18.3.2.1 Simple Recovery from Bus Error

The simplest recovery from a bus error is a QUICC Engine subsystem reset followed by an overall QUICC Engine subsystem re-initialization procedure, regardless of the peripheral that has actually caused the error. The reasoning is that for some applications, stopping one peripheral leads to a chain reaction that ultimately disrupts the correct interworking operation of the QUICC Engine subsystem. For debug purposes, it is valuable to observe continued operation, but a selective recovery does not provide any added value. This non-distinctive procedure also reduces the complexity of the recovery flow.

18.3.2.2 Selective Peripheral Recovery Procedure

Selective recovery can be a complex procedure due to the following:

- The Ethernet controllers are multi-thread controllers and SNUM to peripheral/controller translation requires maintaining association tables.
- Status for multiple bus errors is not maintained, so in theory there might be additional non-reported bus errors and the recovery will not be complete.

For these reason, a full reset and re-initialization are recommended.

18.3.3 SDMA and Reset

When the QUICC Engine subsystem is reset through the CECR[RST] bit (see the QUICC Engine Block Reference Manual with Protocol Interworking (QEIWRM) for details), the SDMA continues to process outstanding transactions in its FIFOs although the data related to these transactions may be corrupted. During system reset ($\overline{\text{SRESET}}$ or $\overline{\text{HRESET}}$), all SDMA FIFOs are flushed and all outstanding transactions are stopped.

18.3.4 MBus Access

The SDMA requests the bus from the CLASS at two possible priority levels. When the SDMA is in normal state, it requests the bus at priority level programmed by the user in the SDMR[EBPR] bit field. When the SDMA is in emergency state, it requests the bus at the highest priority level that the CLASS supports; the QUICC Engine subsystem is assigned an arbitration weight through a field in GCR11 (see **Section 8.2.27**, *General Control Register 11 (GCR11)*, on page 8-42 for details). SDTR and SDHY program the threshold and hysteresis values that affect the conditions for SDMA normal and emergency states. It is possible to mask the emergency state priority requests globally in SDMR[EBMSK] and enforce the priority set in SDMR[EBPR] regardless of the SDMA state.

The SDMA asserts the highest priority request (emergency state) for any one of the following reasons:

- When one of the FIFOs in the QUICC Engine subsystem reaches an emergency state (too full on Rx or too empty in the middle of a frame during Tx)
- When the internal SDMA data buffers are filled beyond a certain level (programmed in SDTR/SDHY registers).
- When the internal SDMA command queue is filled beyond a certain level (programmed in SDTR/SDHY registers).

A priority request to the multi-user RAM is also asserted by the SDMA, if needed, for the same reasons. It is possible to mask the high priority request globally in SDMR[ERMSK].

18.3.5 SDMA Internal Resource

The SDMA requires temporary buffering for some data in the multi-user RAM. The base address for the SDMA temporary buffer is programmed in the SDEBCR. The base address must be aligned to a 4-KB boundary. The size of the multi-user RAM needed for this buffering is programmable via the STBSZ bit field in the SDMR. The size the temporary buffer that must be allocated ranges from a minimum of 512 bytes to maximum of 3 KB in the multi-user RAM. See the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM) for details.

18.4 Clocking

The QUICC Engine subsystem uses a programmable multiplexing system to route the various clocks used to transfer data through the external interfaces. Depending on the application and interface used, these signals can be supplied externally or taken from the internal programmable baud-rate generators. The following subsections describe the multiplexing unit and the internal baud-rate generators.

18.4.1 Multiplexer Logic

The QUICC Engine subsystem multiplexing logic routes clocks and connects the physical interfaces to the QUICC Engine subsystem peripherals, including the two UCCs. The multiplexer logic routes clocks to all the QUICC Engine subsystem peripherals from a bank of internal clocks (BRG[5–8]) and a bank of external clocks.

Physical signal connections are also configured using the clock route registers. However, because these signal lines are multiplexed with other functions at the I/O lines, you must make sure that

the lines are also enabled through the GPIO registers and initial mode selection pins. **Figure 18-5** shows a block diagram of the QUICC Engine subsystem multiplexing logic.

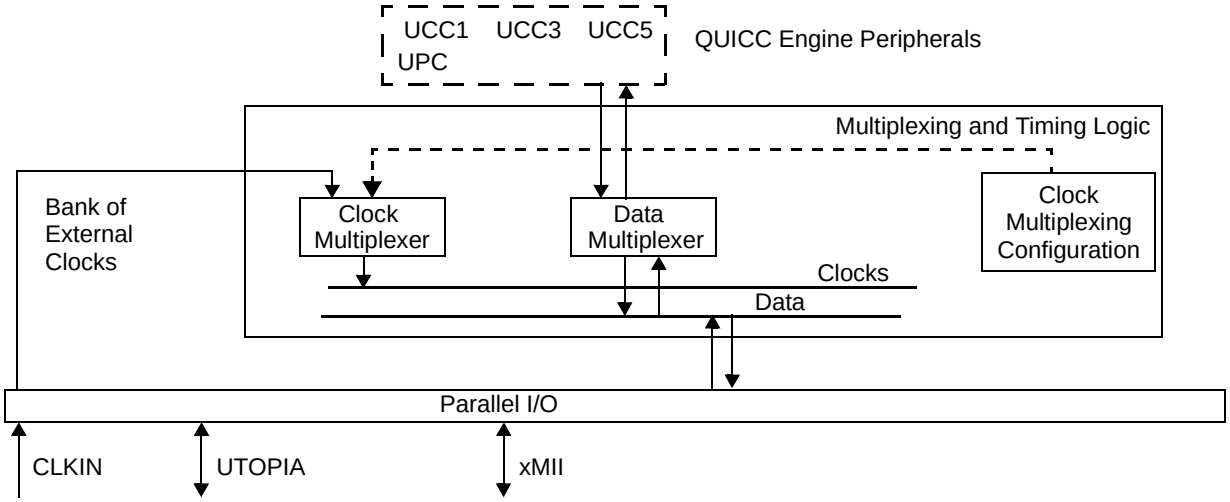


Figure 18-5. QUICC Engine Multiplexing Logic Block Diagram

The multiplexing logic is primarily used for clock assignment for:

- UCC1, UCC3, and UCC5.
- UPC
- UPC internal clock rate.
- Serial management interface (SMI) initiator.

The clocks are derived from a bank of internal BRGs and external clock inputs as shown in **Figure 18-6**. The clock routing options are described in **Table 18-4** and **Table 18-5**. The bank of clocks selection logic applies an available clock to a peripheral that requires a clock. Because the peripheral is not directly connected to a specific clock source, peripherals can share the same clock. There are two main advantages to the bank-of-clocks approach. First, a peripheral is not forced to choose a serial device clock from a predefined input or BRG. Second, peripheral receivers and transmitters that need the same clock rate can share the same source. This

configuration leaves additional signal lines for other functions and minimizes potential skew between multiple clock sources.

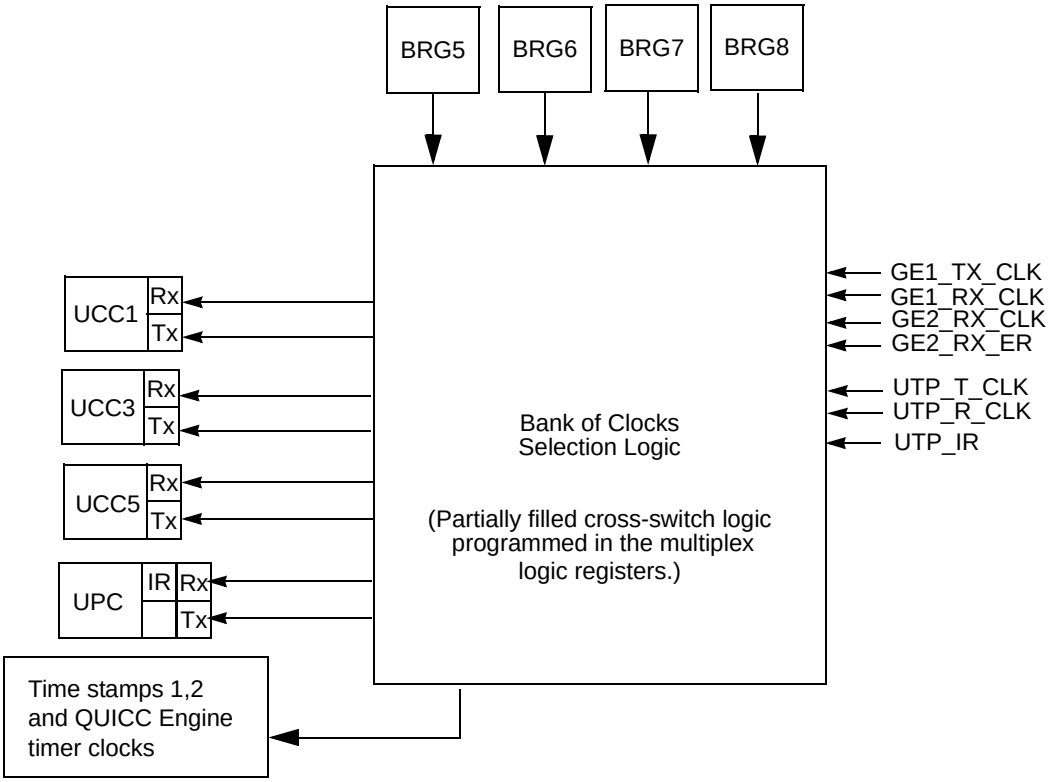


Figure 18-6. Bank of Clocks

Table 18-4. Clock Source Options Using External Clock Signals

Clock	External CLK						
	GE1_RX_CLK	GE1_TX_CLK	GE2_RX_ER	GE2_RX_CLK	UTP_RCLK	UTP_TCLK	UTP_IR
UCC1 Rx	V						
UCC1 Tx		V					
UCC3 Rx			V				
UCC3 Tx				V			
UPC Rx					V		
UPC Tx						V	
UPC internal rate							V
Time Stamp 1			V	V			
Time Stamp 2			V	V			

Table 18-5. Clock Source Options - Internal Clock Generators

Clock	BRG Number			
	5	6	7	8
UCC1 Rx			V	
UCC1 Tx			V	
UCC3 Rx				V
UCC3 Tx				V
UPC Rx		V		
UPC Tx	V			
UPC internal rate is the UPC1 Tx clock	V			

18.4.2 Baud-Rate Generators (BRGs)

The QUICC Engine subsystem contains four independent, identical baud-rate generators (BRGs) that can be used with the UCCs. The clocks produced by the BRGs are sent to the bank-of-clocks selection logic, where they can be routed to the controllers. Each BRG can be routed to one or more UCCs. **Figure 18-7** shows the block diagram for a BRG.

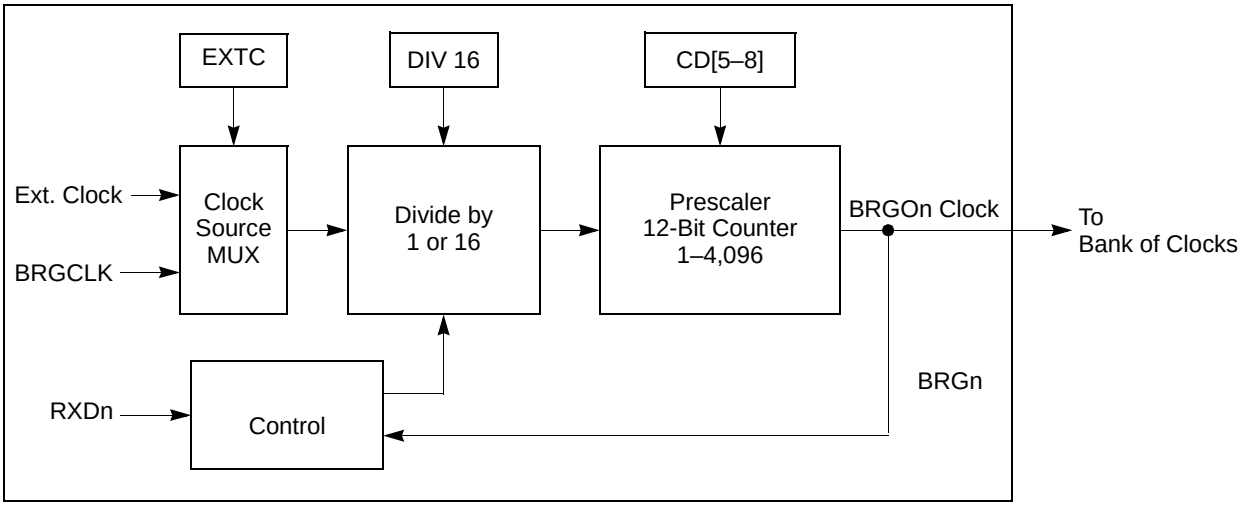


Figure 18-7. Baud-Rate Generator (BRG) Block Diagram

All BRGs can use BRGCLK as its source clock, or the external clock input selected by the value of BRGC*x*[EXTC]. The BRGCLK is an internal signal generated in the clock synthesizer. The external source option allows flexible baud-rate frequency generation, independent of the system frequency. Additionally, the external source option allows a single external frequency to be the source for more than one BRG. The external source signals are not synchronized internally before being used by the BRG. The BRG provides a divide-by-16 option (BRGC*x*[DIV16]) and a 12-bit prescaler (BRGC*x*[CD]) to divide the source clock frequency. The combined source-clock divide factor can be changed on-the-fly, except when changing to or from a CD value of 1, 2, or 3. For these values, disable the BRG and reset it before you program the new value. In addition, you should not make two changes within two source clock periods. If the BRG divides the clock by an even value, the transitions of BRGO*n* always occur on the rising edge of the source clock. If the divide factor is odd, the transitions alternate between the falling and rising edges of the source clock. The output of the BRG can be sent to the autobaud control block.

18.5 Interrupt Controller

The QUICC Engine subsystem interrupt controller sends general interrupts to the DSP cores in the MSC8144E 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 MSC8144E uses five general configuration registers to expedite some interrupt responses:

18.5.1 QUICC Engine Subsystem List of Interrupts to Core

Table 18-6 lists the categories and QUICC Engine subsystem interrupts that are sent to the DSP core.

Table 18-6. QUICC Engine Subsystem Core Interrupts

Category	Interrupt
Global Interrupts	QUICC Engine subsystem DRAM ECC ERROR
	QUICC Engine subsystem IMEM ECC ERROR
	QUICC Engine subsystem Interrupt High
	QUICC Engine subsystem Interrupt Low
	UCC1 SMII Interrupt
	UCC3 SMII Interrupt

Table 18-6. QUICC Engine Subsystem Core Interrupts (Continued)

Category	Interrupt
UCC1 Interrupts	UCC1 All
	UCC1 RX0
	UCC1 RX1
	UCC1 RX2
	UCC1 RX3
	UCC1 RX4
	UCC1 RX5
	UCC1 RX6
	UCC1 RX7
	UCC1 TX0
	UCC1 TX1
	UCC1 TX2
	UCC1 TX3
	UCC1 TX4
	UCC1 TX5
	UCC1 TX6
	UCC1 TX7
UCC3 Interrupts	UCC3 All
	UCC3 RX0
	UCC3 RX1
	UCC3 RX2
	UCC3 RX3
	UCC3 RX4
	UCC3 RX5
	UCC3 RX6
	UCC3 RX7
	UCC3 TX0
	UCC3 TX1
	UCC3 TX2
	UCC3 TX3
	UCC3 TX4
	UCC3 TX5
	UCC3 TX6
	UCC3 TX7
UCC5 (ATM) Interrupts	Global buffer pool busy
	Global interrupt 0
	Global interrupt 1
	Global interrupt 2
	Global interrupt 3
	Global red-line
	Interrupt queue 0 overflow
	Interrupt queue 1 overflow
	Interrupt queue 2 overflow
	Interrupt queue 3 overflow
	Transmit internal rate underrun

18.5.2 Interrupt Configuration

Figure 18-8 shows the QUICC Engine interrupt structure. The interrupt controller receives interrupts from various internal sources within the QUICC Engine subsystem.

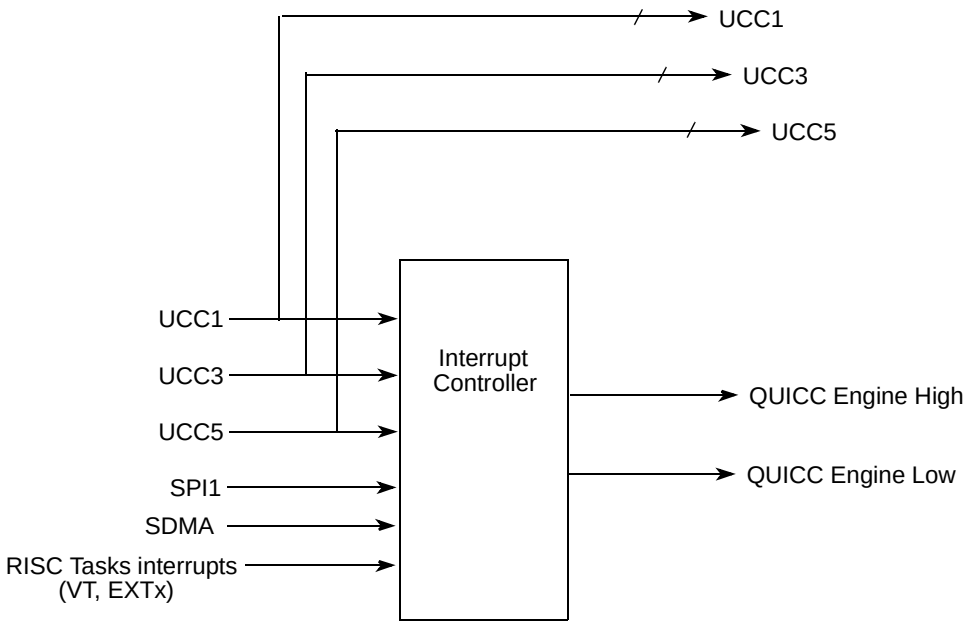


Figure 18-8. QUICC Engine Interrupt Structure

The interrupt controller allows masking of each interrupt source. In addition, multiple events within a QUICC Engine subsystem peripheral event are also maskable. The interrupt controller can be programmed to split the interrupts between two interrupt outputs: QUICC Engine Subsystem High and QUICC Engine Subsystem Low. In addition to the QUICC Engine Subsystem High and Low line, each UCC has dedicated interrupts for some of its event. The Ethernet controllers (UCC1 or UCC3) have 17 interrupts each. One interrupt is an OR of all events. Each Ethernet controller has also eight RX and eight TX queue interrupts. UCC3 Has only thirteen ATM dedicated interrupts. For more details about these dedicated interrupts see UCCE and UCCM of each protocol.

Note: See the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM) for details about configuring the QUICC Engine interrupt system.

18.6 UCCs

The QUICC Engine subsystem UCCs implement the Ethernet protocols. This section provides a general overview of the UCCs. For a detailed description of the feature set and the protocol-specific programming model, refer to the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM). The key features of the UCCs include:

- Supports Ethernet and ATM protocols:
  - Full 10/100 Mbps, full-duplex/half-duplex Ethernet/IEEE 802.3x/VLAN through MII/RMII/SMII.
  - Full 1000 Mbps, full-duplex Ethernet/IEEE 802.3x/VLAN through RGMII/SGMII.
  - Full-duplex ATM AAL5 segmentation and reassembly (SAR) through UTOPIA L2.
- UCC clock can be derived from a internal clock or an external signal.
- Uses bursts to improve bus usage.
- Multibuffer data structure for received and transmitted data.
- Buffers and buffer descriptors (BDs) may reside anywhere in system memory.
- Programmable size-virtual FIFO buffers.
- Echo and local loopback modes for testing.

The UCC block diagram is shown in **Figure 18-9**.

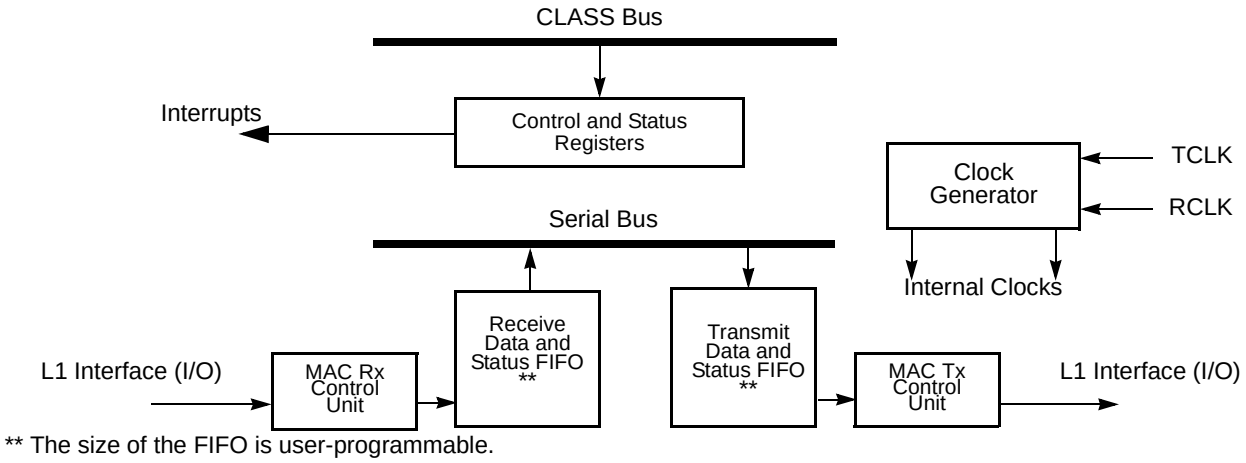


Figure 18-9. UCC Block Diagram

High-speed protocols require large FIFO depth. Sufficient FIFO size is important for increasing the QUICC Engine subsystem performance by eliminating overrun and underrun bottlenecks. Each protocol has an optimized FIFO size that depends not only on the bit rate, but also on other parameters such as packet or frame size. The UCC, when configured for high-speed protocols, extends the hardware FIFO into the QUICC Engine subsystem internal RAM. This extension is called virtual FIFO or VFIFO, because its size and location within the RAM are programmable according to the specific protocol. The FIFO as shown in **Figure 18-9** is constructed using a real hardware FIFO embedded in the UCC and its extension to a virtual FIFOs in the QUICC Engine

subsystem RAM. This flexible FIFO structure enables the QUICC Engine subsystem to sustain wire speed frame or cell bursts by allocating larger FIFO memory when required. The size of the virtual FIFO is user-programmable. The actual size that is needed depends on a number of factors, especially the following:

- Maximum packet size
- Protocols running on the UCC
- Memory bus latency

18.7 Ethernet Controllers

The Ethernet interface is a widely-used local area network (LAN) that is based on the carrier-sense, multiple access, collision detect (CSMA/CD) approach. The **IEEE** 802.3 standard was developed to codify the requirements of such a system to insure interoperability between devices operating on the LAN. Because Ethernet and the **IEEE** protocols are similar and can coexist on the same LAN, this manual uses the generic term Ethernet unless otherwise noted. The MSC8144E 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:

- 10/100 Mbps MII interface (Ethernet 1 only).
- 1000 Mbps RGMII interface
- 10/100 Mbps RMII interface
- 10/100 Mbps SMII interface
- 1000 Mbps SGMII interface.

The media-independent interface (MII) was developed first and used four data lines to transmit information by nibbles (half-bytes); this is the interface defined in the **IEEE** 802.3 standard. To reduce the number of interconnect signals, the RMII, or reduced mode was developed; it transfers data using two data lines and was standardized by the RMII consortium. To further reduce the required signals, the SMII, or serial mode was developed. All three modes use the same frame structure to transfer data, but because of the reduced number of transmission line, the transfer clock frequency must be increased to maintain the same bit transfer rate. RMII requires a clock twice as fast as the MII clock and SMII requires a clock four times as fast as the MII clock to transfer data at the same rate. All three modes also support the standard MDC and MDIO signals to provide network statistics information.

As an alternative to the **IEEE** 802.3z™ (GMII) Ethernet standard that discusses 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 MSC8144E implements the SGMII through a SerDes interface shared with the Serial RapidIO signals. Refer to **Chapter 16**, *Serial RapidIO® Controller* for details on this shared interface.

Refer to the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM) for complete functional and programming details.

Figure 18-10 illustrates the block diagram of the UCC Ethernet controller.

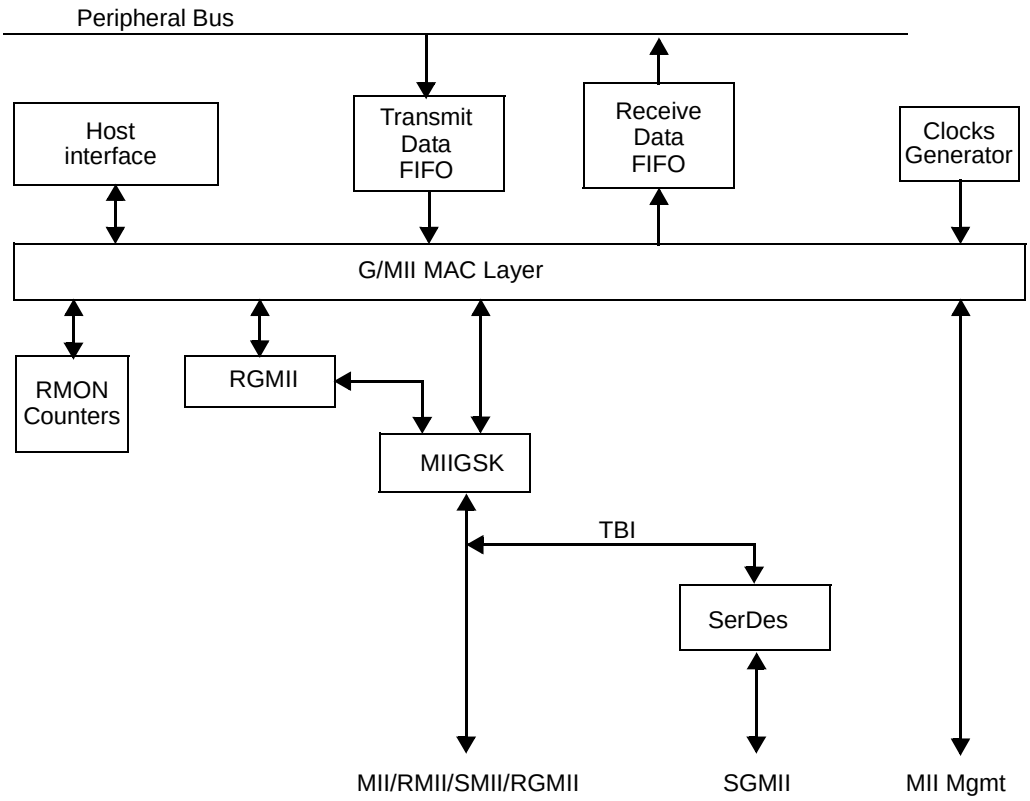


Figure 18-10. UCC Ethernet Controller Block Diagram

The UEC, supports several standard MAC-PHY interfaces to connect to an external Ethernet transceiver.

- 10/100 Mbps MII interface (**IEEE** 802.3-2002 standard)
- 1000 Mbps RGMII interface.
- 10/100 RMII interface.
- 10/100 SMII interface. The SMII interface is supported by the MIIGSK.The MIIGSK controller also supports a SYNC_IN signal.
- 1000 Mbps SGMII interface.

18.7.1 Operating Modes

Each Ethernet controller can a number of Ethernet modes, as described in the following subsections.

18.7.1.1 MII Mode

MII is the media-independent interface defined by the **IEEE** 802.3 standard for 10/100 Mbps operation. The actual transfer speed is determined by the TX_CLK and RX_CLK signals, which are driven by the transceiver. The transceiver either auto-negotiates the speed, or software controls it via the transceiver serial management interface (MDC/MDIO). This mode is only supported by Ethernet controller 1.

18.7.1.2 RMII Mode

The RMII interface (low pin count Reduced Media Independent Interface as specified by the RMII Consortium standard), is a reduced MII interface. The purpose of this interface is to provide a low cost alternative to the MII, with an 8-pin interface instead of 16 (MDC and MDIO pins not included).

18.7.1.3 SMII Mode

SMII is a serial MII interface. The SMII can operate as a MAC-to-PHY or a MAC-to-MAC connection:

- A MAC-to-PHY conveys complete MII information between a 10/100 PHY and MAC using two signals per port, and generates the output SYNC signal to allow a MAC-to-PHY connection. The SMII reference clock generates both transmit and receive clocks for the MII interface clocks.
- For a MAC-to-MAC connection, the SMII interface uses a SYNC input signal to support data synchronization and disables the typical output SYNC signal generation. The SMII reference clock generates both transmit and receive clocks for the MII Mode interface.

18.7.1.4 RGMII Mode

The RGMII is intended to be an alternative to the **IEEE** 802.3u MII, the **IEEE** 802.3z GMII, and TBI standards. The principle objective is to reduce the number of pins required to interconnect the MAC and the PHY from a maximum of 28 pins (TBI) to 12 pins in a cost effective and technology independent manner. In order to accomplish this objective, the data paths and all associated control signals are reduced and control signals are multiplexed together and both edges of the clock are used. For Gigabit operation, the clocks operate at 125 MHz.

18.7.1.5 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 MSC8144E, SGMII is implemented used the SerDes interface.

Note: The internal ten-bit interface (TBI) circuitry is used to serialize/deserialize the Ethernet frame data. The TBI must be configured to perform this function. See the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM) for programming details.

18.7.2 Ethernet Physical Interfaces

You can program the physical interfaces by configuring the PSMR, MIIGSK_CFGR, and MACCFG2 registers. See the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM) for details.

In addition to configuring the Ethernet operating mode, you also have to configure the signal multiplexing so that the specified signals are made available for use. See **Chapter 3**, *External Signals*, **Chapter 5**, *Reset*, **Chapter 8**, *General Configuration Registers*, and **Chapter 22**, *GPIO* for programming details.

Note: The MSC8144E allows adjustment of the transmission delays for the Ethernet signal lines (except SGMII) using GCR4. Recommended settings are listed in the MSC8144E data sheet. Guidelines for adjusting these numbers in individual designs is provided in *Using GCR4 to Adjust Ethernet Timing in MSC8144 DSPs* (AN3811), available at www.freescale.com.

This section defines the communications controller Ethernet MAC-to-device pin I/O, as follows:

- MII requires 18 I/O pins and supports both data and a management interface to the PHY (transceiver) device. The MII option supports both 10 and 100 Mbps Ethernet rates.
- RMII is a reduced pin implementation of the MII (10/100 Mbps).
- RGMII is a reduced pin implementation of the GMII (1000 Mbps = 1 Gbps).
- SMII is a serial implementation of the MII (10/100 Mbps).
- SGMII is a serial implementation of the GMII (1000 Mbps = 1 Gbps).

Table 18-7 lists the external signal properties.**Chapter 3**, *External Signals* describes the assignment of the signals to the device pins.

Table 18-7. Signal Properties

Name	Function	I/O
COL	**MII.** Collision. **RMII, RGMII, SMII**. Not used.	Input
CRS	**MII**. Carrier sense. **RMII, RGMII, SMII.** Not used.	Input
MDC	**Management Data Clock for all Ethernet interfaces** The MDIO signal clock reference (25 MHz clock).	Input
MDIO	**Management Data Input/Output for all Ethernet interfaces** Transfers control signals between the PHY layer and the manger entity.	Input/ Output
RX_CLK	**MII, RGMII.** Receive clock from the PHY **RMII, SMII.** Not used.	Input
RX_DV	**MII**. Receive data valid. **RMII**. Carrier sense data valid (CRS_DV). **RGMII**. Rising edge (receive data valid). **RGMII**. Falling edge (receive error). **SMII**. Not used.	Input Input Input Input —
RXD[0–3]	**MII**. Receive data bits 0–3. **RGMII**. Rising edge (receive data bits 0–3. **RGMII**. Falling edge (receive data bits 4–7). **RMII**. Receive data bits 1–0. **SMII**. Receive data bit 0.	Input Input Input Input Input
RX_ER	**MII**. Receive error. **RMII**. Receive error **RGMII**, **SMII**. Not used.	Input Input —
SRIO_REF_CLK $\overline{\text{SRIO_REF_CLK}}$	**SGMII**: Clock **MII, RMII, SMII, RGMII**: Not used.	Input
GEn_SGMII_TX $\overline{\text{GEn_SGMII_TX}}$	**SGMII**: Transmit data. **MII, RMII, SMII, RGMII**: Not used.	Output
GEn_SGMII_RX $\overline{\text{GEn_SGMII_RX}}$	**SGMII**: Receive data. **MII, RMII, SMII, RGMII**: Not used.	Input
TX_CLK	**MII**. Transmit clock. **RMII**. Reference clock. RGMII Ethernet 1. Oscillator source for transmit clock. RGMII Ethernet 2. Inverted transmit clock feedback. **SMII**. Clock **Note:** This signal usually comes from an external oscillator or PHY.	Input Input Input Output Input
TXD[0–3]	**MII**. Transmit data bits 0–3. **RMII**. Transmit data bits 0–2. **SMII**. Transmit data bit 0. **SMII**. SYNC on TXD[1] **RGMII**. Rising edge (transmit data bits 0–3). **RGMII**. Falling edge (transmit data bit 4–7).	Output Output Output Output Output Output

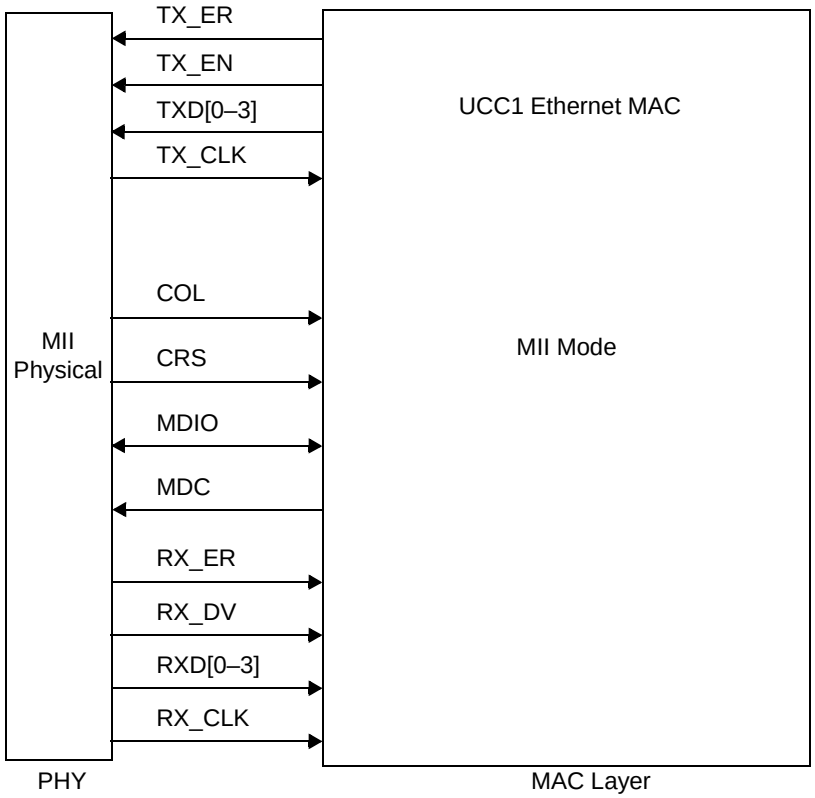
Table 18-7. Signal Properties (Continued)

Name	Function	I/O
TX_EN	**MII**. Transmit data enable. **RMII**. Transmit data enable. **RGMII**. Rising edge (transmit data enable). **RGMII**. Falling edge (transmit error). **RGMII**. SMII. Not used.	Output Output Output Output —
TX_ER	**MII**. Transmit error **RMII, RGMII, SMII**. Not used.	Output —

18.7.3 Media-Independent Interface (MII)

The Ethernet controller implements the **IEEE** Std. 802.3 standard MII interface for fast Ethernet. **Figure 18-11** shows the MII interface signals and the basic components, including the signals required for an Ethernet connection with a PHY.

The speed of operation is determined by the TX_CLK and RX_CLK pins, which are driven by the transceiver. The bit rate of the transceiver is determined either by auto-negotiation or by software via the serial management interface (MDC/MDIO pins) to the transceiver. This mode is only supported by Ethernet controller 1.



Note: Signal names are according to 802.3 standard (total:16 ports).

Figure 18-11. MII MAC-PHY Interface

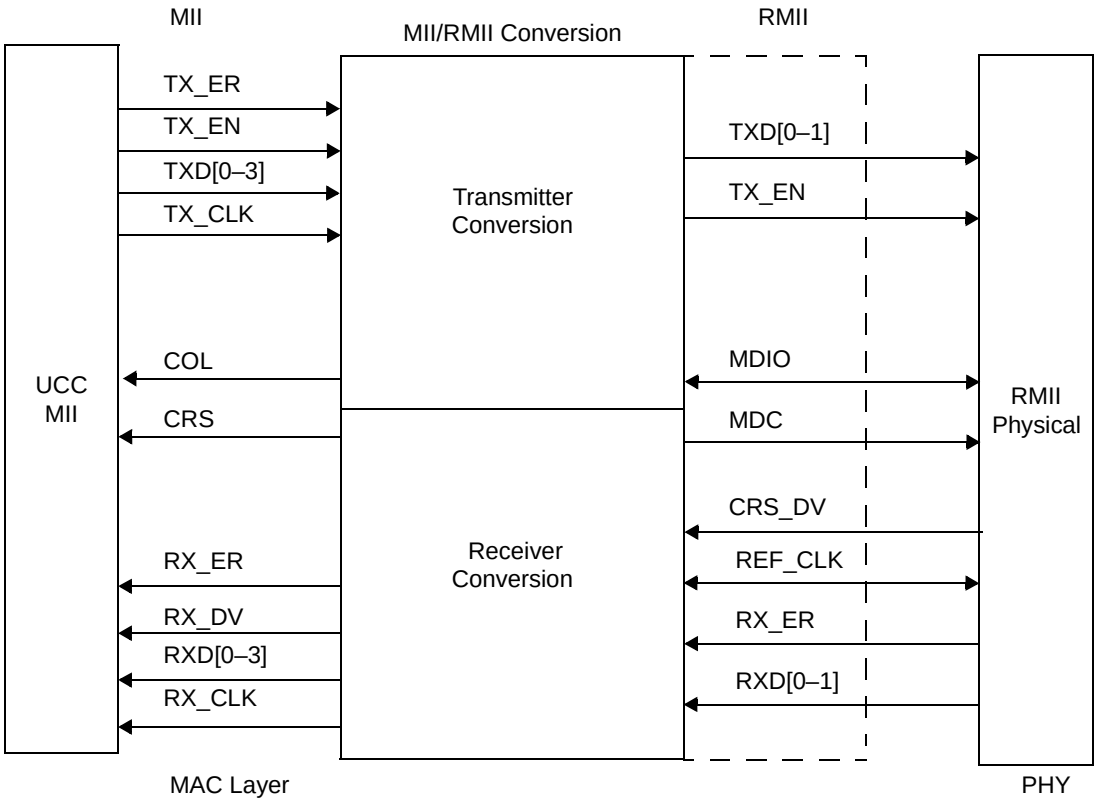
Table 18-7 lists the MII signals.

Table 18-8. MII Signal Descriptions

IEEE Name	I/O	Size	Function	Reference Clock
TX_ER	O	1	**Transmit Error** Asserted by the MAC layer to generate a coding error on the byte currently transferred over TXD[0–3].	TX_CLK
TX_EN	O	1	**Transmit Enable** Asserted by the MAC sublayer when the first transmit preamble byte is driven over the MII. It remains asserted for the remainder of the frame, up to the last CRC byte.	TX_CLK
TXD[0–3]	O	4	**Transmit Data**	TX_CLK
TX_CLK	O	1	**Transmit Clock** A 25 MHz clock that provides the timing reference for the transfer of the TX_EN, TX_ER, and TXD signals.	—
COL	I	1	**Collision** Asserted when a collision is detected.	TX_CLK
CRS	I	1	**CRS** Asserted when the transmit or receive medium is not idle.	—
MDIO	I/O	1	**Management Data Input/Output** Transfers control signals between the PHY layer and the manger entity.	MDC
MDC	I	1	**Management Data Clock** The MDIO signal clock reference (25 MHz clock).	—
RX_ER	I	1	**Receive Error** Asserted by the PHY layer to indicate an error the MAC can not detect. If asserted during frame reception, indicates a coding error on the frame currently being transferred on RXD[0:3].	RX_CLK
RX_DV	I	1	**Receive Data Valid** Asserted by the PHY layer when the first received preamble byte is driving over the MII. It remains asserted for the remainder of the frame, up to the last CRC byte	RX_CLK
RXD[0–3]	I	4	**Receive Data**	RX_CLK
RX_CLK	I	1	**Receive Clock** Synchronizes all receive signals (RX_DV, RXD, RX_ER).	—

18.7.4 Reduced Media-Independent Interface (RMII) Signals

The RMII is defined by the RMII Consortium for 10/100 Mbps operation. In this mode, the Ethernet controller converts MII signals to/from RMII signals. A single clock reference is sourced from the MAC to the PHY (or from an external source). **Figure 18-12** shows the basic components of the RMII, including the signals required to make an Ethernet module connection with a PHY.



Note: Signals names are according to the **IEEE** 802.3 standard (total: 8 ports).

Figure 18-12. RMII MAC-PHY Interface

Table 18-9 lists the RMII signals.

Table 18-9. RMII Signals

Consortium Name	I/O	Size	Function	Reference Clock
TXD[0–1]	O	2	**Transmit Data**	REF_CLK
TX_EN	O	1	**Transmit Enable** Asserted by the MAC sublayer when the first transmit preamble byte is driven over the RMII. It remains asserted up to the last CRC byte	REF_CLK
MDIO	I/O	1	**Management Data I/O** Transfers control signals between the PHY layer and the manger entity.	MDC
MDC	I	1	**Management Data Clock** The MDIO signal clock reference (25 MHz clock).	—
CRS_DV	I	1	**Receive Data Valid** Asserted by the PHY layer when the first received preamble byte is driving over the RMII. It remains asserted f up to the last CRC byte.	REF_CLK
REF_CLK	O	1	**Reference Clock** 50 MHz synchronous clock reference for receive, transmit, and control interface.	—
RX_ER	I	1	**Receive Error** Asserted by the PHY layer to indicate an error the MAC cannot detect. If it is asserted during frame reception, it indicates a coding error on the frame currently transferred on RXD[1–0].	REF_CLK
RXD[0–1]	I	2	**Receive Data**	REF_CLK

18.7.5 SMII Interface

The Ethernet controller SMII receive and transmit interface complies with the Cisco serial MII specification. In this mode, the controller supports a MAC-to-PHY or a MAC-to-MAC connection:

- *SMII MAC-to-PHY interface*. Conveys complete MII information between a 10/100 PHY and MAC using two signals per port and generates the output SYNC signal to allow a MAC-to-PHY connection. The SMII reference clock generates both transmit and receive clocks for the MII interface clocks. To configure the Ethernet controller, write a value of 0b10 to MIIGSKCFGR[IFMODE] and select the SYNC output signal by writing a 0 to MIIGSK_SMII_SYNCDIR[SYNC_IN] and a 1 to MIIGSK_SMII_SYNCDIR[SYNC]. The operating mode is determined by the Frequency Control bit (MIIGSK_CFGR[FRCONT]); the default value 0 selects 100 Mbps operation. For 10 Mbps operation, set the frequency by writing a 1 to MIIGSK_CFGR[FRCONT].
- *MAC-to-MAC connection*. The SMII uses a SYNC input signal to support data synchronization and disables the typical output SYNC signal generation. Select this mode by writing a value of 0b10 to MIIGSK_CFGR[IFMODE] and selecting the ETHSYNC_IN input by writing a 1 to MIIGSK_SMII_SYNCDIR[SYNC_IN] and a 0 to MIIGSK_SMII_DYNCDIR[SYNC]. The operating mode is determined by the Frequency Control bit (MIIGSK_CFGR[FRCONT]); the default value 0 selects 100 Mbps operation.

For 10 Mbps operation, set the frequency by writing a 1 to MIIGSKCFGR[FRCONT]. The SMII reference clock generates both transmit and receive clocks for the MII interface.

Figure 18-13 and **Figure 18-14** show the PHY and MAC connections in SMII mode.

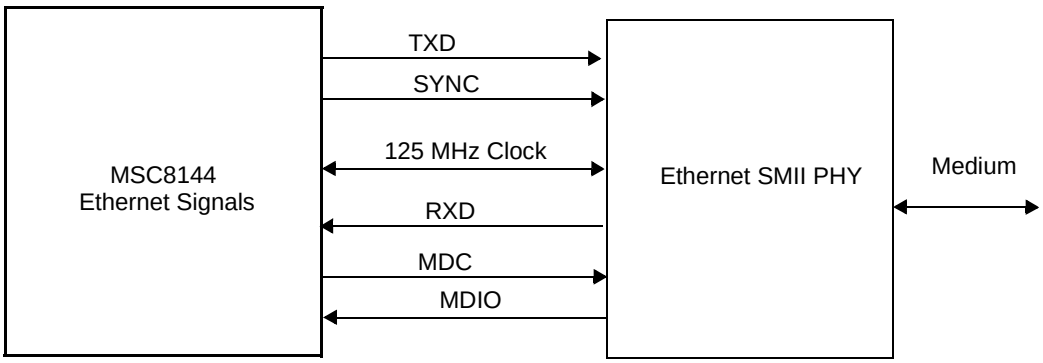


Figure 18-13. Ethernet Controller-to-PHY Connection in SMII Mode

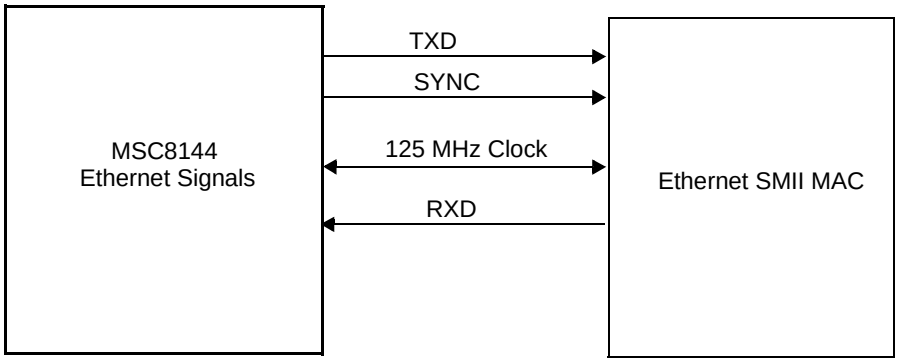


Figure 18-14. Ethernet Controller-to-MAC Connection in SMII Mode

18.7.5.1 Reduced Gigabit Media-Independent Interface (RGMII) Signals

The RGMII is an alternative to the **IEEE** Std. 802.3u MII, the **IEEE** Std. 802.3z GMII, and the TBI. It reduces the number of signal pins that connect the MAC and PHY from a maximum of 26 pins (GMII) to 13 pins (GTX_CLK included). The data paths and all associated control signals are reduced, control signals are multiplexed, and both edges of the clock are used. **Figure 18-15** shows the RGMII connections between the Ethernet controller and a PHY.

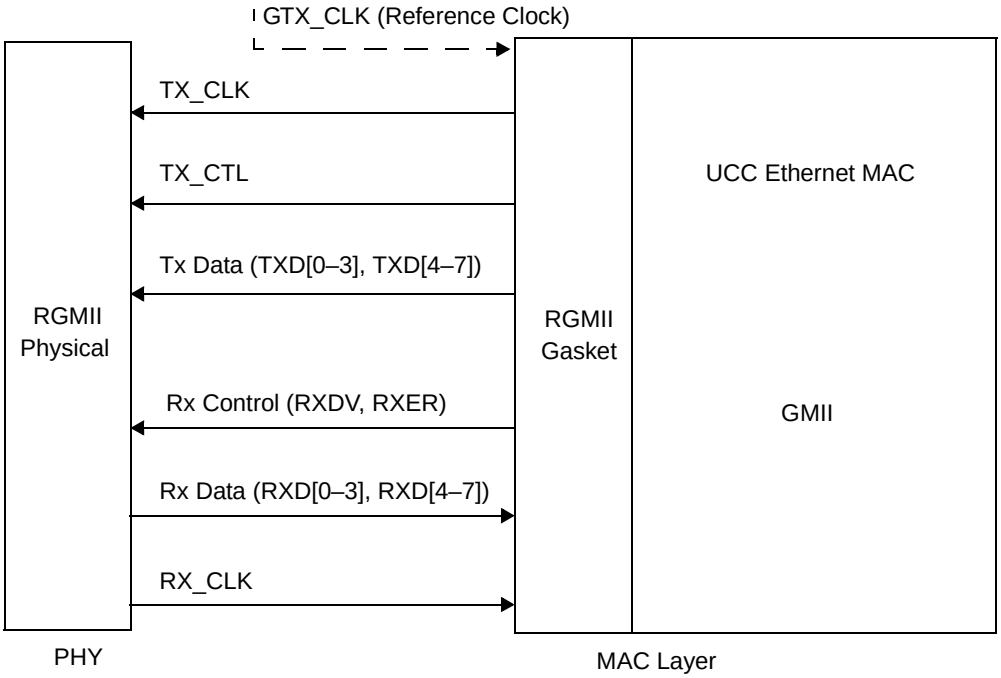


Figure 18-15. RGMII MAC-PHY Interface

Table 18-10 lists the RGMII signals.

Table 18-10. RGMII Signals

Consortium Name	I/O	Size	Function	Reference Clock
GTX_CLK	I	1	**Transmit Reference Clock** 125 MHz	—
TX_CLK	O	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
RX_CTL	O	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

Table 18-10. RGMII Signals (Continued)

Consortium Name	I/O	Size	Function	Reference Clock
MDC	I	1	**Management Data Clock** The MDIO signal clock reference (25 MHz clock).	—
RX_CLK	I	1	**Continuous Receive Reference Clock** 125 MHz	—

18.7.5.2 Serial Gigabit Media-Independent Interface (SGMII) Signals

The SGMII is an alternative to the RGMII interface that further reduces the number of pins required to interconnect the MAC and PHY by using a SerDes 4 interface. It does not support auto-negotiation. The Ethernet controller SGMII interface uses the UEC ten-bit interface (TBI) connection internally, which connects to the SerDes block that serializes the transmitted data and deserializes the received data to/from the SGMII interface.

18.7.5.2.1 SGMII Signals

The SGMII physical interface transmits and receives data using two data signals and a clock signals to convey frame data and link rate information between a 1000 Mbps PHY and an Ethernet MAC. The data signals operate at 1.25 Gbaud. Due to the speed of operation, each of these signals (including the clock signal) is realized as a differential pair thus providing signal integrity while minimizing system noise. Therefore, each data and clock signal path uses two physical signal lines (the differential pair). **Table 18-11** lists the SGMII signals.

Table 18-11. SGMII Signals

Signal Name	I/O	Size	Function	Reference Clock
SRIO_REF_CLK $\overline{\text{SRIO_REF_CLK}}$	I I	2	**Reference Clock** 125 MHz differential pair.	—
SG1_TX $\overline{\text{SG1_TX}}$	O O	2	**Transmit Data 1** Differential pair for Ethernet 1 controller.	$\overline{\text{SRIO_REF_CLK}}$ SRIO_REF_CLK
SG2_TX $\overline{\text{SG2_TX}}$	O O	2	**Transmit Data 2** Differential pair for Ethernet 2 controller.	$\overline{\text{SRIO_REF_CLK}}$ SRIO_REF_CLK
SG1_RX $\overline{\text{SG1_RX}}$	I I	2	**Receive Data 1** Differential pair for Ethernet 1 controller.	$\overline{\text{SRIO_REF_CLK}}$ SRIO_REF_CLK
SG2_RX $\overline{\text{SG2_RX}}$	I I	2	**Receive Data 2** Differential pair for Ethernet 2 controller.	$\overline{\text{SRIO_REF_CLK}}$ SRIO_REF_CLK
MDIO	I/O	1	**Management Data I/O** Transfers control signals between the PHY layer and the manager entity.	MDC
MDC	I	1	**Management Data Clock** The MDIO signal clock reference (25 MHz clock).	—

18.7.6 Controlling PHY Links (Management Interface)

The support for MII Ethernet Management can be done by the SPI or by one UCC that can be selected using CMXGCR[SMI]. Control and status to and from the PHY is provided via the two-wire MII management interface described in the **IEEE** 802.3u standard. The MII management registers (MII management configuration, command, address, control, status, and indicator registers) exercise this interface between a host processor and one or more PHY devices.

The UEC MII registers support continuous read cycles (called a scan cycle); even through scan cycles are not explicitly defined in the standard. If requested (by setting MIIMCOM[scan cycle]), the controller performs repetitive read cycles of the PHY status register, for example. This allows you to monitor link characteristics more efficiently. The different fields in the MII management indicator register (scan, not valid, and busy) indicate availability of each read of the scan cycle to the host via MIIMSTAT[PHY scan] bit field.

The length of the MII management interface preamble can also be modified through the MII registers. After establishing that a PHY supports preamble suppression, the host may configure the UEC to suppress the preamble. When enabled, the length of MII management frames are reduced from 64 to 32 clocks. This effectively doubles the efficiency of the interface.

18.7.7 Ethernet Controller Initialization

After the Ethernet Controller completes the reset sequence, software must initialize certain UEC registers and the required parameters in the parameter RAM. Based on system requirements, other optional registers and parameters can also be initialized at the same time.

Table 18-12 lists the minimum steps required for register and parameter initialization

Table 18-12. Minimum Register Initialization

Initialization Step	Registers
Configure the UCC to Fast protocols	URMODE,UTMODE
Set the Tx Global Parameter RAM	
Set the Rx Global Parameter RAM	
Set CMXUCR1	Select UCC1, UCC3 RxClk and TxClk
Initialize the MAC Station address	MACSTNADDR1 and MACSTNADDR2
Initialize the Media media access control configuration register and the UCC protocol specific mode register	This MACCFG2 together with UPSMR register adjust frame length and preamble length, specifies various CRC/pad combinations, specifies Full/Half Duplex and operating mode
Initialize the Fast Protocol Fifo Configuration registers	URFB, URFET, URFS, URFSET, UTFB, UTFS, UTFET, UTFTT, URTRY
UCC event register, Fast UCCE	Initialize interrupts to prepare for interrupt events
UCC mask register, Fast UCCM	Initialize the interrupt mask to prepare for interrupt events
Activate The Ethernet Controller	

Table 18-12. Minimum Register Initialization (Continued)

Initialization Step	Registers
Initialize the InitEnet parameter	CECDR
Initialize the Tx and Rx parameters of UCC1 ethernet.	CECR
Enable the Ethernet Controller MAC TX and RX	MACCFG1
Note: See the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM) for register addressing, structure, and programming details.	

After initializing the registers, you must complete the following steps to bring the Ethernet Controller into a functional state:

1. To transmit Ethernet frames, build the TxBDs in memory, link them together as a ring, and point to the ring. A minimum of two TxBDs per ring is required.
2. To receive Ethernet frames, link the RxBDs together as a ring and point the corresponding registers to them. Both transmit and receive can be gracefully stopped after transmission and reception begins.

18.8 Asynchronous Transfer Mode (ATM) Controller

The MSC8144E uses UCC5 to support one ATM controller managed through the QUICC Engine subsystem. UCC5 manages the data flow internally through its receive and transmit FIFOs. The QUICC Engine subsystem performs frame control and manipulation using firmware executed by the RISC engines according to the specified protocol requirements. External data flow is managed using a separate, associated UTOPIA/POS bus controller (UPC) to a UTOPIA or POS interface. The UPC can operate in master and slave modes.

The QUICC Engine subsystem supports the following ATM applications:

- ATM line card controllers
- ATM-to-WAN interworking (frame relay, T1/E1 circuit emulation)
- Residential broadband network interface units (NIU) (ATM-to-Ethernet)
- High-performance ATM network interface cards (NIC)
- Bridges and routers with ATM interface

18.8.1 Background

Asynchronous transfer mode (ATM) was developed as an international standard to support consistent, reliable, and uninterrupted data transmission world-wide. Data transmission is performed asynchronously because the distance over which the transfer occurs prevents the use of the same clock signal at both ends and even at points within the transfer network. Each network consists of user end stations that transmit and receive the specified unit of transfer, called a cell, using virtual connections. The original protocol used 8-bit data transfers and the cell size was 53 bytes, which included 48 bytes of data and a 5 byte header. As technology has

advanced to support 16-bit and larger transfers, the cell size has increased, with the additional bytes being added to the header. The header portion of the cell includes several sections of identifying information used by the cell processor to route and transmit the cell, including the type of data being transmitted and the origin and destination. As with other modes of electronic transfer, the individual cells are combined with other cells (for efficiency of transfer) and transmitted over long distances. During the transfer, cells are evaluated and, if required, separated from transmission stream and rerouted, similar to commuters who share transportation routes during parts of their journeys and then transfer to alternate routes to complete their journey. A special type of cell is used for operation, administration, and maintenance (OAM cells) at the physical level within the ATM network. The OAM cells can be used to perform continuity checking, fault testing and failure notification, and timing functions for the transfers.

Each ATM network consists of user end stations that transmit and receive the data cells using virtual connections. Virtual connections are implemented through physical links and switching elements that interconnect them. The specific combination of physical links that implements a virtual connection is chosen when the connection is established. For a given physical link, each connection is assigned a unique connection identifier. The connection identifier is placed in the header of each cell by the transmitting equipment and is used by the receiving equipment to route the cell to the next physical link on the connection path. All cells belonging to a specific virtual connection follow the identical path from the transmitting end station through the switching elements to the receiving end station.

The signal connections for an ATM network are defined by the Universal Test and Operation Physical-Layer for ATM (UTOPIA) protocol. The protocol has multiple levels that reflect the data transfer size. Level 1 supports 8-bit transfers with no addressing. Level 2 supports both 8-bit and 16-bit transfers with addressing and single or multi-PHY support. Higher levels support larger transfer sizes.

In addition to the UTOPIA interface, the ATM network uses a variety of data handling protocols to interface with the higher logical levels in the network. This ATM Adaptation Layer (AAL) relays ATM cells between the ATM Layer and higher layers. When relaying information received from the higher layers, it segments the data into ATM cells. When relaying information received from the ATM Layer, it must reassemble the payloads into a format the higher layers can understand. This operation, which is called Segmentation and Reassembly (SAR), is the main task of AAL. Different AALs are defined to support different traffic or services.

18.8.2 ATM Controller Architecture

The ATM interface manages ATM network operations at three levels:

- UCC5 provides the FIFOs for internal data storage and flow
- RISC operations manipulate data as required by the supported protocol
- The UPC maintains traffic flow on the ATM network (UTOPIA or POS)

Figure 18-16 shows the basic architecture of the ATM controller.

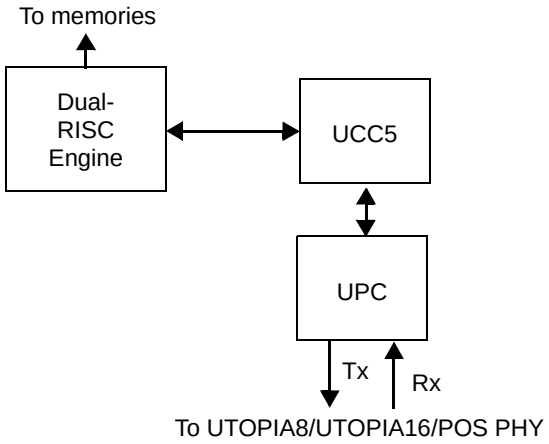


Figure 18-16. ATM Control Architecture

See the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM) for details on configuring and programming ATM operations.

18.8.3 UTOPIA Physical Interfaces

The UPC external signal descriptions are divided into UTOPIA mode and POS mode signal groups. There are several bus configurations, depending on the implemented protocol (UTOPIA/POS), the polling method, number of devices in the UTOPIA/POS configuration, and the data bus width. For example, one UTOPIA device configuration requires a total of 36 I/O ports for 8-bit mode and 52 bits in 16-bit mode. **Table 18-13** summarizes all possible I/O pins assignments combinations.

Table 18-13. UCC UTOPIA/POS I/O Pin Count

	UTOPIA 31 PHYs	POS 31 PHYs
8 bit Data	16	16
16 bit Data	32	32
parity	2	2
Address	10	10
Clav/xPTA	2	4[1]
Clock	2	2

Table 18-13. UCC UTOPIA/POS I/O Pin Count

	UTOPIA 31 PHYs	POS 31 PHYs
Framing	2	8[2]
Enable	2	2[3]
Total for 8 bit	36	42[4]
Total for 16 bit	52	60

1. PTPA, STPA, RVAL, PRPA
2. TSOP, TEOP, RSOP, REOP, TMOD, RMOD, TERR, RERR
3. TENB, RENB
4. Non standard solution. RMOD, TMOD not needed

Note: UTOPIA signal lines are multiplexed with those used by other peripheral blocks. See **Chapter 3**, *External Signals* for details on which signals are multiplexed and how to select them for ATM operation.

The following subsections define the available UTOPIA configurations.

18.8.4 UTOPIA Interface Master Mode

UTOPIA master signals are shown in **Figure 18-17**.

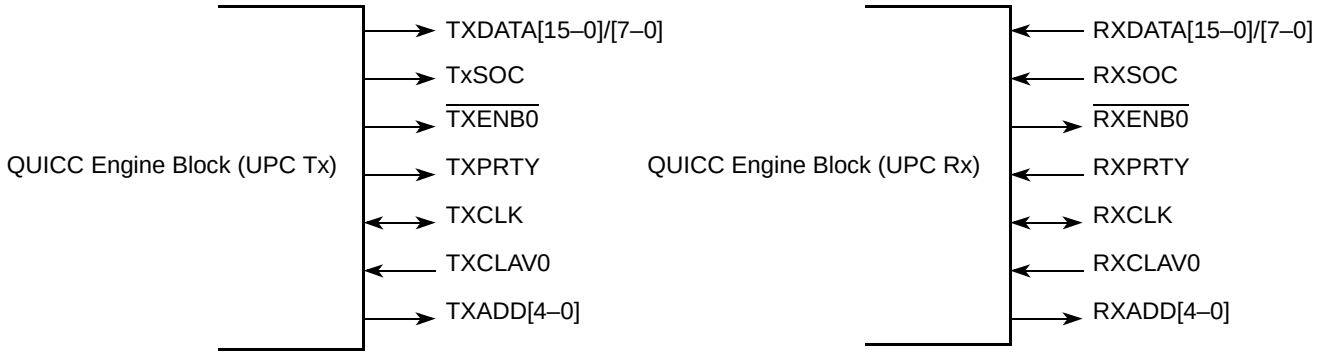


Figure 18-17. UTOPIA Master Mode Signals

See **Chapter 3**, *External Signals* for detailed signal descriptions.

18.8.5 UTOPIA Interface Slave Mode

There are two UTOPIA level 2 bus configurations, depending on the data bus width. For example, UTOPIA device configuration requires a total of 36 I/O ports for 8-bit mode and 52 bits in 16-bit mode. Both configuration use the same number of control signals; only the data signal lines are different. UTOPIA slave signals are shown in <Cross Refs>Figure 18-18.

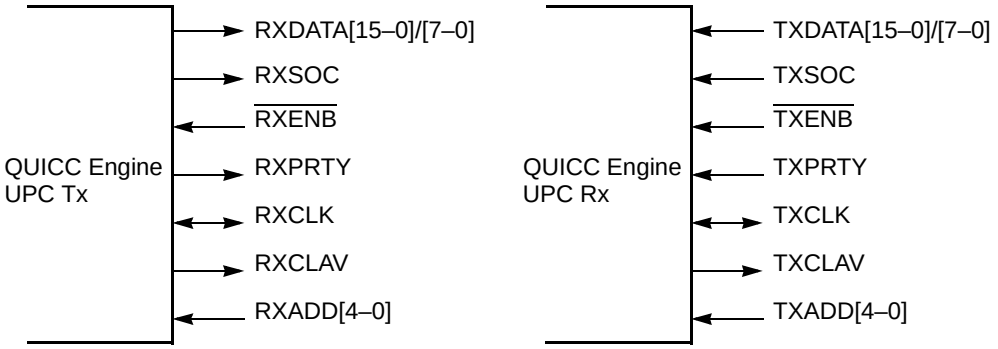


Figure 18-18. UTOPIA Slave Mode Signals

See **Chapter 3**, *External Signals* for detailed signal descriptions.

18.8.6 POS Interface Master Mode

POS master signals are shown in **Figure 18-19**.

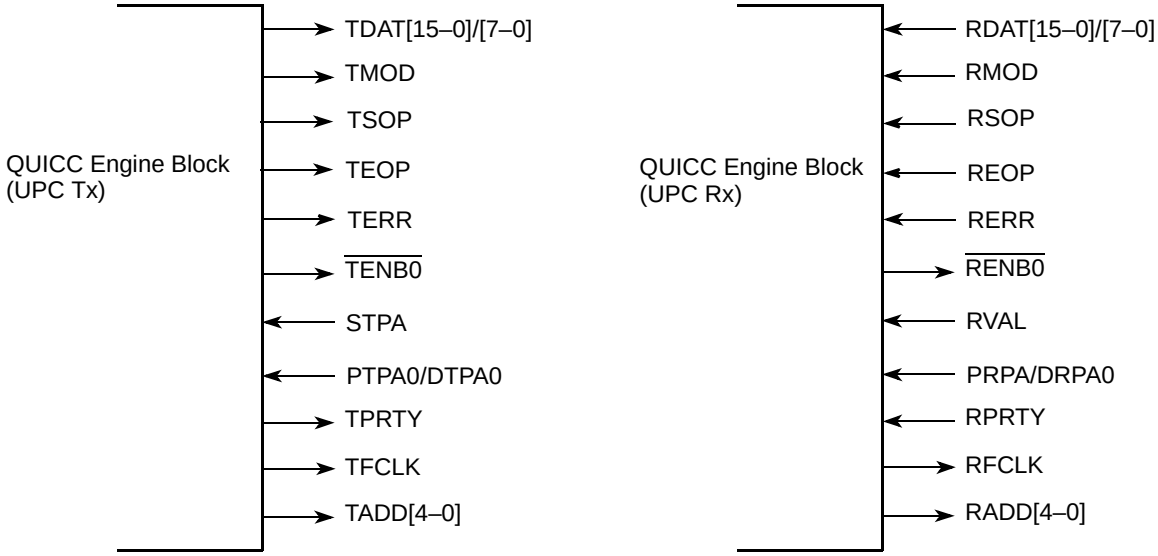


Figure 18-19. POS Master Mode Signals

See **Chapter 3**, *External Signals* for detailed signal descriptions.

18.8.7 POS Interface Slave Mode

POS slave signals are shown in **Figure 18-20**.

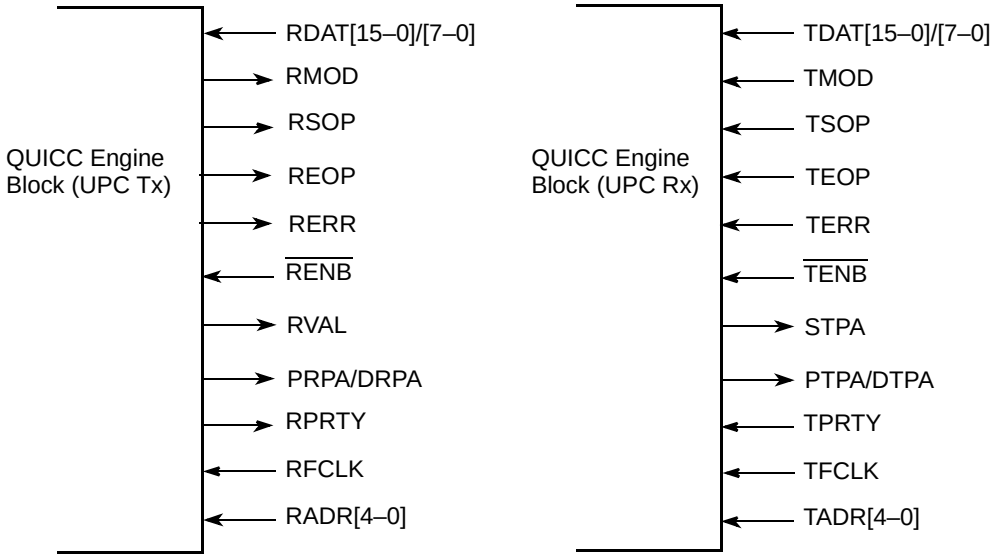


Figure 18-20. POS Slave Mode Signals

See **Chapter 3**, *External Signals* for detailed signal descriptions.

18.9 Serial Peripheral Interface (SPI)

The serial peripheral interface (SPI) allows the exchange of data with other devices containing an SPI. The SPI also communicates with peripheral devices such as EEPROMs, real-time clocks, A/D converters, and ISDN devices. The SPI is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface (receive, transmit, clock, and slave select). The SPI block consists of transmitter and receiver sections, an independent baud-rate generator, and a control unit. The transmitter and receiver sections use the same clock, which is derived from the SPI baud rate generator in master mode and generated externally in slave mode. During an SPI transfer, data is sent and received simultaneously. The SPI receiver and transmitter are double-buffered, as shown in **Figure 18-21**, 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. The SPI operates in QUICC Engine mode. In QUICC Engine mode SPI, which is compatible to the MPC826x SPI, and is controlled by QUICC Engine RISC.

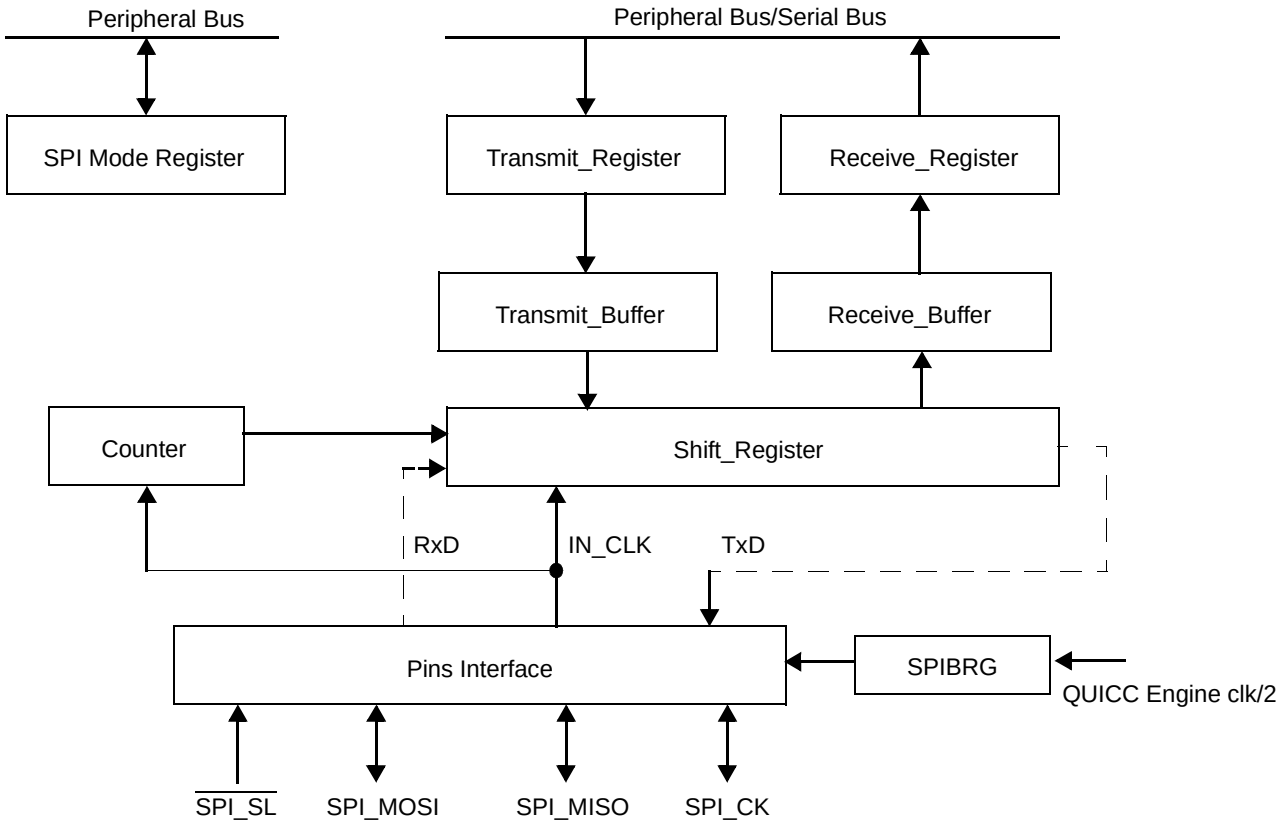


Figure 18-21. SPI Block Diagram

18.9.1 SPI Operating Modes

The SPI can be programmed to work in a single- or multiple-master environment. This section describes the SPI master and slave operation in a single-master configuration and then discusses the multi-master environment. The following sections present a summary of the main modes of operation which the SPI supports.

18.9.1.1 SPI as a Master Device

In master mode, the SPI sends a message to the slave peripheral, which sends back a simultaneous reply. A single-master device with multiple slaves can use general-purpose parallel I/O signals to selectively enable slaves, as shown in **Figure 18-22**. To eliminate the multi-master error in a single-master environment, the master $\overline{\text{SPI_SL}}$ input can be forced inactive by selecting $\overline{\text{SPI_SL}}$ for general-purpose I/O.

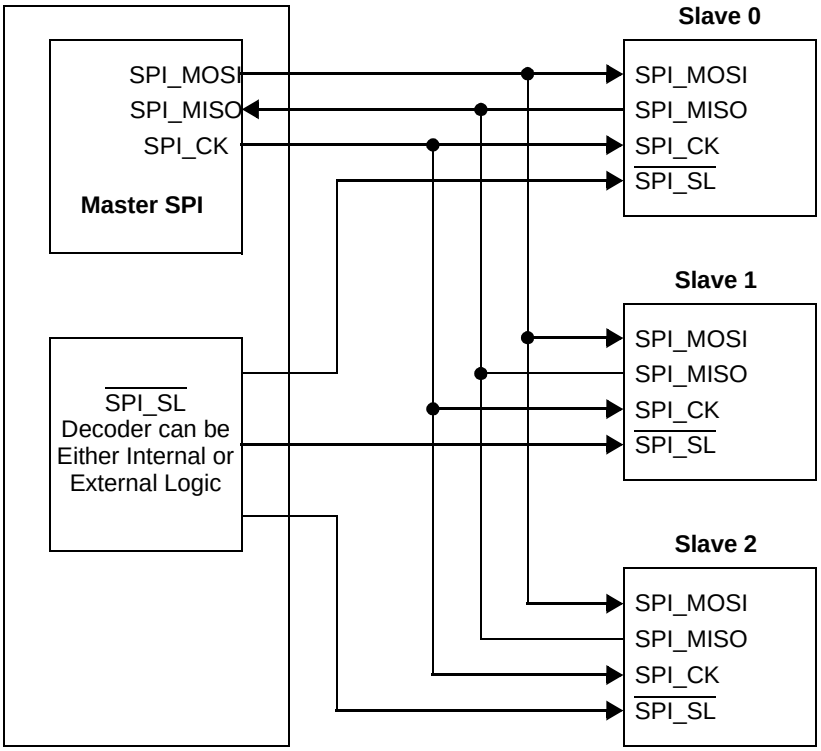


Figure 18-22. Single-Master/Multi-Slave Configuration

To start exchanging data, the QUICC Engine subsystem writes the data to be sent into a buffer, configures a TxBD with TxBD[R] set, and configures one or more RxBDs. The QUICC Engine subsystem then sets SPCOM[STR] in the SPI command register to start sending data, which starts once the SDMA channel loads the Tx FIFO with data.

The SPI then generates programmable clock pulses on SPI_CK for each character and simultaneously shifts Tx data out on SPI_MOSI and Rx data in on SPI_MISO. Received data is written into a Rx buffer using the next available RxBD. The SPI keeps sending and receiving characters until the whole buffer is sent or an error occurs. The QUICC Engine subsystem then clears TxBD[R] and RxBD[E] and issues a maskable interrupt to the interrupt controller.

When multiple TxBDs are ready, TxBD[L] determines whether the SPI keeps transmitting without SPCOM[STR] being set again. If the current TxBD[L] is cleared, the next TxBD is processed after data from the current buffer is sent. Typically, there is no delay on SPI_MOSI between buffers. If the current TxBD[L] is set, sending stops after the current buffer is sent. In addition, the RxBD is closed after transmission stops, even if the Rx buffer is not full; therefore, Rx buffers need not be the same length as Tx buffers.

18.9.1.2 SPI as a Slave Device

In slave mode, the SPI receives messages from an SPI master and sends a simultaneous reply. The slave's SPI_SL must be asserted before Rx clocks are recognized; once 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 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 SPI_SL is deasserted.

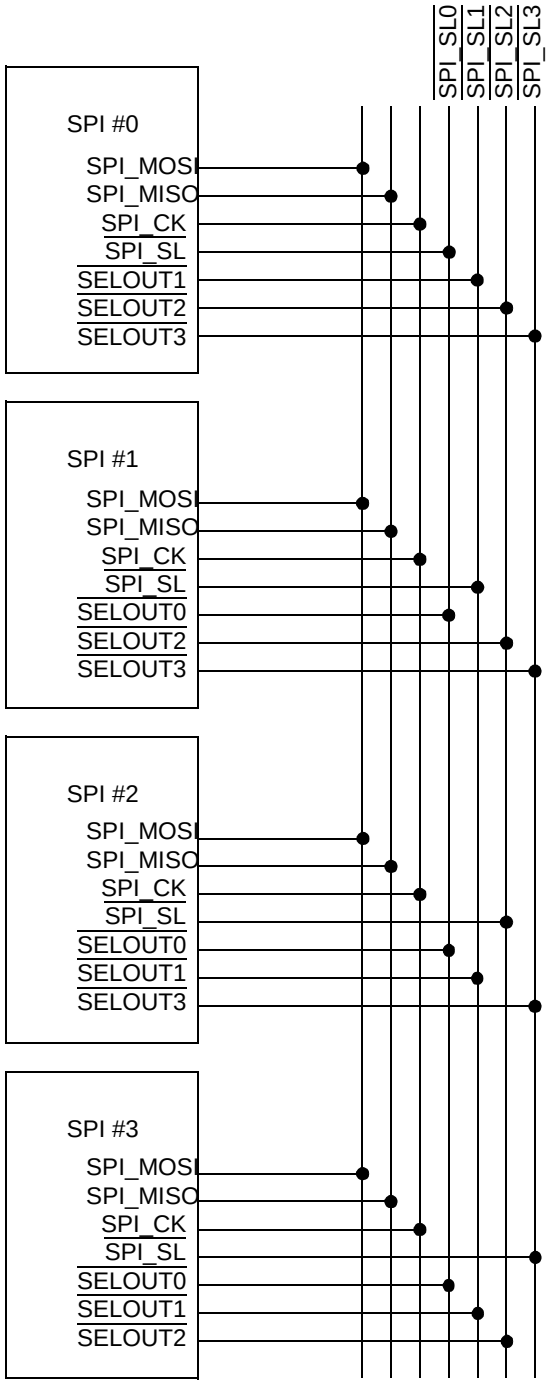
Transmission continues until no more data is available or SPI_SL is deasserted. If it is deasserted before all data is sent, it stops but the TxBD stays open. Transmission continues once SPI_SL is reasserted and SPI_CK begins toggling. After the characters in the buffer are sent, the SPI sends ones as long as SPI_SL remains asserted.

Note: When enabling the SPI or changing parameters in SPI Mode Register (like CP,CI), SPI_SL must remain deasserted for at least 2 QUICC Engine clk/2 clocks afterwards. Also if SPI_SL is deasserted between transfers, its deassertion time should be at least 2 QUICC Engine clk/2 clocks.

18.9.2 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 SPI_SL inputs are connected separately, as shown in **Figure 18-23**. 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 SPI_SL input is asserted, a multi master error occurs because more than one SPI device is a bus master. The SPI sets SPIE[MME] in the SPI event register and a maskable interrupt is issued to the QUICC Engine subsystem. It also disables SPI operation and the output drivers of SPI signals. The core processor must clear SPMODE[EN] before the SPI is used again. After correcting the problems, clear SPIE[MME] and re-enable the SPI.

The maximum sustained data rate that the SPI supports is QUICC Engine clk/50. However, the SPI can transfer a single character at much higher rates—QUICC Engine clk/8 in master mode and QUICC Engine clk/4 in slave mode. Gaps should be inserted between multiple characters to keep from exceeding the maximum sustained data rate.



Notes:
- All signals are open-drain
- For a multi-master QUICC Engine subsystem with more than two masters, SPI_SL and SPIE[MME] will not detect all possible conflicts.
- It is the responsibility of the software to arbitrate for the SPI bus (with token passing, for example).
- SPI_SLx signals are implemented in the software with general-purpose I/O signals.

Figure 18-23. Multimaster Configuration

18.9.3 External Signal Configuration

The SPI supports a four-wire interface—transmit, receive, clock, and slave select. See **Chapter Figure 3-1.**, *MSC8144 External Signals* for detailed signal descriptions. After reset, the signals are assigned as GPIO17 to GPIO20. They must be configured as SPI signals by writing the correct values to the GPIO configuration registers. Refer to **Table 3-11** on page 3-41 and **Chapter 22**, *GPIO* for programming information.

The SPI can be configured as a slave or as a master in single- or multiple-master environments. The master SPI generates the transfer clock SPI_CK, using the SPI baud rate generator (BRG). The SPI BRG input is QUICC Engine clk /2. The selection as slave or master determines the signal direction (input or output) to select when configuring the signals using the GPIO configuration registers.

SPI_CK is a gated clock, active only during data transfers. Four combinations of SPI_CK phase and polarity can be configured by using SPMODE[CI, CP]. SPI signals can also be configured as open-drain to support a multimaster configuration in which a shared SPI signal is driven by the processor or an external SPI device.

The SPI master-in slave-out SPI_MISO signal acts as an input for master devices and as an output for slave devices. Conversely, the master-out slave-in SPI_MOSI signal is an output for master devices. The dual functionality of these signals allows the SPIs in a multimaster environment to communicate with one another using a common hardware configuration.

- When the SPI is a master, SPI_CK is the clock output signal that shifts received data in from SPI_MISO and transmitted data out to SPI_MOSI. SPI masters must output a slave select signal to enable SPI slave devices by using a separate general-purpose I/O signal. Assertion of $\overline{\text{SPI_SL}}$ while it is a master causes an error.
- When the SPI is a slave, SPI_CK is the clock input that shifts received data in from SPI_MOSI and transmitted data out through SPI_MISO. $\overline{\text{SPI_SL}}$ is the input enable to the SPI slave. In a multi-master environment, $\overline{\text{SPI_SL}}$ (always an input) is also used to detect an error when more than one master is operating.

18.9.4 SPI Transmission and Reception Process

Because the SPI is a character-oriented communication unit, the core processor must pack and unpack the receive/transmit frames. A frame consists of all of the characters transmitted or received during a completed SPI transmission session, from the first character written to the internal SPI transmit data register (SPITD) to the last character transmitted following the setting of the LST bit in the SPI command (SPCOM) register.

The core processor receives data by reading the SPI receive data register (SPIRD) when the SPI event register (SPIE) not-empty bit (SPIE[NE]) is set. The core processor transmits data by writing it into the SPITD. When the next character to transmit is the final one in the current

frame, the core processor sets the last (SPCOM[LST]) bit and then writes the final character to SPITD. The SPI sets the not-full (SPIE[NF]) bit whenever its transmit FIFO is not full. It clears the bit when the last character is written to SPITD and resets it after sending the last data.

The SPI-core processor handshake protocol can use a polling or interrupt mechanism. When using polling, the core processor reads the SPIE at a predefined frequency and acts according to the value of the SPIE bits. The polling frequency depends on the SPI serial channel frequency. When using the interrupt mechanism, setting either SPIE[NF] or SPIE[NE] causes an interrupt to the core processor. The core processor then reads the SPIE and acts accordingly. There are three basic modes of operation for transmitting and receiving: master, slave, and multimaster.

18.10 Programming Model

This section provides a summary list of the MSC8144E QUICC Engine subsystem, Ethernet controller, ATM, and SPI registers with their offsets.

Note: The QUICC Engine registers use a base address of 0xFEE00000.

Table 18-14. MSC8144E 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—YCC Peripherals	CIPYCC.	0x0094
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

Table 18-14. MSC8144E QUICC Engine Register Summary (Continued)

Register Name	Acronym	Offset
QUICC Engine Command Data Register	CECDR	0x0108
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
CMX Clock Route Register 1	CMXUCR1	0x0410
CMX Clock Route Register 2	CMXUCR2	0x0414
CMX UPC Clock Route Register	CMXUPCR	0x0420
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

Table 18-14. MSC8144E QUICC Engine Register Summary (Continued)

Register Name	Acronym	Offset
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
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

Table 18-14. MSC8144E QUICC Engine Register Summary (Continued)

Register Name	Acronym	Offset
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
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

Table 18-14. MSC8144E QUICC Engine Register Summary (Continued)

Register Name	Acronym	Offset
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
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
Ethernet 2 Tx Frames	E2TxframeMin	0x2380
Ethernet 2 Tx Frames MINLength	E2TxPkts65	0x2384
Ethernet 2 Tx Frames 128	E2TxPkts128	0x2388
Ethernet 2 Rx Frames Minimum	E2EtherStatsframe64	0x238C
Ethernet 2 Rx Frames MINLength	E2EtherStatsPkts65	0x2390
Ethernet 2 Rx Frames 128	E2EtherStatsPkts128	0x2394
Ethernet 2 Octet Transmitted OK	E2OcTxOK	0x2398
Ethernet 2 Tx Pause Frames	E2PausFrTx	0x239C
Ethernet 2 Multicast Frame Transmitted OK	E2MulCastFrTxOK	0x23A0
Ethernet 2 Broadcast Frames Transmitted OK	E2BroadCastFrTxOK	0x23A4
Ethernet 2 Number of Frames Received OK	E2FrRxOK	0x23A8
Ethernet 2 Rx Octets OK	E2OCRxOK	0x23AC
Ethernet 2 Rx Octets	E2EtherStatsOctet	0x23B0
Ethernet 2 Multicast Frame Received OK	E2MulCastFrRxOK	0x23B4
Ethernet 2 Broadcast Frames Received OK	E2BroadCastFrRxOK	0x23B8
UCC 5 Mode Register	GUMR5	0x2400
UCC 5 Protocol Specific Mode Register	UPSMR5	0x2404
UCC 5 Transmit-on-Demand Register	UTODR5	0x2408
UCC 5 Event Register	UCCE5	0x2410
UCC 5 Mask Register	UCCM5	0x2414
UCC 5 Status Register	UCCS5	0x2418
UCC 5 Receive l FIFO Base	URFB5	0x2420
UCC 5 Receive FIFO Size	URFS5	0x2424
UCC 5 Receive FIFO Emergency Threshold	URFET5	0x2428

Table 18-14. MSC8144E QUICC Engine Register Summary (Continued)

Register Name	Acronym	Offset
UCC 5 Receive FIFO Special Emergency Threshold	URFSET5	0x242A
UCC 5 Transmit FIFO Base	UTFB5	0x242C
UCC 5 Transmit FIFO Size	UTFS5	0x2430
UCC 5 Transmit FIFO Emergency Threshold	UTFET5	0x2434
UCC 5 Transmit FIFO Transmit Threshold	UTFTT5	0x2438
UCC 5 Transmit Polling Timer	UFPT5	0x243C
UCC 5 Retry Counter	URTRY5	0x2440
UCC 5 General Extended Mode Register	GUEMR5	0x2490
MIIGSK Configuration Register 1	MIIGSK1_CFGR	0x2800
MIIGSK Enable Register 1	MIIGSK1_ENR	0x2808
MIIGSK SMII SYNC Direction Register 1	MIIGSK1_SMII_SYNCDIR	0x280C
MIIGSK SMII Transmit Inter Frame Bits Register 1	MIIGSK1_TIFBR	0x2810
MIIGSK SMII Receive Inter Frame Bits Register 1	MIIGSK1_RIFBR	0x2814
MIIGSK SMII Expected Receive Inter Frame Bits Register 1	MIIGSK1_ERIFBR	0x2818
MIIGSK Interrupt Event Register 1	MIIGSK1_IEVENT	0x281C
MIIGSK Interrupt Mask Register 1	MIIGSK1_IMASK	0x2820
MIIGSK Configuration Register 2	MIIGSK2_CFGR	0x2A00
MIIGSK Enable Register 2	MIIGSK2_ENR	0x2A08
MIIGSK SMII SYNC Direction Register 2	MIIGSK2_SMII_SYNCDIR	0x2A0C
MIIGSK SMII Transmit Inter Frame Bits Register 2	MIIGSK2_TIFBR	0x2A10
MIIGSK SMII Receive Inter Frame Bits Register 2	MIIGSK2_RIFBR	0x2A14
MIIGSK SMII Expected Receive Inter Frame Bits Register 2	MIIGSK2_ERIFBR	0x2A18
MIIGSK Interrupt Event Register 2	MIIGSK2_IEVENT	0x2A1C
MIIGSK Interrupt Mask Register 2	MIIGSK2_IMASK	0x2A20
UPC General Configuration Register	UPGCR	0x2E00
UPC Last PHY Address	UPLPA	0x2E04
UPC HEC Register	UPHEC	0x2E08
UPC UCC Configuration Register	UPUC	0x2E0C
UPC Device 1 Configuration Register	UPDC1	0x2E10
UPC Device 1 Transmit Rate Select High	UPDRS1H	0x2E30
UPC Device 1 Transmit Rate Select Low	UPDRS1L	0x2E34
UPC Device 1 Receive Port Priority Register	UPDRP1	0x2E50
UPC Device 1 Event	UPDE1	0x2E60
UPC Device 1 Internal Rate Configuration Register	UPRP1	0x2E70
Device 1 Transmit Internal Rate 1	UPTIRR1_1	0x2E80
Device 1 Transmit Internal Rate 2	UPTIRR1_2	0x2E82
Device 1 Transmit Internal Rate 3	UPTIRR1_3	0x2E84

Table 18-14. MSC8144E QUICC Engine Register Summary (Continued)

Register Name	Acronym	Offset
Device 1 Transmit Internal Rate 4	UPTIRR1_4	0x2E86
Device 1 Port Enable Register	UPER1	0x2EA0
SDMA Registers		
Serial DMA Status Register	SDSR	0x4000
Serial DMA Mode Register	SDMR	0x4004
Serial DMA Threshold Register	SDTR	0x4008
Serial DMA Hysteresis Register	SDHY	0x4010
Serial DMA Transfer Address Register	SDTA	0x4018
Serial DMA Transfer Channel Number Register	SDTM	0x4020
Serial DMA Address Qualify Register	SDAQR	0x4038
Serial DMA Address Qualify Mask Register	SDAQMR.	0x403C
Serial DMA Temporary Buffer Base in Multi-User RAM Value	SDEBCR	0x4044

TDM Interface 19

The MSC8144E Time-Division Multiplexing (TDM) interface enables communication among many devices over a single bus. Traffic is managed according to a time-division multiplexing method in which only one device drives the bus (transmit) for each channel. Each device drives its active transmit channels and samples its active receive channels when its channel is active. It is the system designer's responsibility to guarantee that there is no conflict in transmit channel allocation.

The TDM interface is composed of eight identical and independent TDM modules, each supporting 256 bidirectional channels (256 transmit and 256 receive channels) running at up to 62.5 Mbps with 2-, 4-, 8-, and 16-bit word size. The TDM bus connects gluelessly to most T1 /E1 framers as well as to common buses such as the ST-Bus. Each TDM module operates in independent or shared mode when receiving or transmitting data:

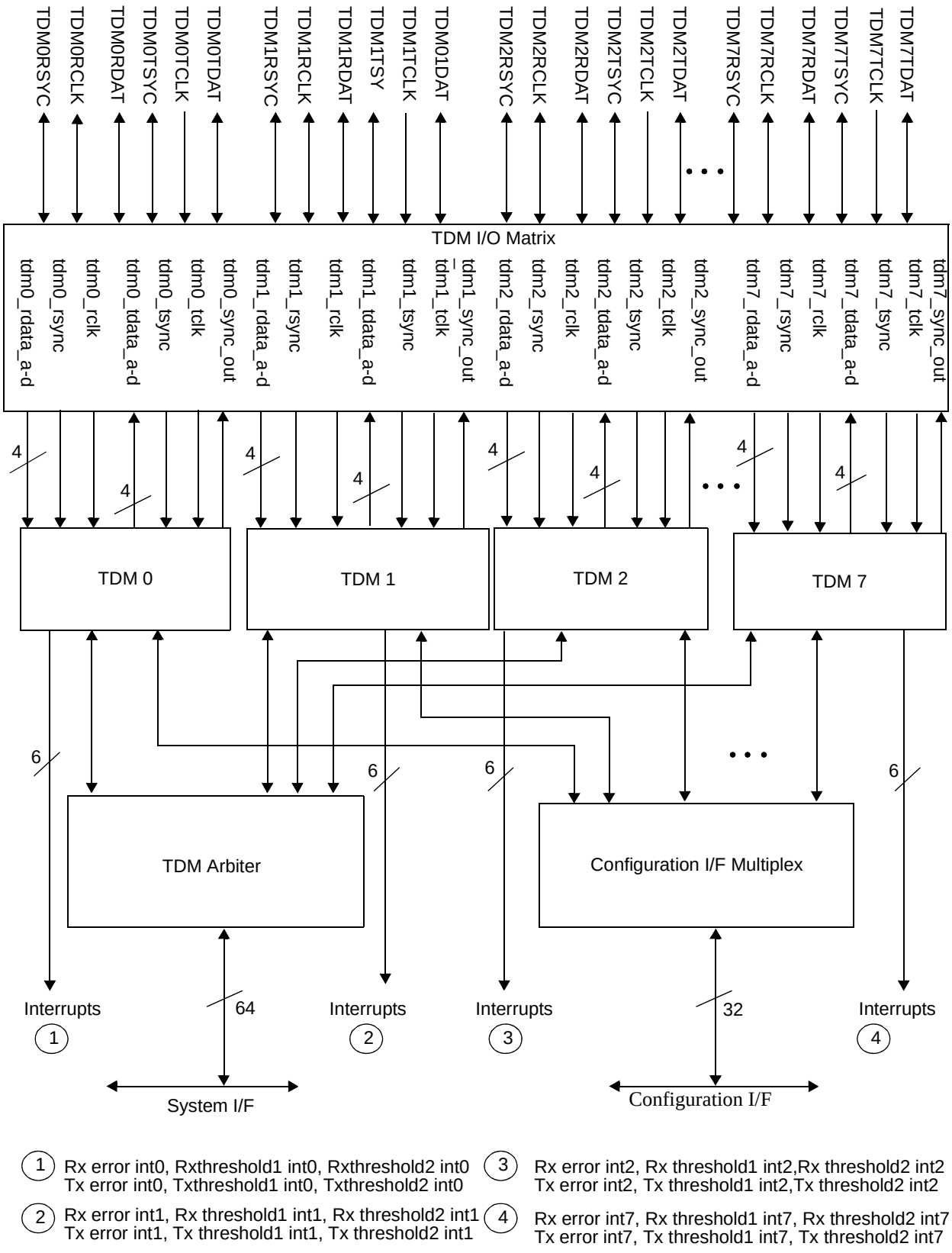
- In independent mode, there are different sync, clock, and data links for receive and transmit.
- In shared sync and clock mode, the clock and the sync are shared between the transmit and receive with different data links for the receive and transmit.
- In shared data link mode, the receive and transmit share sync, clock, and full duplex data links between the transmit and receive. The clock and the sync signals can also be shared between the TDM modules.

At any time, each channel is individually set to active or inactive. An on-the-fly hardware A-law/µ–law conversion is supported for 8-bit channels. A channel is transparent or A-law/µ–law. Its data is collected in its own buffer location independently from other channel buffers. Memory space size is 16 MB for transparent channels and 32 MB for A-law/µ–law channels.

The direction of the bits in the channel (MSB first/LSB first) is configured globally for each TDM module. The direction of TSYN is set to input or output. The polarity of the clock (sample/drive at clock rise or fall) is independently configured for the receiver and transmitter. The polarity of TSYN/RSYN/FSYN is configured to positive or negative.

The eight TDM modules have an I/O matrix that routes the clock and sync signals between the TDM modules and the MSC8144E signal lines. The TDM may be configured by all four SC3400 cores (see **Figure 19-1**), as well as by an external host. Data is received and transmitted from the TDM modules to the channel buffers through the internal MBus. **Figure 19-2** shows the TDM block diagram and the receive and transmit data flows. The dashed line depicts the transmit data flow from the system I/F to the I/O matrix; the solid line depicts the receive data flow from the I/O matrix to the receive buffers on the system I/F.

Serial data received from the I/O matrix is packed and stored in the TDM local memory buffer. From the local memory buffer, the data is converted according to A/µ transformation (if needed) and re-packed for transaction to the system I/F. Data transmission occurs in a similar way but in reverse order. The channel data is transferred from the transmit data buffers being converted by the A/µ logic and stored in the TDM local memory buffer. Then the data is transmitted to the transmit serial block and to the I/O matrix.



1 Rx error int0, Rxthreshold1 int0, Rxthreshold2 int0
Tx error int0, Txthreshold1 int0, Txthreshold2 int0

2 Rx error int1, Rx threshold1 int1, Rx threshold2 int1
Tx error int1, Tx threshold1 int1, Tx threshold2 int1

3 Rx error int2, Rx threshold1 int2,Rx threshold2 int2
Tx error int2, Tx threshold1 int2,Tx threshold2 int2

4 Rx error int7, Rx threshold1 int7, Rx threshold2 int7
Tx error int7, Tx threshold1 int7, Tx threshold2 int7

Figure 19-1. General TDM Module Interface

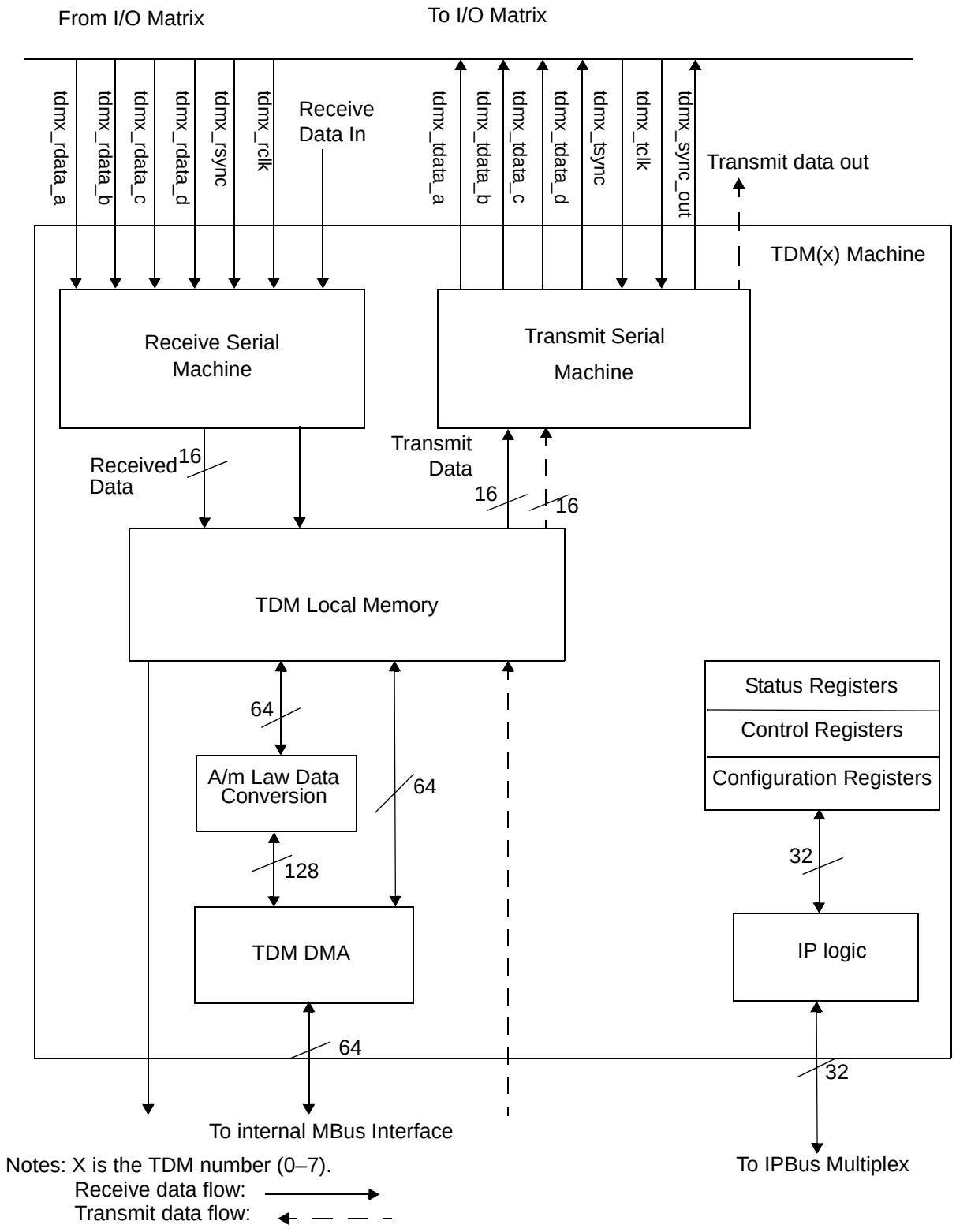


Figure 19-2. TDM Block Diagram

19.1 Typical Configurations

The TDM connects in various configurations. **Figure 19-3** shows two MSC8144E devices that connect point-to-point. Data transmits from the device on the left to the device on the right or *vice versa*.

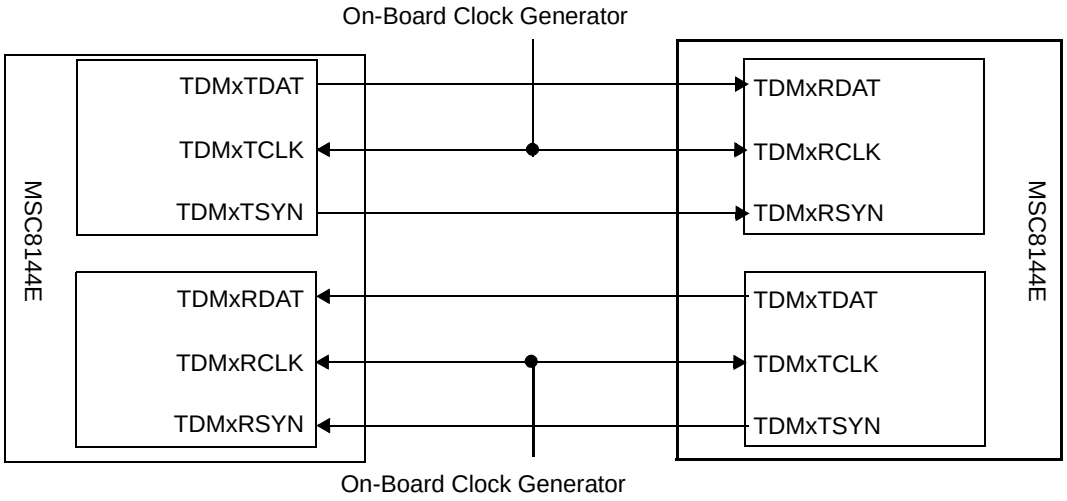


Figure 19-3. TDM Point-to-Point Configuration

Figure 19-4 depicts a TDM point to multi-point configuration. Multiple MSC8144E devices connect on the same TDM bus, which connects to the network through a framer.

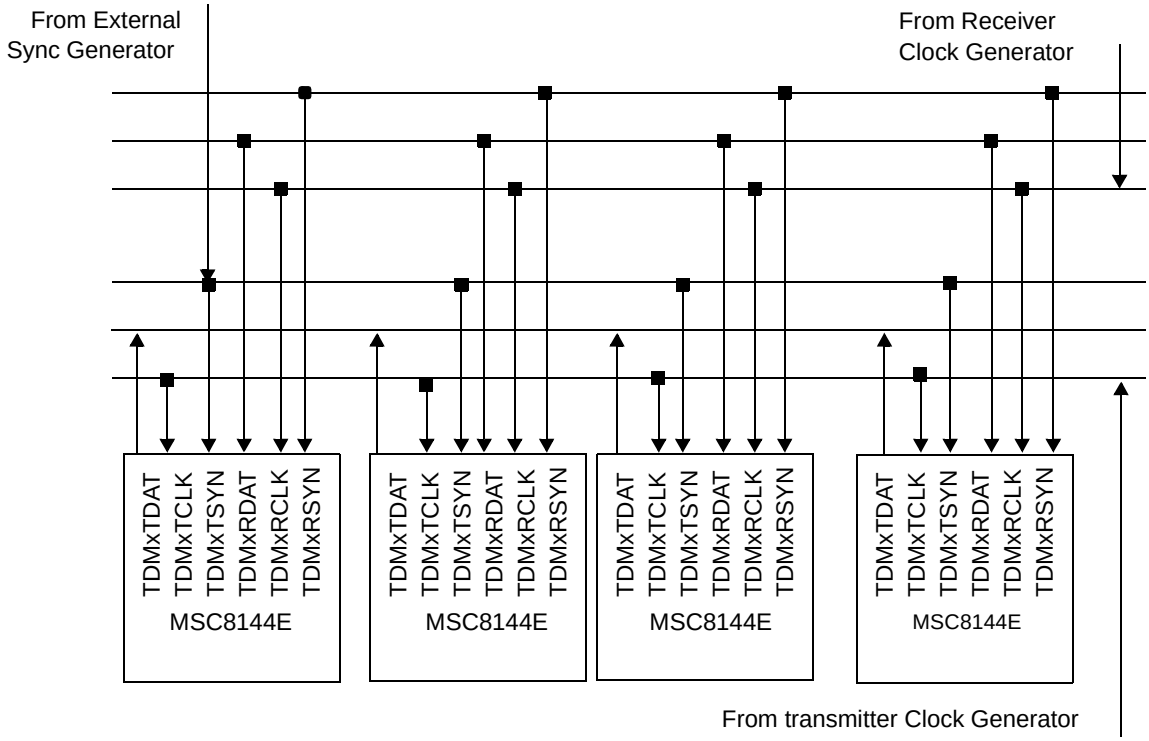


Figure 19-4. TDM Point-to-Multi-Point Configuration

Figure 19-5 depicts an application in which all the TDM modules share the sync and the clock (see **Figure 19-11**). Therefore, each TDM module supports one or two active links. In this example, 16 receive link and 16 transmit links connect to two MSC8144E devices.

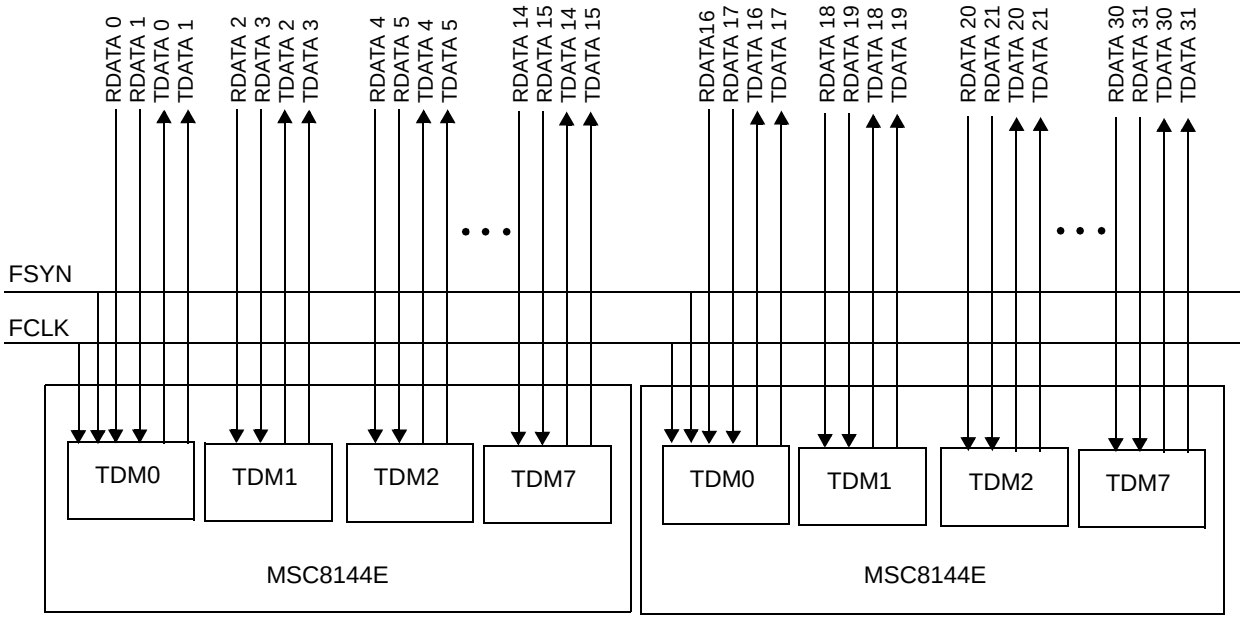


Figure 19-5. Common Frame Sync and Clock

19.2 TDM Basics

Multiple TDM channels are transferred sequentially in a structure called a frame. The frame start is identified by a frame sync signal that is briefly asserted at the beginning of every frame. Each of the eight TDM modules can receive or transmit up to 256 channels at a granularity of two. The number of receive channels is determined by the RNCF field in the TDMx Receive Frame Parameters Register (TDMxRFP) (seepage 19-48). The number of transmit channels is determined by the TNCF field in the TDMx Transmit Frame Parameters Register (TDMxTFP) (see page 19-51).

The size of all the channels (for each TDM module) is unified and it can be 2-, 4-, 8-, and 16 bits. The receive channel size is determined by the RCS field in the TDMxRFP; the transmit channel size is determined by the TCS field in the TDMxTFP (refer to page 19-51).

When the TDM connects to a T1 framer, the RT1 field in the TDMx Receive Frame Parameters Register (TDMxRFP) (see page 19-48) and the TT1 field in the TDMx Transmit Frame Parameters Register (TDMxTFP) (see page 19-51) should be set. The T1 frame contains 193 bits (24 channels of 8 bits each) when the first bit of the frame is a Frame Alignment bit that is not used by the TDM. At the T1 received frame, the Frame Alignment bit is removed and does not transfer to the main memory. At the transmit T1 frame, the bit is not driven out.

Figure 19-6 shows an example of TDMx frames. The receive frame contains two 2-bit channels. The transmit frame contains four 4-bit channels. **Figure 19-7** shows an example of T1 frame. In T1 mode, the first bit of the frame is not used by the TDM.

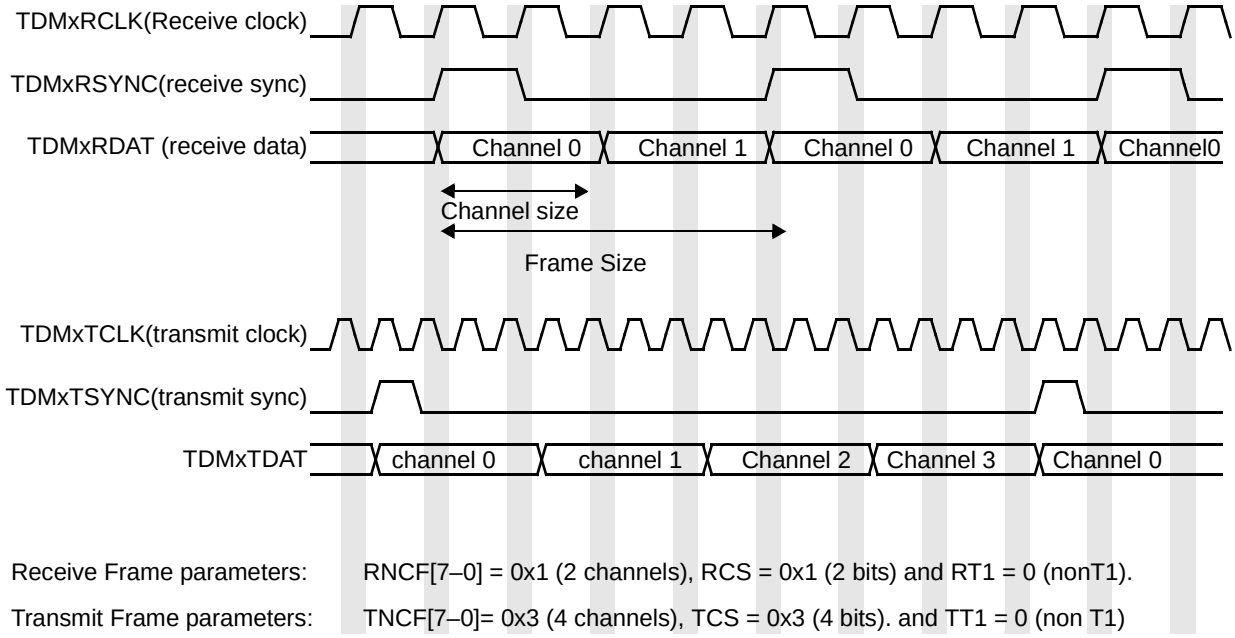


Figure 19-6. TDM Frames

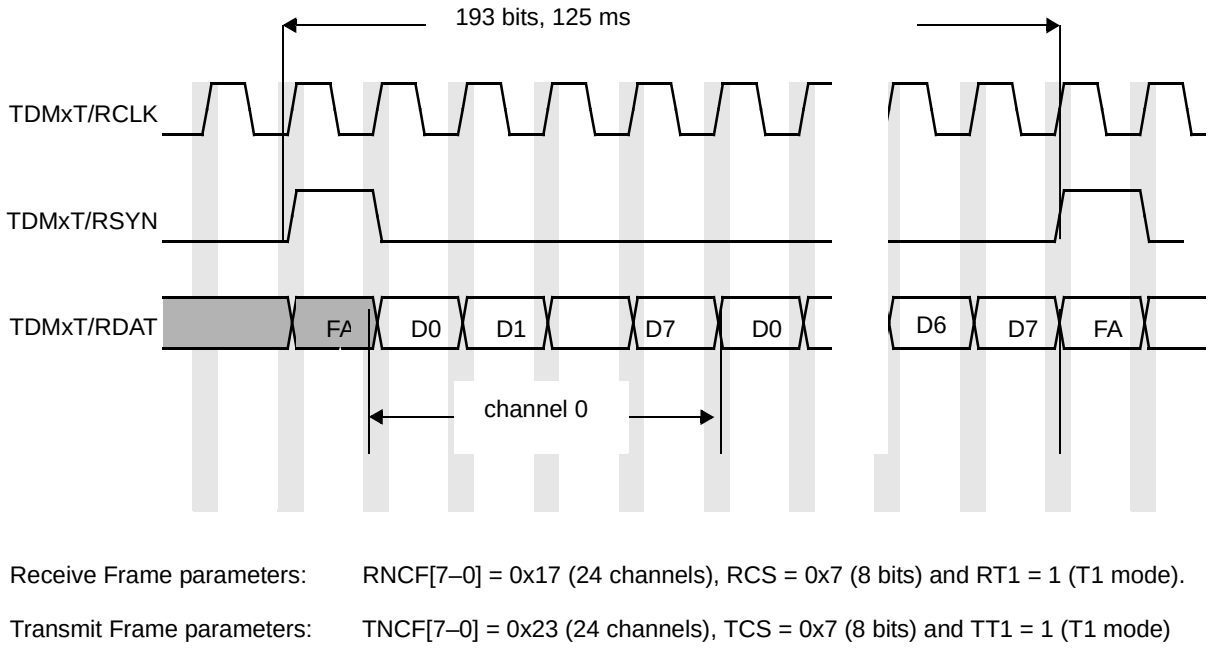


Figure 19-7. T1 Frame

19.2.1 Common Signals for the TDM Modules

The sync and clock signals can be shared among the TDM modules or separate for each TDM module. When the CTS bit of the TDMx General Interface Register (see page 19-36) is equal to 1, the TDM modules share sync and clock signals. In this mode, the common signals connect to the following signal lines:

- In non-independent mode, connect the shared sync to TDM0TSYN (receive and transmit of all TDM modules share the same sync signal).
- In non-independent mode, connect the shared clock to TDM0TCLK (receive and transmit of all TDM modules share the same clock signal).
- In independent mode, connect the transmit shared sync to TDM0TSYNC (transmit of all TDM modules share the same sync signal). Connect the receive shared sync to TDM1TSYNC (receive of all TDM modules share the same sync signal)
- In independent mode, connect the transmit shared clock to TDM0TCLK (transmit of all TDM modules share the same clock signal). Connect the receive shared clock to TDM1TCLK (receive of all TDM modules share the same clock signal).
- When the TDMxTIR[TSO] bit is set to a value of 1 (see page 19-46), the sync out signal drives out through TDM0TSYN.

The configuration registers (see page 19-36) should be identical for the TDM modules that share signals. There are only seven possibilities for sharing TDMs:

- TDM0 and TDM1.
- TDM0, TDM1, and TDM2.
- TDM0, TDM1, TDM2 and TDM3.
- TDM0, TDM1, TDM2, TDM3 and TDM4.
- TDM0, TDM1, TDM2, TDM3, TDM4 and TDM5.
- TDM0, TDM1, TDM2, TDM3, TDM4, TDM5 and TDM6.
- TDM0, TDM1, TDM2, TDM3, TDM4, TDM5, TDM6 and TDM7.

Figure 19-8 illustrates a common receive sync, receive clock, transmit sync, and transmit clock for TDM0 and TDM1. When the CTS bit of the TDMx General Interface Register (see page 19-36) is cleared, the TDM modules do not share signals. In **Figure 19-8**, TDM2 - TDM7 do not share signals with the other TDM modules

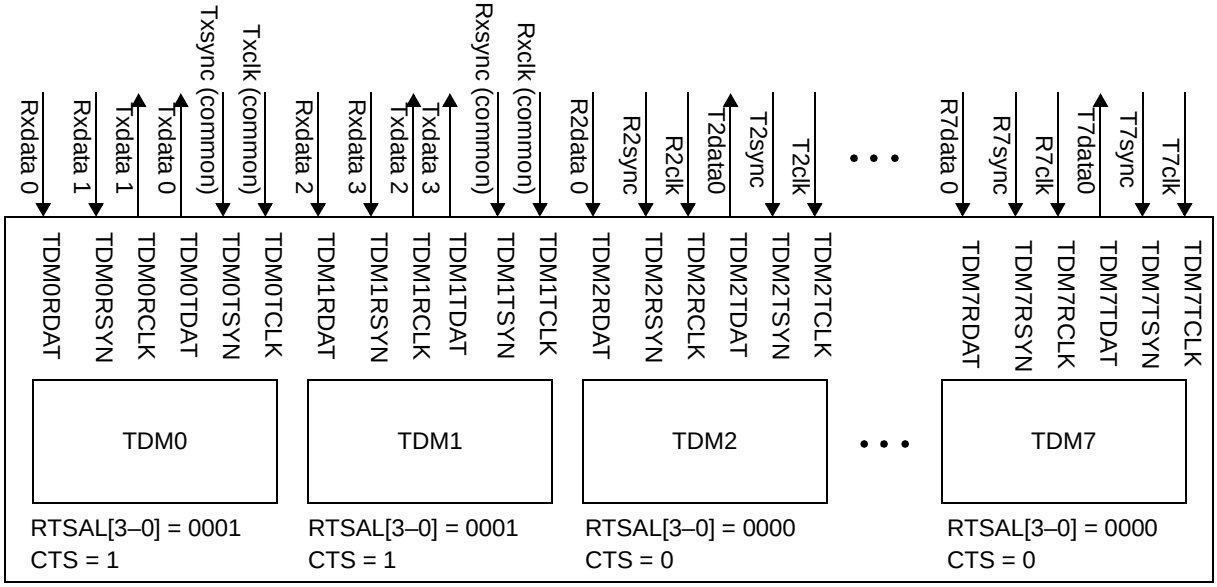


Figure 19-8. TDM Modules Model

In **Figure 19-9**, all eight TDM modules share the same receive clock and sync and the same transmit clock and sync. The receive clock connects to the TDM1TCLK port. The transmit clock connects to TDM0TCLK, the receive sync connects to TDM1TSYN, and the transmit sync connects to TDM0TSYNC. Each module has two active data links. Notice that TDMxTSYN, TDMxTCLK when $2 \le x \le 7$ are not used.

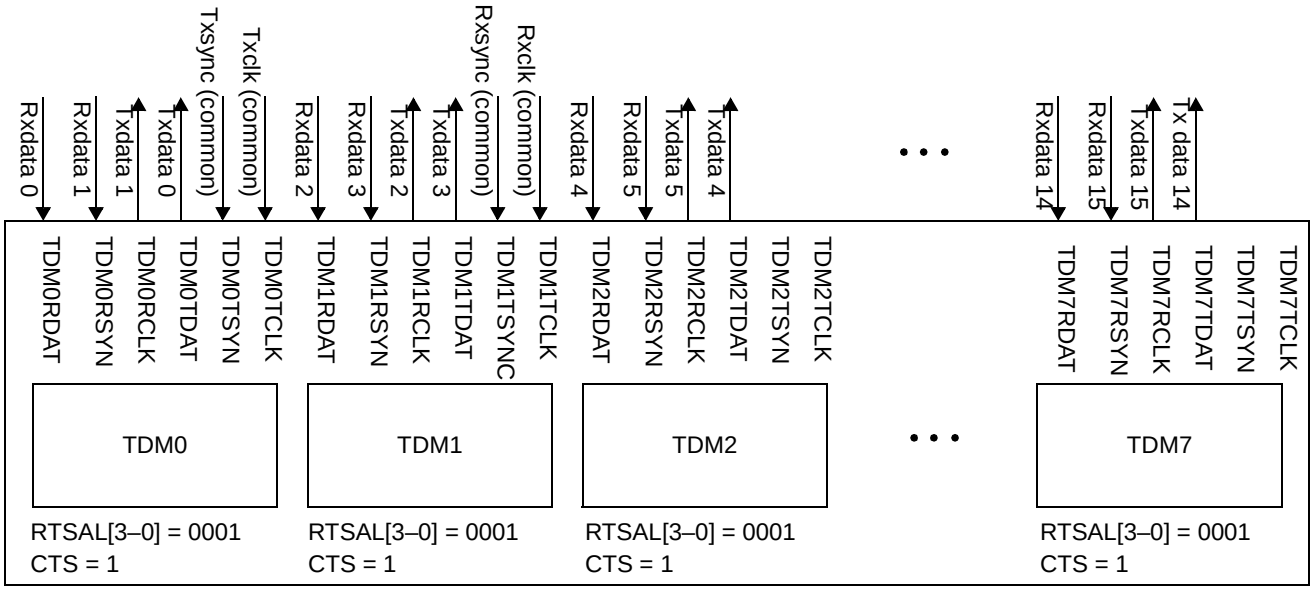


Figure 19-9. Shared Receive Sync and Clock and Transmit Sync and Clock

In **Figure 19-10**, all eight TDM modules share the same frame sync, clock, and data links. Notice that TDMxTCLK and TDMxTSYN when $1 \leq x \leq 7$ are not used.

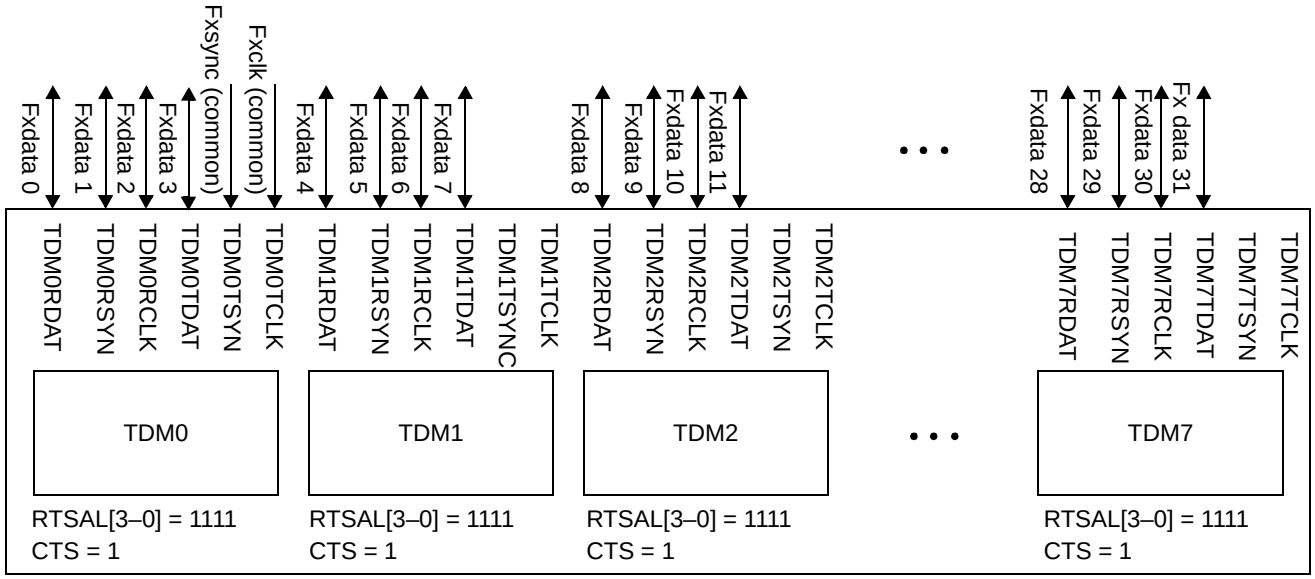


Figure 19-10. Shared Frame Sync, Clock, and Data Links

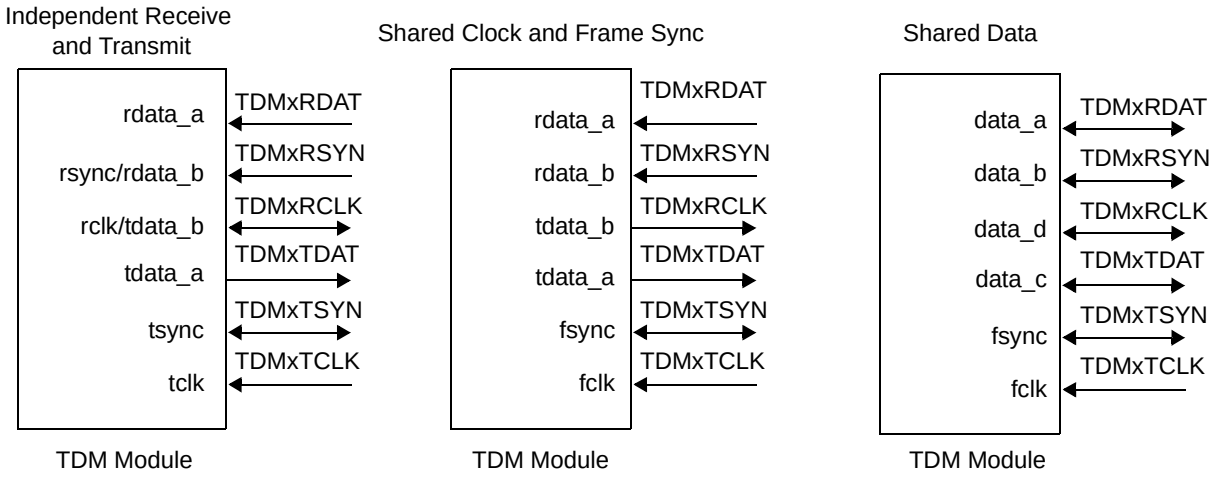
19.2.2 Receiver and Transmitter Independent or Shared Operation

The TDM operates with the transmit and receive operations running either independently or shared, as illustrated in **Figure 19-11**. When the two most significant bits of the RTSAL field (RTSAL[3–2]) in the TDMx General Interface Register (see page 19-36) equal 0b00, the receive and the transmit are independent as illustrated on the left side of **Figure 19-11**. In this mode, there is one input receive data link and one output transmit data link. If the TDM shares signals with other TDM modules (CTS = 1), it can receive two data links and it can output two data links.

When the RTSAL[3–2] in the TDMx General Interface Register (see page 19-36) equal 0b01, the receive and transmit are shared as illustrated in the middle of **Figure 19-11**. The transmit and the receive share the Frame Sync (FSYN) and the Frame Clock (FCLK) signals. The number of receive and the transmit active links can be one or two. The direction of the receive links is input, and the direction of the transmit links is output.

When RTSAL[3–2] in the TDMx General Interface Register (see page 19-36) equal 0b11, the receive and the transmit are shared as illustrated on the right side of **Figure 19-11**. The transmit and the receive share the Frame Sync (FSYN), the Frame Clock (FCLK), and the data signals. In this mode, the data links are full duplex and are used for both transmit and receive, so the number of active links can be 1, 2, or 4.

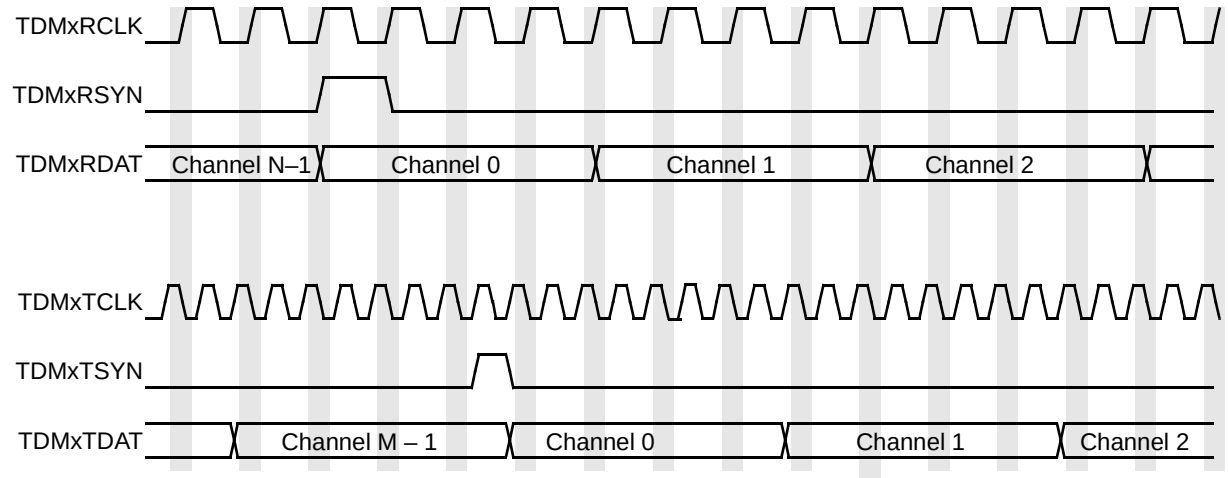
When RTSAL [1–0] equals 0b11, there are four active links: DATA_A, DATA_B, DATA_C, and DATA_D. When RTSAL[1–0] equals 0b01, there are two active links: DATA_A and DATA_B. When RTSAL[1–0] equals 0b00, there is one active link, DATA_A.



x Defines the TDM number.
FSYNC (frame sync) specifies that the receiver and transmitter share the same sync.
FCLK (frame clock) specifies that the receiver and transmitter share the same clock.

Figure 19-11. TDM Module Sharing Modes

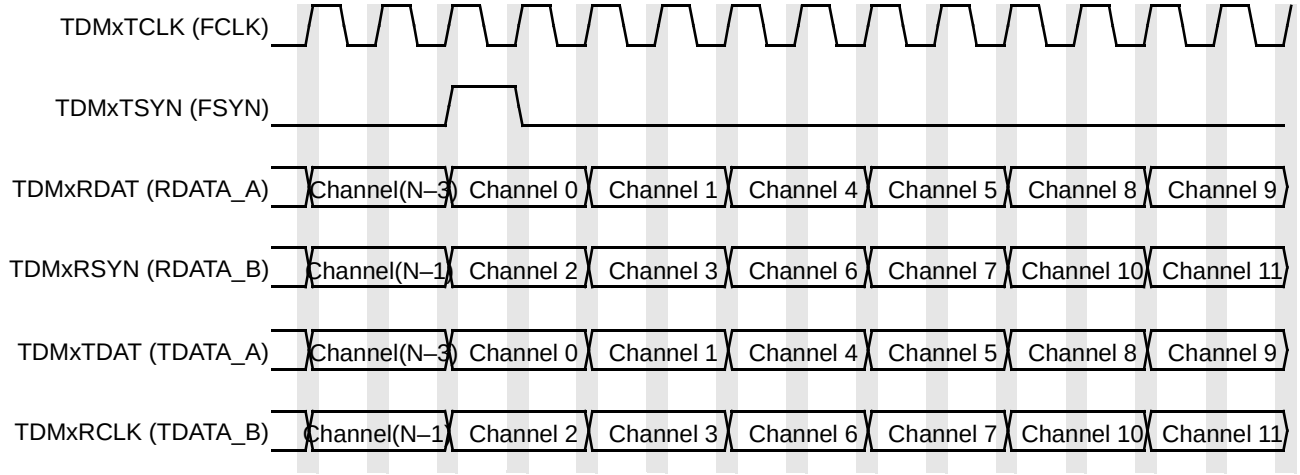
Figure 19-12 describes the TDM interface when the receive and transmit are totally independent (TDMxGIR[RTSAL] = 0b0000). The TDMxRCLK is not synchronized to the TDMxTCLK. They differ according to the sync location relative to the beginning of the frame and the number of bits.



X The TDM number.
N The number of channel in the receive TDM frame.
M The number of channels in the transmit TDM frame.

Figure 19-12. Receive and Transmit Totally Independent

Figure 19-13 describes the TDM timing interface when TDMxGIR[RTSAL] = 0b0101. The frame sync and the clock is shared between the receive and transmit, and links A and links B are active. RDAT and RSYN are used as received data links, and TDAT and RCLK are used as transmit data links. Channels are organized in pairs: channel i and channel i+1 are always received/transmitted on the same link, one after another.

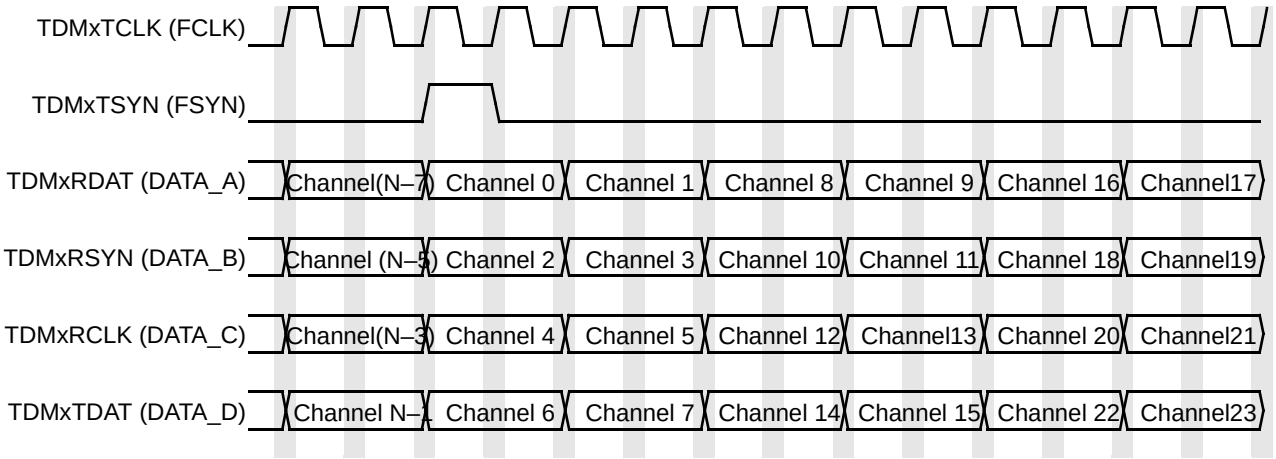


x The TDM number.

N The number of channels in a TDM frame.

Figure 19-13. Shared Sync and Clock (Two Active Data Links)

Figure 19-14 shows the TDM timing interface when the RTSAL field is set to 0b1111.The frame sync, the clock, and the data links are common. All four data links are active and are used for both transmit and receive.



x The TDM number.

N Number of channels in a TDM frame.

The data links are bidirectional.

The clock and the sync are common.

Figure 19-14. Receive and Transmit Share Sync, Clock, and Data (Four Active Links)

Note: The number of channels in a frame must be a multiple of 2 × the number of data links.

Table 19-1 shows the number of channels each active link supports.

Table 19-1. Maximum Number of Channels Per Active Link

Number of Active Links	1 Active Link	2 Active Links	4 Active Links
Maximum channel number per active link in one TDM module	256	128	64

The TDM bit rate depends on:

- *System bus clock*. The TDM processes the data in CLASS64 clock rate, so the maximum data bit rate is limited to one half of the rate of the CLASS64 clock.
- *Number of active links*. The total bit rate is shared by all active links, so one active link supports the highest bit rate.
- *Channel width*. When there are more bits per channel, there are fewer channels per second, so higher bit rates can be processed per second.

Table 19-2 describes the maximum bit rate as a function of these parameters. Factors other than the width of the channel can affect the bit rate, for example, the memory system load.

Table 19-2. Factors Affecting Maximum Bit Rate

Channel Width (Bits)	1 Active Link	2 Active Links	4 Active Links
2	CLASS64 clock/8	CLASS64 clock/12	CLASS64 clock/20
4	CLASS64 clock/4	CLASS64 clock/6	CLASS64 clock/10
8	CLASS64 clock/2	CLASS64 clock/3	CLASS64 clock/5
16	CLASS64 clock/2	CLASS64 clock/2	CLASS64 clock/2.5

Note: In addition to the limits defined in **Table 19-2**, the maximum serial frequency is limited to 62.5 MHz.

19.2.3 TDM Data Structures

TDM data structures are stored in transmit and receive local memory, as follows:

- *TDM receive local memory*. Received data is stored in 256 8-byte entries located in addresses between 0x0000–0x07FF, which is offset from the TDMx receive local memory (see **Chapter 9**, *Memory Map*). This memory contains 1, 2, 4, 8, 16, or 32 indexed buffers starting at 0. Each buffer contains multiple frames. The number of buffers used to store the received data is indicated in the RNB field of the TDMx Receive Number of Buffers Register (TDMxRNB) (discussed on page 19-68). Channel C in buffer B is the 8 bytes starting at (256 / (RNB + 1) × B + C) × 8.

- *TDM transmit local memory*. Transmit data is located in the TDM local memory before it is transmitted externally. The data is stored in 256 8-byte entries in addresses between 0x1800– 0x1FFF, which is offset from the TDMx receive local memory (see **Chapter 9**, *Memory Map*). This memory can contain 1, 2, 4, 8, 16, or 32 indexed buffers starting at 0. Each buffer contains multiple frames. The number of buffers used to store the transmitted data is indicated in the TNB field of the TDMx Transmitter Number of Buffers Register (TDMxTNB). Channel C in buffer B is the 8 bytes starting at (256 / (TNB + 1) × B + C) × 8.

Figure 19-15 shows an example of TDM local memory that contains four transmit buffers and one receive buffer. Up to 32 transmit bytes of channel 2 are located in four buffers (TNB = 3). Only 8 receive bytes of channel 2 are located in one buffer (RNB = 0). Each buffer contains 8 bytes per channel.

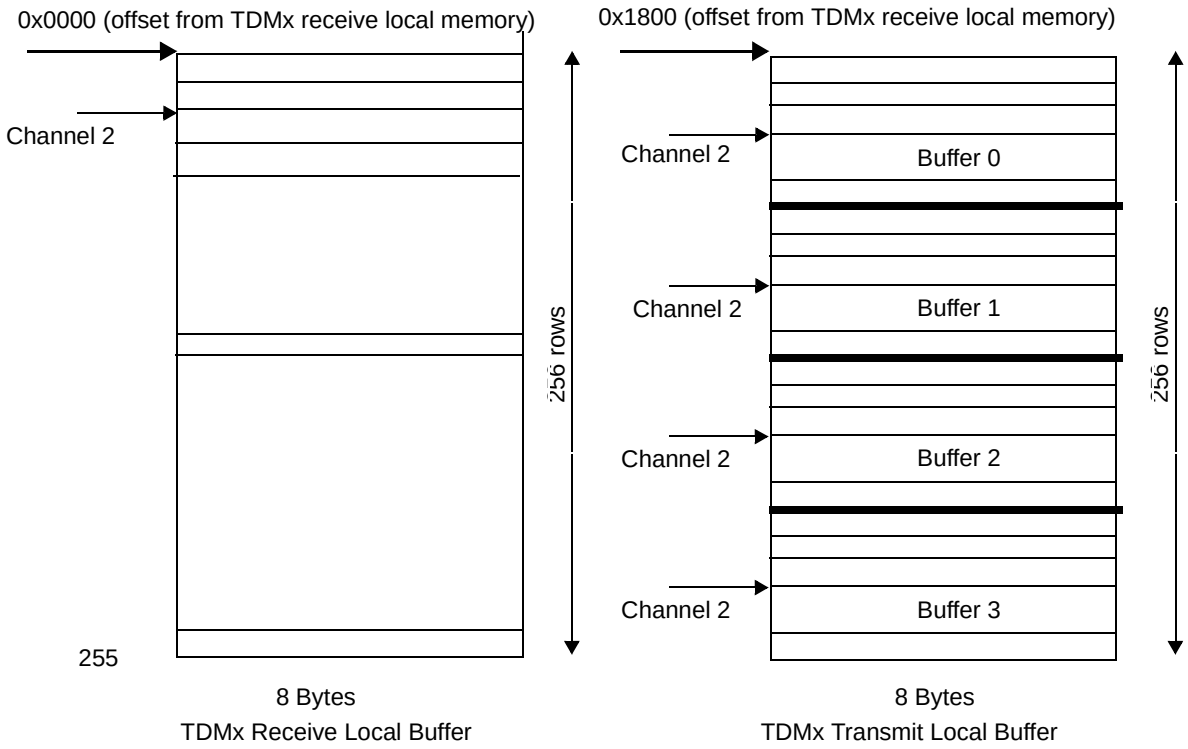


Figure 19-15. TDM Local Buffer (Receive and Transmit)

When the TDM transmit local memory is accessed using addresses with 8-byte alignment, data is written to the 4 LSB of the memory row. **Figure 19-16** describes the TDMx local memory after write access of 0x01234567 to address 0x1800 (offset from TDMx Receive Local Memory) and 0x89ABCDEF to address 0x1804 (offset from TDMx Receive Local Memory). If TDMxTIR[TBOR] = 1, the 0x89ABCDEF data is transmitted before the 0x01234567 data.



Figure 19-16. TDMx Local Memory Write Example

When the TDM receive local memory is accessed using addresses with 8-byte alignment, data is read from the 4 LSB of the memory row. **Figure 19-17** describes a row in the TDMx local memory, in which the 0x00112233 data is received before the 0x44556677 data (if TDMxRIR[RBOR] = 1). In this example, the data to be read from address 0x1000 (offset from TDMx Receive Local Memory) is 0x44556677, and the data to be read from address 0x1004 (offset from TDMx Receive Local Memory) is 0x00112233.

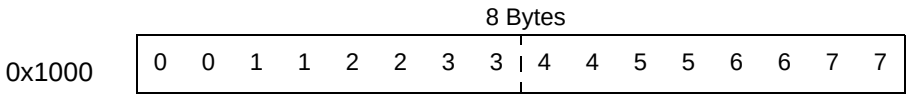


Figure 19-17. TDMx Local Memory Read Example

19.2.4 Serial Interface

This section covers issues related to the serial interface, such as how to configure the frame sync and how to control the data order of the bits in the channel.

19.2.4.1 Sync Out Configuration

TDMxTSYN is programmed as either an input or output by writing 1 to the TSO bit in the Transmit Interface Register (TDMxTIR) (see page 19-46). When the TSO bit value is equal to 1, the sync_out signal connects to the sync out signal in the TDM I/O matrix and is output via TDMxTSYN. When the TDM modules share a sync and clock signals (the CTS bit is set), then the TDMx[TSO] bits should be equal for all the TDM modules and they determine whether the sync arrives from the board or is generated by the TDM0 transmitter. Configuring the sync out signal involves the parameters listed in **Table 19-3**.

Table 19-3. Parameters in Configuring the Frame Sync (TDMxTIR[TSO] = 1)

Task	Register
Control the length of the sync_out signal. If the SOL bit is clear then the sync_out width is one transmit bit, else the sync_out length is one transmit channel.	TDMxTIR[SOL], page 19-46
Control the transmit clock edge on which the sync_out is driven out. If the SOE bit is clear, the sync_out is driven out on the rising edge of the transmit clock.	TDMxTIR[SOE], page 19-46
Control the sync_out level. The sync out level must be identical to the transmit sync. It is determined by the TSL configuration field.	TDMxTIR[TSL], page 19-46
Control the sync_out distance. The distance between two consecutive sync out events is constant and equal to one transmit frame. The transmit frame length is determined by the transmitter configuration fields TCS,TNCF, TT1 and RTSAL[1–0]. The distance is = (TCS + 1) × (TNCF + 1) / (RTSAL[1–0] + 1) + TT1.	TDMxTFP[TNCF], page 19-51 TDMxTFP[TCS], page 19-51 TDMxTFP[TT1], page 19-51 TDMxGIR[RTSAL], page 19-36

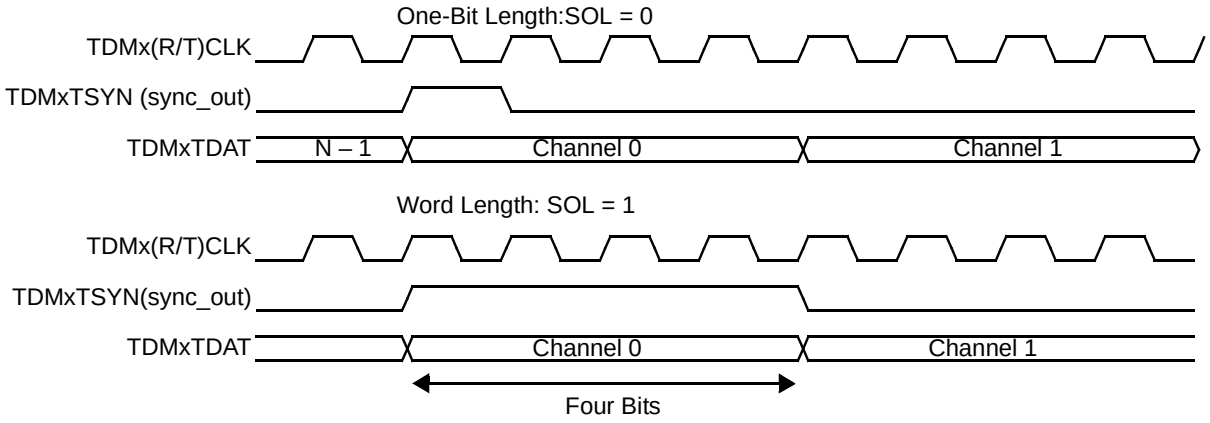


Figure 19-18. Sync Length Selection

19.2.4.2 Sync In Configuration

TDMxRSYN is an input that identifies the beginning of the received frame. TDMxTSYN can be an input or output from the TDM, but the transmitter refers to the transmit sync as an input because the connection between the sync_out signal and the transmit sync (tsync) occurs only in the TDM I/O matrix. **Figure 19-19** illustrates the relation between the data, the sync, and the clock for various configurations. The receive data and frame sync are sampled with the rising or falling edge of the receive clock. The transmit frame sync is sampled with the rising or falling edge of the transmit clock. The transmit data drives out at the rising or falling edge of the transmit clock. The delay between the first data bit of the frame and the sync is referred to as the rising edge of the sync. **Table 19-4** lists the frame sync controls.

Table 19-4. Transmit and Receive Frame Configuration

Control	Register
Which receive clock edge samples the receive frame sync. If RFSE is clear, the receive frame sync is sampled on the rising edge of the receive clock.	TDMxRIR[RFSE] bit, page 19-44.
Which transmit clock edge samples the transmit frame sync. If TFSE is clear, the transmit frame sync is sampled on the rising edge of the transmit clock.	TDMxTIR[TFSE] bit, page 19-46
Which receive clock samples the receive data. If RDE is clear, the receive data is sampled on the rising edge of the receive clock.	TDMxRIR[RDE] bit, page 19-44
Which transmit clock edge drives out the data. If TDE is clear, then the transmit data is driven out on the rising edge of the transmit clock.	TDMxTIR[TDE] bit, page 19-46
Determines the receive sync level. If RSL is clear the receive sync level is high.	TDMxRIR[RSL] bit, page 19-44
Determines the transmit sync level. If TSL is clear the transmit sync level is high.	TDMxTIR[TSL] bit, page 19-46
Determines the timing of the receive frame sync signal relative to the first data bit of the receive frame.	TDMxRIR[RFSD] field, page 19-44
Determines the timing of the transmit frame sync signal relative to the first data bit of the transmit frame.	TDMxTIR[TFSD] field, page 19-46

The receive delay when the receive sync and the receive data are not sampled at the same clock edge is **RFSD** + 0.5. The transmit data can be driven out before the transmit sync sample. Therefore, the transmit delay when the transmit sync and transmit data are sampled/driven out at the same clock edge is (**TFSD** – 1). And when the sync and the data sampled/driven out at different clock edge is (**TFSD** – 1 + 0.5).

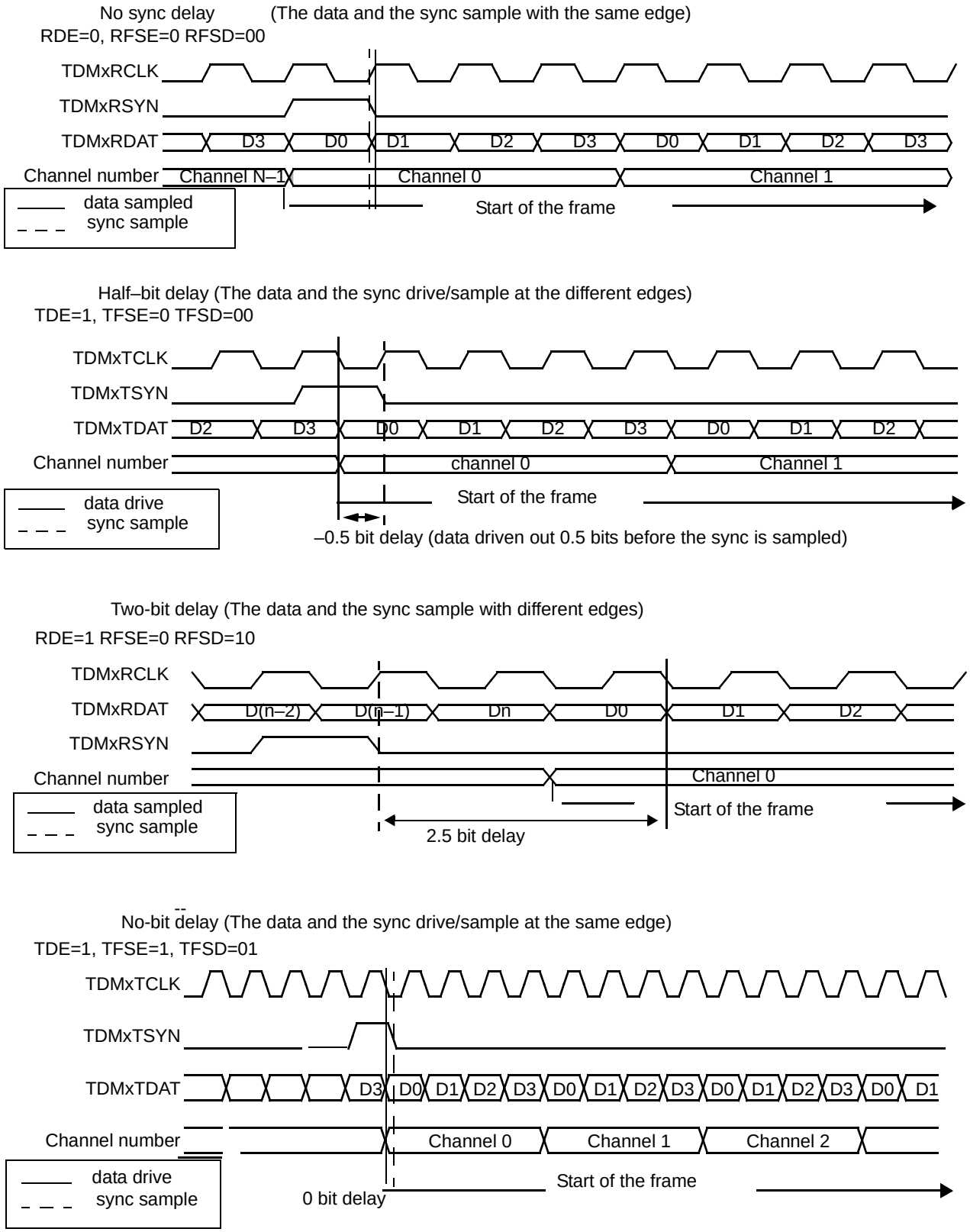


Figure 19-19. Frame Sync Configurations

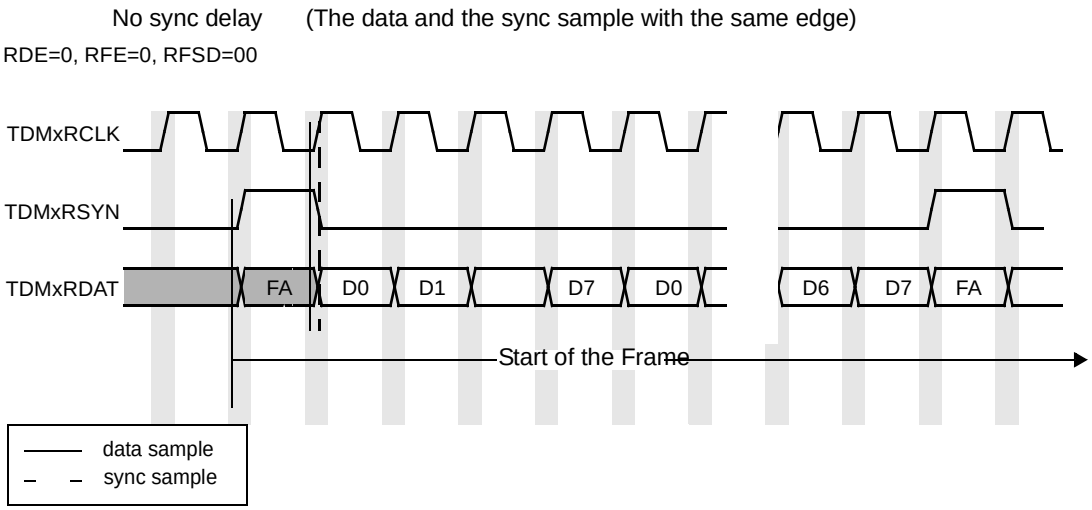


Figure 19-20. Frame Sync Configuration At T1 Mode

Figure 19-21 illustrates how the polarity of the receive sync and the transmit sync signals is controlled by the TDMxRIR[RSL] and the TDMxTIR[TSL] bits.

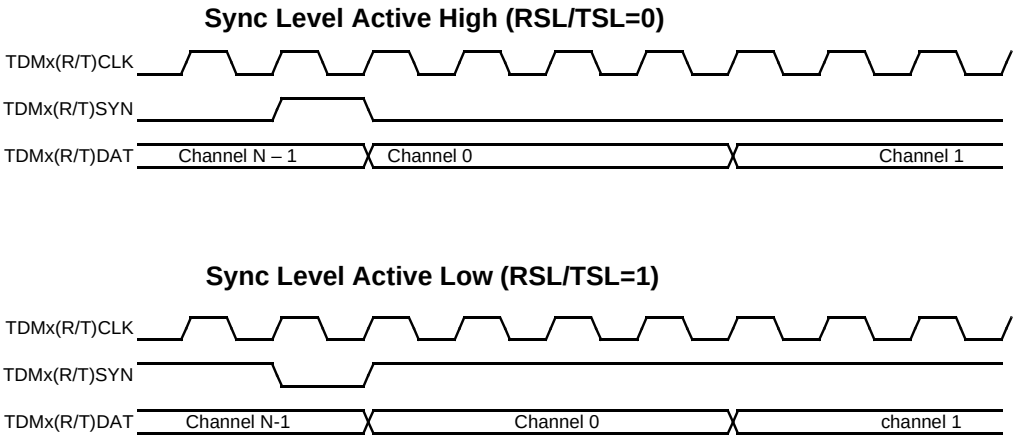


Figure 19-21. Frame Sync Polarity

19.2.4.3 Serial Interface Synchronization

The TDM module enables communication among many devices over a single bus. The receive and transmit of each TDM frame is identified by a frame sync signal that is asserted at the beginning of every frame. The frame sync synchronization is necessary when more than one device drives the bus. **Figure 19-22** shows the state diagram of the frame sync synchronization.

The details of the state diagram are as follows:

- *HUNT* (0b00). A sync event is constantly sought. As soon the sync event is detected, the state machine changes to a WAIT state. During the Hunt state, data is neither received nor transmitted.
- *WAIT* (0b01). At least one sync has been detected. The next sync event is accepted after one TDM frame. If the sync appears in the correct position, the state changes to the PRESYNC state (0b11). If the sync does not appear, the state returns to the hunt state. During the WAIT state, data is neither received nor transmitted.
- *PRESYNC* (0b11). Two sync events have been detected and the distance between the syncs is one TDM frame. If the sync event is recognized early, the state returns to the WAIT state. Otherwise, the machine transfers to the SYNC state at the last bit of the TDM frame. During PRESYNC state, data is neither received nor transmitted.
- *SYNC* (0b10). At least one sync event has appeared exactly where it was expected. This state is maintained as long as the sync event continues to appear where expected. If a sync is missed or a sync event is recognized early, the state changes to the HUNT state (0b00). During the SYNC state, data is both received and transferred.

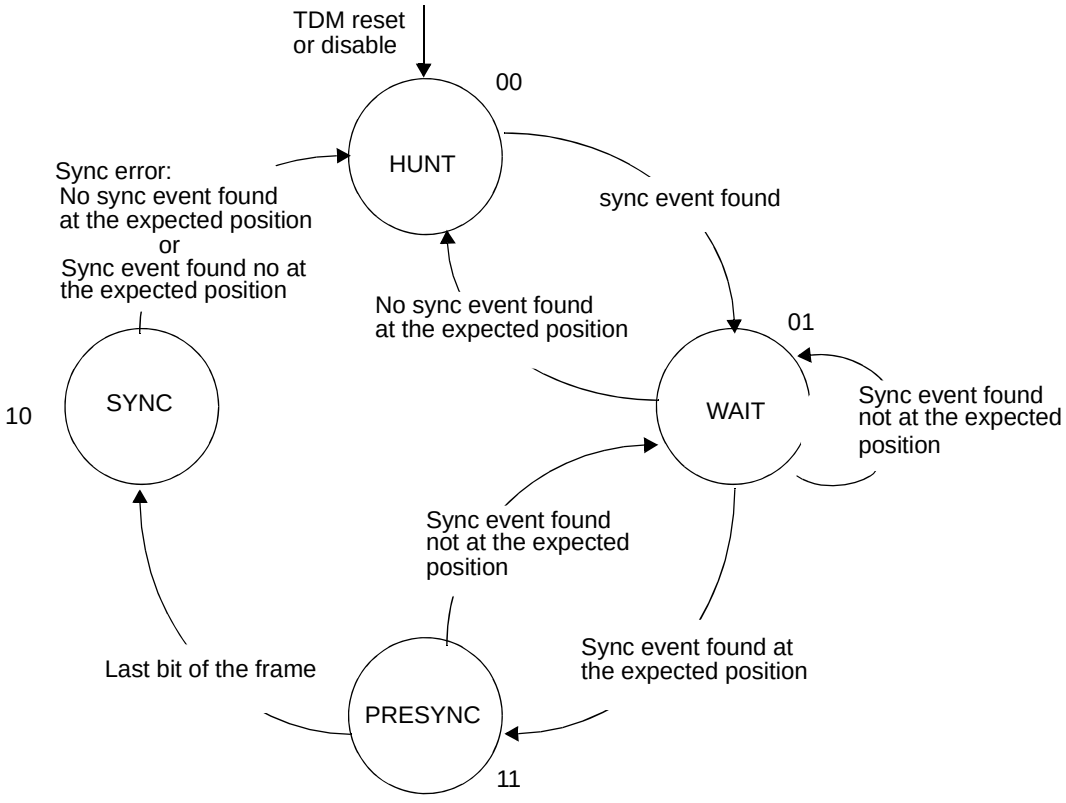


Figure 19-22. Frame Sync Synchronization State Diagram

The TDM receiver synchronizes on the receive frame sync (rsync). The state of the receive frame sync synchronization is indicated by the TDMxRSR[RSSS] field (see page 19-72). During the HUNT, WAIT, and PRESYNC states, the received data is not transferred to the buffers in the main memory for processing. When the receive sync synchronization is lost, the state transfers from SYNC to HUNT (the TDMxRER[RSE] bit is asserted) (see page 19-69). If the TDMxRIER[RSEEE] bit (see page 19-64) is also set, a receive error interrupt is generated. The interrupt service routine (ISR) should clear the TDMxRER[RSE] bit by writing a 1 to the bit before clearing the related status bit in the EPIC and before returning from the ISR.

The transmit frame sync synchronization state is indicated by the TDMxTSR[TSSS] field (see page 19-73). During the HUNT, WAIT, and PRESYNC states, new data is not driven out. If the Transmit Always Out (TDMxTIR[TAO]) field (see page 19-46) is set, then the last data is driven out until the frame sync synchronization state returns to SYNC state. If the TDMxTIR[TAO] bit is clear, data is not driven out and TDMxTDAT is tri-stated. When the transmit sync synchronization is lost, the TDMxTER[TSE] bit (see page 19-70) is asserted. If the TDMxTIER[TSEIE] bit (see page 19-65) is also set, a transmit error interrupt is generated. The ISR should clear the TDMxTER[TSE] bit by writing a 1 to the bit before clearing the related status bit in the EPIC and before returning from the ISR.

The frame sync synchronization state can identify different problems. In the initial design stages, the frame sync summarization state indicates whether the TDM programming matches the actual TDM stream. During operation, the synchronization state and the error interrupts may indicate errors in the TDM module signal processing.

19.2.4.4 Reverse Data Order

Figure 19-23 illustrates how the bit order of the stored data relates to the bit order of the receive or the transmit data. The TDMxRIR[RRDO] bit defines how the receive channel data is stored in memory. If TDMxRIR[RRDO] is clear, the first bit of the received channel data is stored as the most significant bit. The TDMxTIR[TRDO] bit selects the transmit data bits order. If TDMxTIR[TRDO] is clear, the most significant bit of the memory is transmitted as the first transmit data.

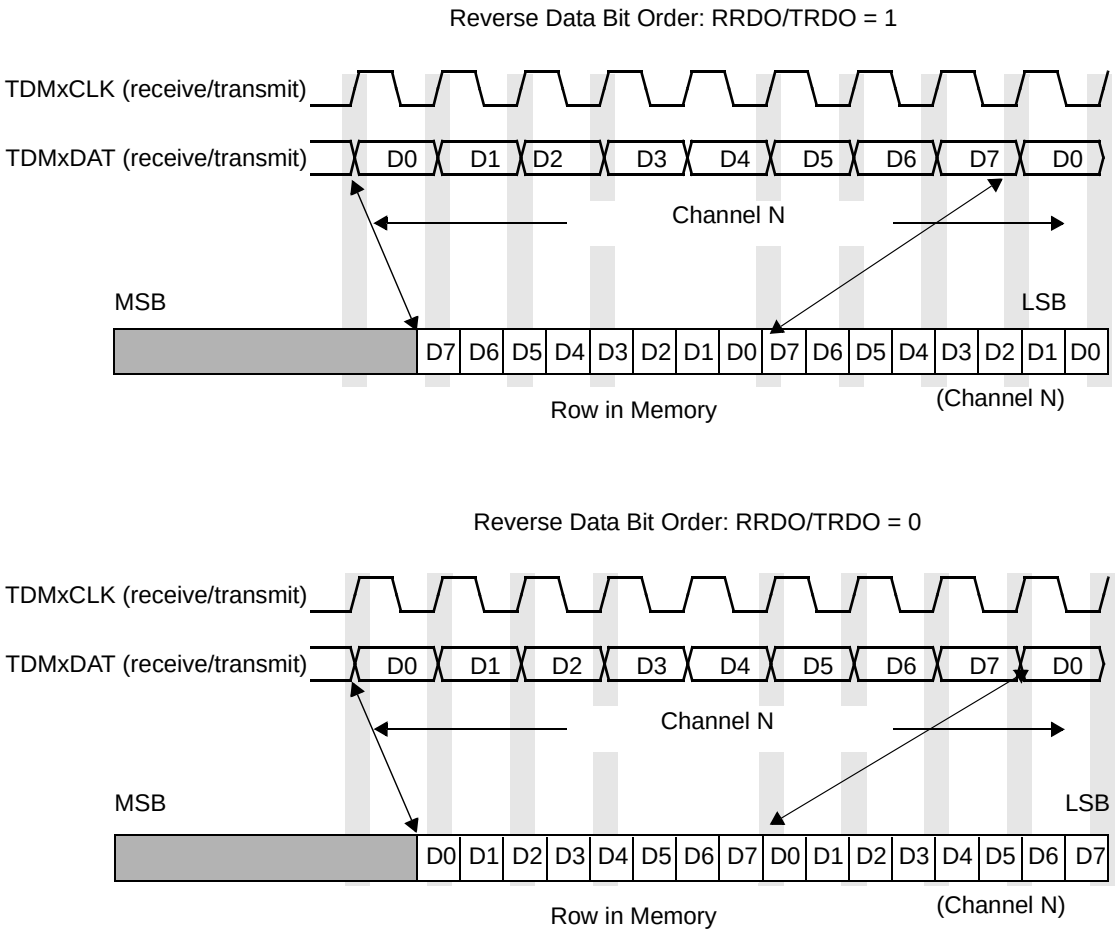


Figure 19-23. Reserve Bit Order

19.2.5 TDM Local Memory

Received data is temporarily stored in the TDM receive local memory until it is transferred to the receive buffers mapped on the system I/F. A single data transfer from TDM local memory to a memory-mapped region on the system I/F transfers at least 64 bits of data. Each channel can store more than 64 bits before the data is written to the buffers mapped on the system I/F. The TDMxRFP[RCDBL] field (see page 19-48) provides an upper boundary on the number of receive bits that can be stored in the TDM local memory. The receive data latency is defined as the time between receiving data and the time when it is available for processing by an SC3400 core. Reducing the TDMxRFP[RCDBL] value reduces the receive data latency. However, reading the TDM local memory imposes more strict latency requirements on the system I/F. The maximum receive data latency is calculated as follows: RCDBL /RCS × receive frame time.

When the amount of received data exceeds the size of the TDM receive local memory, the TDMxRER[OLBE] bit is set (see page 19-69). If the TDMxRIER[OLBEE] bit is also set, an error interrupt is generated. This error should not occur during normal operation. It indicates that the TDM has not received enough bandwidth on the system I/F and therefore cannot write the data into the destination memory (the data buffer).

Data transmitted from memories mapped on the system I/F is temporarily stored in the TDM transmit local memory until it is transferred externally. A single data transfer from the system I/F to TDM local memory transfers at least 64 bits of data. Each channel can store more than 64 bits before it is transmitted externally. The TDMxTFP[TCDBL] field provides an upper boundary on the number of transmit bits that can be stored in TDM local memory. The transmit data latency is defined as the time between when the data is read from the buffers mapped to the system I/F and when it is transmitted externally. Reducing the TDMxTFP[TCDBL] value reduces the transmit data latency. However, writing the TDM local memory imposes more strict latency requirements on the internal MBus. The maximum transmit data latency is calculated as follows: TCDBL / TCS × transmit frame time.

When the TDM cannot transfer data from data buffers to TDM local memory, an underrun occurs. When the TDM transmit local memory is empty, the TDMxTER[ULBE] bit (see page 19-70) is set and the TDMxTIER[ULBEE] bit is also set, an error interrupt is generated. This error should not occur during normal operation. It indicates that the TDM has not received enough bandwidth on the system I/F and therefore cannot read the data from the source memory into TDM transmit local memory. The minimum latency is achieved when the RCDBL/TCDBL field is clear (only 64 bits are stored in the TDM local memory). For example, the minimum latency for a T1 application with 8 bits per channel and a frame length of 125 µs is equal to 1 ms. T1 minimum latency= 64/8 × 125 µs.

19.2.6 Buffers Mapped on System Memory

Each receive or transmit data channel is stored in a different buffer mapped on the internal system memory. This buffer can be located in any of the internal memories (such as the M2 or M3 memory) that are shared by all the SC3400 cores.

19.2.6.1 Data Buffer Size and A/μ-law Channels

Data buffer size is identical for all receive channels belonging to a TDM module and is indicated in the TDMxRDBS[RDBS] field. Data buffer size is also identical for all the transmit data buffers and is indicated in the TDMxTDBS[TDBS] field (see page 19-54). An exception is the A/μ-law channels (buffer size × 2). When the TDMxRCPRn[RCONV] field (see page 19-62) indicates that a channel is an A-law channel, the received 8 bits are converted into a 13-bit PCM sample padded with three zeros on the right. This channel therefore occupies 16 bits per 8 received bits, essentially occupying double the size. When the TDMxRCPRn[RCONV] field indicates that a channel is a μ-law channel, the received 8 bits are converted into a 14-bit PCM sample padded with two zeros on the right. This channel also occupies 16 bits per 8 received bits, essentially occupying double the size.

When the TDMxTCPRn[TCONV] field (see page 19-63) indicates that a channel is an A-law channel, the transmitted 13 bits are converted into an 8-bit PCM sample. This channel therefore occupies 16 bits (13 bits padded with three zeros at the right) per 8 transmit bits, essentially occupying double the size. When the TDMxTCPRn[TCONV] field indicates that a channel is a μ-law channel, the received 14 bits are converted into an 8-bit PCM sample. This channel also occupies 16 bits (14 bits padded with two zeros at the right) per 8 transmit bits, essentially occupying double the size. The A/μ-law conversion is performed according to the ITU-T recommendation G.711.

Note: The minimum buffer size for both transmit and receive is 16 bytes (that is, the RDBS/TDBS value is 0x00000F). The maximum buffer size for both transmit and receive is 16 MB (that is, the RDBS/TDBS value is 0xFFFFFF), but it can be further limited by the main memory size according to the number of channels in the frame.

Figure 19-24 shows how the samples are stored in the receive main data buffer (if the receive channel is A/μ law and TDMxRIR[RBOR] = 1) or the transmit main data buffer (if the transmit channel is A/μ law and TDMxTIR[TBOR] = 1).

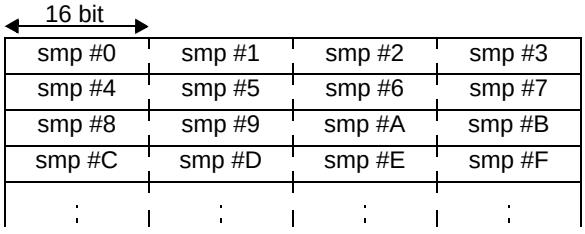


Figure 19-24. Receive/Transmit Main Data Buffer For A-Law/μ-Law Channel

19.2.6.2 Data Buffer Address

The address of a receive buffer is a function of the following:

- *Receive Global Base Address*. TDMxRGBA[RGBA], page 19-54.
- *Receive Channel Data Base Address*. TDMxRCPRn[RCDBA] field, page 19-62. RGBA << 16 + RCDBA points to the first byte of receive data buffer *n*. The four lsbs of RCDBA must be 0000.
- *Receive Data Buffer Displacement*. TDMxRDBDR[RDBD] field, page 19-67. Adding this field to the first byte of receive data buffer *n* indicates the location to which the TDM will write next: The address is calculated as follows: RGBA << 16 + RCDBA + RDBD. The RDBD can be used to show that data is written to the buffer and can be processed.

The address of a transmit buffer is a function of the following:

- *Transmit Global Base Address*. TDMxTGBA[TGBA] field, page 19-55.
- *Transmit Channel Data Base Address*. TDMxTCPRn[TCDBA] field, page 19-63. TGBA << 16 + TCDBA points to the first byte of transmit data buffer *n*. The four lsbs of TCDBA must be 0000.
- *Transmit Data Buffer Displacement*. TDMxTDBDR[TDBD] field, page 19-67. Adding this field to the first byte of transmit data buffer *n* indicates the location to which the TDM will read next: The address is calculated as follows: TGBA << 16 + TCDBA + TDBD. The TDBD can be used to show which data is already read from the buffer so that the buffer can be filled with new data.

Note: For A/μ-law channels the RDBD and the TDBD fields should be doubled before use.

Figure 19-25 illustrates the pointers associated with receive and transmit buffers that are mapped on the system I/F.

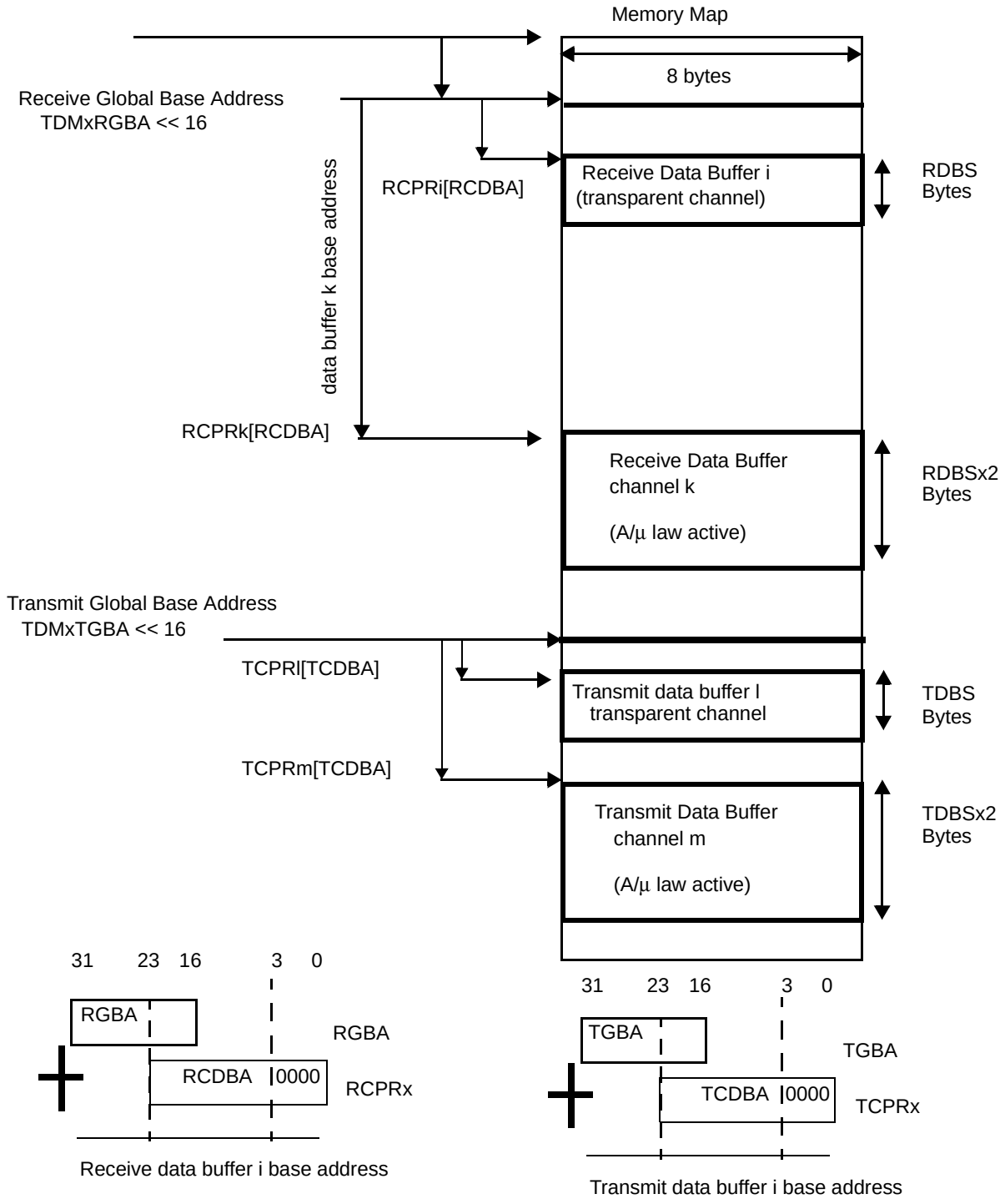


Figure 19-25. Data Buffer Location in Main Memory

19.2.6.3 Threshold Pointers and Interrupts

The receive data buffers share two threshold levels. The TDM notifies the SC3400 core each time it fills the receive buffer up to a threshold level. An example use of thresholds is the implementation of double buffering with the first threshold in the middle of a buffer and the second at the last eight bytes of the buffer.

When the TDM receiver fills the receive buffer through the system interface to an offset defined by the first threshold, which is the TDMxRDBFT[RDBFT] field (see page 19-59), the TDMxRER[RFTE] bit is set. If the TDMxRIER[RFTEE] bit is also set, a first threshold interrupt is generated. The interrupt can be generated as pulse or level, as determined by the TDMxRIR[RFTL] bit. If the interrupt is level, the ISR should clear the TDMxRER[RFTE] bit by writing a 1 to it. If the interrupt is pulse, then there is no need to clear the status bit. When the interrupt is asserted in the EPIC, then the SC3400 core can read all the receive buffers from their beginning up to the byte to which the first threshold (RDBFT) points. Meanwhile, the TDM keeps writing new data to the second part of the buffer.

When the TDM receiver fills the receive buffer through the system interface up to an offset defined by the second threshold, which is the TDMxRDBST[RDBST] field (see page 19-61), the TDMxRER[RSTE] bit is set. If the TDMxRIER[RSEEE] bit is also set, a second threshold interrupt is generated. The second threshold interrupt can generate as pulse or level, as determined by the TDMxRIR[RSTL] bit. If the interrupt is level, the ISR should clear the TDMxRER[RSTE] bit by writing a 1 to it. If the interrupt is pulse, there is no need to clear the status bit. When the interrupt is asserted in the EPIC, then the SC3400 core can read all the receive buffers up to the byte to which the second threshold (TDMxRDBST[RDBST]) points. Meanwhile, the TDM keeps writing new data to the first part of the buffer.

The transmit data buffers also share two threshold levels. The TDM notifies the SC3400 core each time it reads from the transmit buffer to a threshold level. When the TDM transmitter reads the transmit buffer through the system interface to an offset defined by the first threshold, which is the TDMxTDBFT[TDBFT] field, the TDMxTER[TFTE] bit is set. If the TDMxTIER[TFTEE] bit is also set, a first threshold interrupt is generated. The interrupt can generate as pulse or level, as determined by the TDMxTIR[TFTL] bit. If the interrupt is level, then the ISR should clear the TDMxTER[TFTE] bit by writing a 1 to it. If the interrupt is pulse, there is no need to clear the status bit. When the interrupt is asserted in the EPIC, then the SC3400 core can fill all the transmit buffers from their beginning up to the byte to which the first threshold (TDBFT) points. Meanwhile, the TDM continues reading new data from the second part of the buffer.

When the TDM transmitter reads the transmit buffer through the system interface up to an offset defined by the second threshold, which is the TDMxTDBST[TDBST] field, the TDMxTER[TSTE] bit is set. If the TDMxTEIR[TSTEE] bit is also set, a second threshold interrupt is generated. The second threshold interrupt can generate as pulse or level, as determined by the TDMxTIR[TSTL] bit. If the interrupt is level, the ISR should clear the TDMxTER[TSTE] bit by writing a 1 to it. If the interrupt is pulse, then there is no need to clear

the status bit. When the interrupt is asserted in the EPIC, then the SC3400 core can fill all the transmit buffers from their beginnings to the byte to which the second threshold (TDBST) points. Meanwhile, the TDM keeps reading new data from the buffer.

Figure 19-26 shows the threshold pointers for transparent and A/μ law channels.

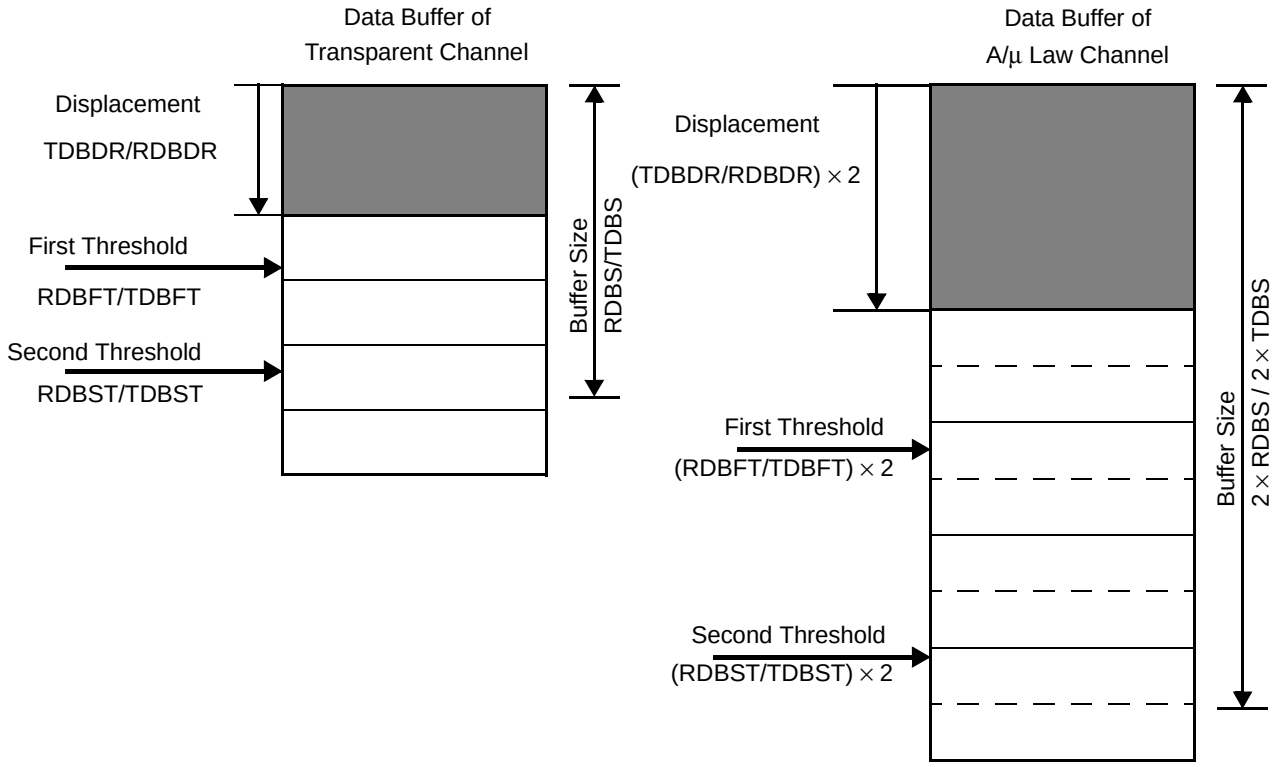


Figure 19-26. Main Memory Buffers Threshold Pointers

The TDMxRDBFT[RDBFT], TDMxRDBST[RDBST], TDMxTDBFT[TDBFT], and TDMxTDBST[TDBST] fields are control fields and can therefore be updated while the TDM is active. For example, to invoke an interrupt for each 64 bits written to the system I/F memory, the interrupt routine that handles the receive first threshold interrupt should include:

```
If (TDMxRDBFT[RDBFT] == (TDMxRDBS[RDBS] - 0xF))
    then TDMxRDBFT[RDBFT] = 0x0
else if (TDMxRDBFT[RDBFT] == (TDMxRDBS[RDBS] - 0x7))
    then TDMxRDBFT[RDBFT] = 0x8
else TDMxRDBFT[RDBFT] = TDMxRDBFT[RDBFT] + 0x10
```

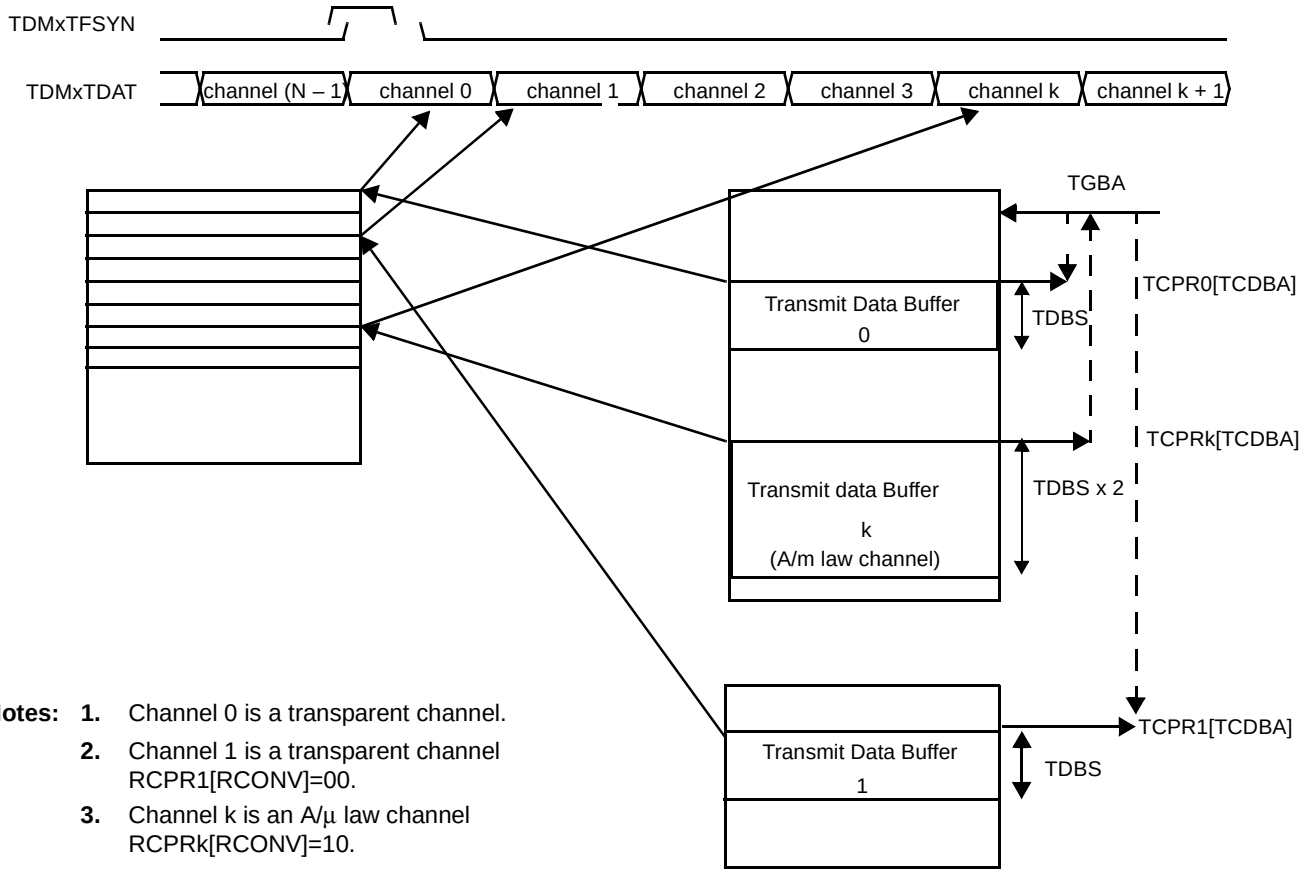
The interrupt routine that handles the receive second threshold interrupt should include:

```
If (TDMxRDBST[RDBST] == (TDMxRDBS[RDBS] - 0xF))
    then TDMxRDBST[RDBST] = 0x0
else if (TDMxRDBST[RDBST] == (TDMxRDBS[RDBS] - 0x7))
    then TDMxRDBST[RDBST] = 0x8
else TDMxRDBST[RDBST] = TDMxRDBST[RDBST] + 0x10
```

19.2.6.4 Unified Buffer Mode

When the TDMxRFP[RUBM] bit is set (see page 22-48), all receive channels are directed to one buffer through the system interface. The number of active links must be one (RTSAL = 0b0000 or 0b0100 or 0b1100). The channel parameters of all the receive channels are located in the TDMxRCPR0 register. Unified Buffer mode essentially creates a one-channel link that is typically used in point-to-point connections. When TDMxTFP[TUBM] =1, data is transmitted from one buffer into all the transmit channels.When TDMxTFP[TUBM] = 1, the first block of data initialized for transmission in the memory is not transmitted. The size (number of bits) of the non-transmitted block is determined by the following equation: 2 × (TDMxTFP[TNCF] - 1) × (TDMxTFP[TCS] + 1). For example, if the number of transmit channels (that is, TDMxTFP[TNCF] + 1) is 32, and the transmit channel size (that is, TDMxTFP[TCS] + 1) is 8 bits, then the number of non transmitted bits is 480. These bits (not transmitted) are located in the TDM local memory and/or in the device level memory (M2, M3, DDR, and so forth).

Figure 19-27 describes the transmit data flow in independent data buffers mode (TDMxTFP[TUBM] = 0). Each data channel transfers data from an independent data buffer to the TDM local memory, and transmit data is read out serially from the local memory.



Notes:
1. Channel 0 is a transparent channel.
2. Channel 1 is a transparent channel RCPR1[RCONV]=00.
3. Channel k is an A/μ law channel RCPRk[RCONV]=10.

Figure 19-27. Transmit Data Buffer in Independent Data Buffer Mode (TUBM=0)

Figure 19-28 illustrates the receive data flow in receive unified buffer mode. All the received channels are stored in the TDM local memory and then written into their unified buffer through the system I/F.

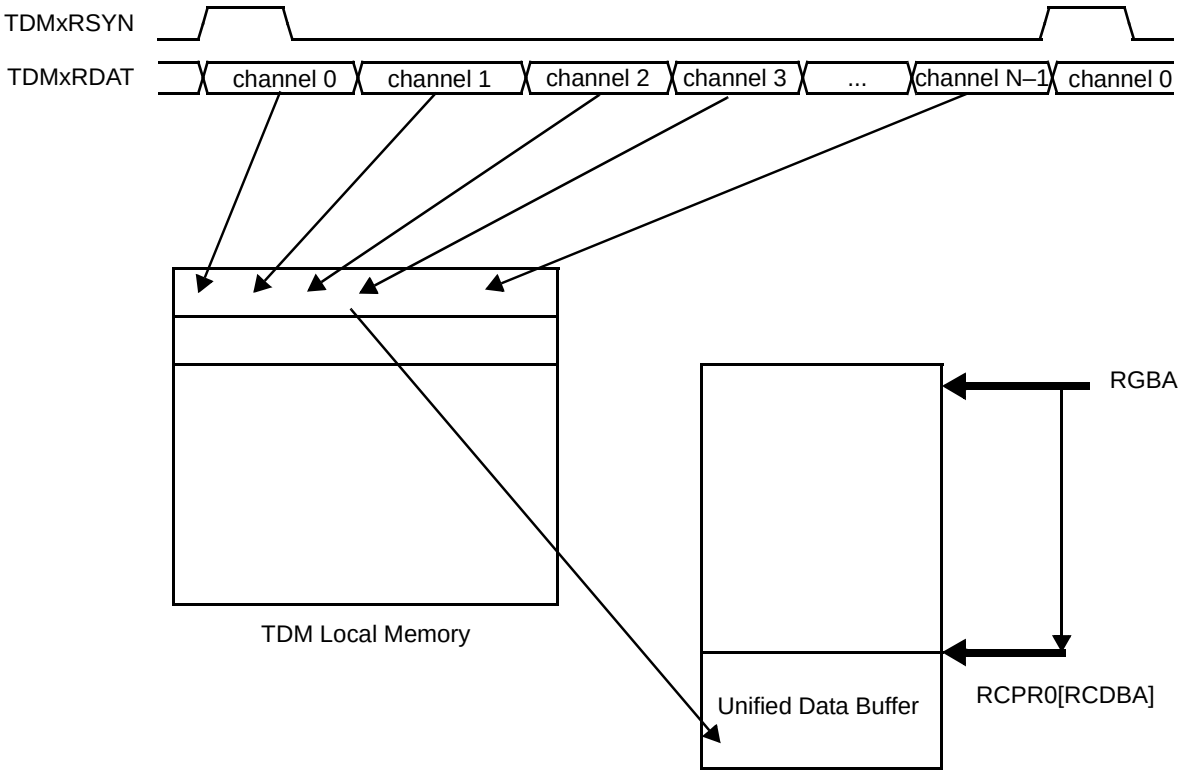


Figure 19-28. Receive Unified Buffer Mode (RUBM = 1)

Note: When the receiver is configured as Unified Buffer Mode, the RRDO bit in the TDMxRIR should be cleared. When the transmitter is configured as Unified Buffer Mode, the TRDO bit in the TDMxTIR should be cleared.

19.2.7 Adaptation Machine

Each TDM module has an Adaptation Machine that counts the number of bits between frame SYNCs. This module can be used to determine the frame size in bits.

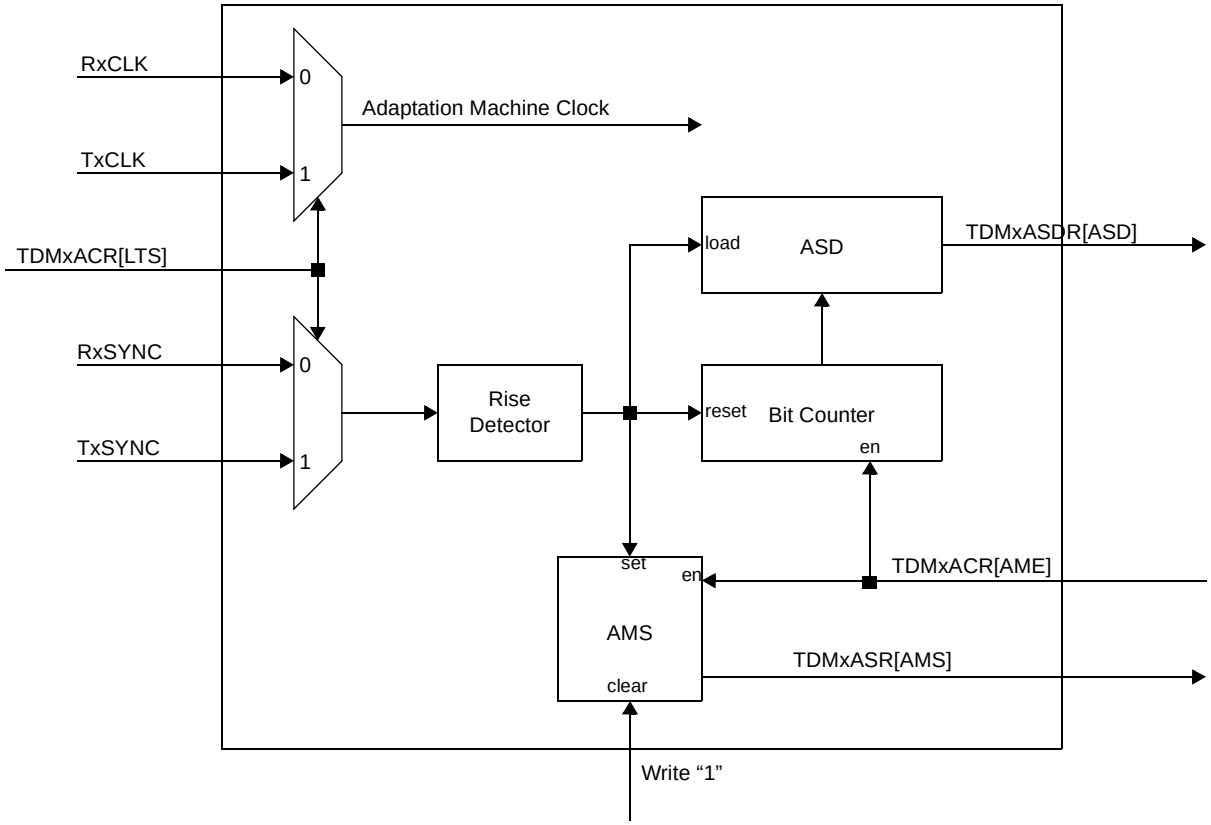


Figure 19-29. Adaptation Machine Block Diagram

Figure 19-29 shows the Adaptation Machine block diagram. The Adaptation Machine can work with either the transmit or receive frame. The LTS bit in the TDMx Adaptation Control Register (TDMxACR) defines whether the Adaptation Machine is fed with the transmit or with the receive frame sync and clock. The Adaptation Machine samples the sync only at the rising edge of the associated clock. When enabled, the Adaptation Machine detects a frame sync, stores the Bit Counter in the ASD field in the TDMx Adaptation Sync Distance Register (TDMxASDR), resets the Bit Counter, and sets the AMS bit in the TDMx Adaptation Status Register (TDMxASR).

The following steps define how to use the Adaptation Machine:

1. Configure the LTS bit to define whether the Adaptation Machine is fed with the Transmit or with the receive frame sync and clock. (See page 19-57).
2. Set the AME bit in the TDMxACR to enable the Adaptation Machine.
3. Wait for AMS bit in the TDMxASR to be set to 1. (See the TDMx Adaptation Status Register on page 19-71).
4. Read the value of the ASD field in the TDMxASDR. (See the TDMxASDR indicates the number of receive/transmit bits between the last two consecutive receive/transmit sync events. The register value updates each time the TDMxASR[AMS] bit is set. on page 19-71).

5. Clear the AMS bit by writing a 1 to the AMS bit in the TDMxASR.
6. Repeat steps 3–5 until you read the same value from the ASD field for 20 consecutive times. At this time the ASD value is valid and can be used to configure the TDM receiver or transmitter.
7. Clear the AME bit in the TDMxACR to disable the Adaptation Machine.
8. Configure the receiver or transmitter according to the following parameters:
   - ASD value.
   - Number of active links.
   - Channel size.
   - SYN

19.3 TDM Power Saving

The MSC8144E TDMs use the stop mode of different clocks to save power. Each TDM has three clock domains: transmit serial, receive serial, and the CLASS64 clock. The transmit serial clock is not supplied to the TDM module when the transmitter is disabled, that is, the TDMxTCR[TEN] bit and the TDMxTSR[TENS] are both clear. The receive serial clock is not supplied to the TDM module when the receiver is disabled, TDMxRCR[REN] bit and TDMxRSR[RENS] bit are both clear. The CLASS64 clock automatically stops when the TDM is disabled, that is, both transmitter and receiver are disabled. In addition, the TDM registers get the CLASS64 clock only at reset or during an access.

19.4 Channel Activation

The TACT and RACT bits in the Transmit/Receive Channel Parameter n Registers (see page 19-62 and page 19-63) are enabled during the receiver/transmitter operation to control the channel activation. If the TACT/RACT bit is clear, the channel is not active. Otherwise, it is active. The procedure for activating an inactive receive channel (C) is as follows:

1. Verify that the active (RACT) bit of the channel is clear.
2. Write the initialization value to the channel locations in the receive TDM local memory.

   The receive local memory contains 1, 2, 4, 8, 16, or 32 buffers so that each buffer contains 8 bytes per channel. The location of channel C in buffer B is the 8 bytes that start at $(256 / (RNB + 1) \times B + C) \times 8$. (See **Section 19.2.3**, *TDM Data Structures*, on page 19-13). For example, if the number of buffers is four, the SC3400 core should write the initialization value to all four receive buffers. Initializing the receive TDM local memory prevents invalid data from being received by the channel buffer in the main memory.
3. Set the TDMxRCPRC[RACT] bit (C indicates the channel number).

The procedure for activating an inactive transmit channel (C) is as follows:

1. Verify that the active (TACT) bit of the channel is clear.
2. Write the initialization value to the channel locations in the transmit TDM local memory.

   The transmit local memory contains 1, 2, 4, 8, 16, or 32 buffers so that each buffer contains 8 bytes per channel. The location of channel C in buffer B is the 8 bytes that start at (256 / (TNB + 1) × B + C) × 8. (See **Section 19.2.3**, *TDM Data Structures*, on page 19-13). Initializing the transmit TDM local memory prevents invalid data from being transmitted out.
3. Set the TDMxTCPRC[TACT] bit (C indicates the channel number).

   For example, if the SC3400 core needs to activate receive channel 2 and the number of receive buffers is 4 (RNB[3–0] = 0011), it should write the initialization value to the following addresses (which are offsets from the TDMx receive local memory, see **Chapter 9**, *Memory Map*):
   - 0x0010–0x0017 (the channel location in buffer 0).
   - 0x0210–0x0217 (the channel location in buffer 1).
   - 0x0410–0x0417 (the channel location in buffer 2).
   - 0x0610–0x0617 (the channel location in buffer 3).

19.5 Loopback Support

In Loopback Test mode, the receiver receives the same data that is transmitted. The frame clock should supply to the TDM, and the frame sync can be generated internally or supplied externally. The receiver and transmitter share the frame sync, frame clock, and data links (RTSAL[3–2] = 0b11). The number of data links can be 1, 2, or 4 and is determined by the RTSAL[1–0] bits. All the receive and transmit frame channels are active. The procedure for loopback is as follows:

1. Configure the RTSAL field in the GIR register (see page 19-36) to shared data links mode- RTSAL[3–2] = 0b11. The number of data links can be 1, 2, or 4.
2. Configure the receive and transmit frame parameters to be the same. The configuration of the RFP register should be identical to that of the TFP register (see page 19-48 and page 19-51).
3. Configure the TDMx Transmit Interface Register (TDMxTIR) and the TDMx Receive Interface Register (TDMxRIR) according to the following instructions:
   - Set the Transmit Sync Out (TSO) bit to 1. The transmit sync is generated by the TDM.
   - Set the Receive Frame Sync Delay field to 0x00 and the Transmit Frame Sync Delay field to 0x01.
   - Set both the Receive Frame Sync Edge (RFSE) bit and the Transmit Frame Sync Edge (TFSE) bits to 1. The sync samples at the negative edge.
   - The value of the Receive Sync level bit should be identical to that of the Transmit Sync Level field (RSL = TSL).

- Clear the Receive Data Edge bit (RDE = 0x0), so that the receive data is sampled on the positive edge.
- Set the Transmit Data Edge bit (TDE = 0x1) to transmit data driven at the negative edge.

4. Set the receive active RACT bit of all the channels to 1.
5. Set the transmit active TACT bit of all the channels to 1.
6. Set the TDMxTCR[TEN] bit.
7. Set the TDMxRCR[REN] bit.

19.6 TDM Initialization

After reset, all TDM registers are reset. **Table 19-5** describes the TDM signal direction after reset.

Table 19-5. TDM Signal Direction at Reset

TDM Signal	Signal Direction
TDMXRCLK	input
TDMxRDAT	input
TDMxRSYN	input
TDMxTCLK	input
TDMxTDAT	input
TDMxTSYN	input

The TDMxRCR[REN] bit (see page 19-58) enables the receiver part of the TDM module. When TDMxRCR[REN] is clear, the receive TDM is disabled, but all the registers retain their values except for the TDMx Receive Data Buffers Displacement Register (TDMxRDBDR). The TDMxTCR[TEN] bit (see page 19-59) enables the transmit part of the TDM module. When TDMxTCR[TEN] is clear, the transmit TDM is disabled, but all the registers retain their values except for the TDMx Transmit Data Buffers Displacement Register (TDMxTDBDR).

The correct flow for initializing the TDM is as follows:

1. Perform a hardware reset to disable the receiver and the transmitter.
2. Program all the configuration registers. Program all the control registers, except the TDMx Receive Control Register (TDMxRCR) and the TDMx Transmit Control Register (TDMxTCR).

3. Fill the sync data in all the TDM receive local memory.

   The received data is stored in 256 entries of 8 bytes each located in the addresses between 0x0000–0x07FF. This memory contains 1, 2, 4, 8, 16, or 32 indexed buffers, starting at 0. Each buffer contains multiple frames. The number of buffers used to store the received data is indicated in the RNB field of the TDMx Receive number of Buffers Register (TDMxRNB) (see page 19-68). Channel C in buffer B is the 8 bytes starting at $(256 / (RNB + 1) \times B + C) \times 8$. (Refer to **Section 19.2.3**, *TDM Data Structures*, on page 19-13 for details)

4. Fill the sync data in all the TDM transmit local memory.

   Transmit data is located in the TDM local memory before it is transmitted externally. The data is stored in 256 8-byte entries in addresses between 0x1800–0x27FF. This memory can contain 1, 2, 4, 8, 16, or 32 indexed buffers starting at 0. Each buffer contains multiple frames. The number of buffers used to store the transmitted data is indicated in the TNB field of the TDMx Transmitter Number of Buffers Register (TDMxTNB). Channel C in buffer B is the 8 bytes starting at $(256/(TNB+1) \times B+C) \times 8$.

5. Clear the TDMxRER and TDMxTER event registers by writing a value of 0xF to each of them.

6. Set the TDMxRCR[REN] bit and/or the TDMxTCR[TEN] bit.

19.7 TDM Programming Model

The handshake between the TDM module and the SC3400 core occurs via a set of registers, data structures in the memory, and interrupts. All TDM registers are mapped into the CCSR address space. See **Chapter 9**, *Memory Map* for details on CCSR addressing. There are eight modules (TDM 0–7), each with its own region in the CCSR address space. Within the module address space, the area is divided into spaces for configuration registers, control registers, and status registers as follows:

- *Configuration registers*. Set the operation modes and provide indications for all channels. They are set before the TDM is enabled and should not be changed while the TDM is active.
- *Control registers*. Set the channel specific parameters individually for each channel and the threshold pointers. These registers can be changed during operation.
- *Status registers*. Read-only registers that can be accessed any time.

This section describes the TDM module registers, which are listed as follows:

- TDMx General Interface Register (TDMxGIR), page 19-36.
- TDMx Receive Interface Register (TDMxRIR), page 19-44.
- TDMx Transmit Interface Register (TDMxTIR), page 19-46.
- TDMx Receive Frame Parameters (TDMxRFP), page 19-48.
- TDMx Transmit Frame Parameters (TDMxTFP), page 19-51.
- TDMx Receive Data Buffer Size (TDMxRDBS), page 19-53.
- TDMx Transmit Data Buffer Size (TDMxTDBS), page 19-54.
- TDMx Receive Global Base Address (TDMxRGBA), page 19-54.
- TDMx Transmit Global Base Address (TDMxTGBA), page 19-55.
- TDMx Transmit Force Register (TDMxTFR), page 19-55
- TDMx Receive Force Register (TDMxRFR), page 19-56
- TDMx Parity Control Register (TDMxPCR), page 19-57
- TDMx Adaptation Control Register (TDMxACR), page 19-57.
- TDMx Receive Control Register (TDMxRCR), page 19-58.
- TDMx Transmit Control Register (TDMxTCR), page 19-59.
- TDMx Receive Data Buffers First Threshold (TDMxRDBFT), page 19-59.
- TDMx Transmit Data Buffers First Threshold (TDMxTDBFT), page 19-60.
- TDMX Receive Data Buffers Second Threshold (TDMxRDBST), page 19-61.
- TDMx Transmit Data Buffers Second Threshold (TDMxTDBST), page 19-61.
- TDMx Receive Channel Parameter Register 0–255 (TDMxRCPR[0–255]), page 19-62.
- TDMx Transmit Channel Parameter Register 0–255 (TDMxTCPR[0–255]), page 19-63.
- TDMx Receive Interrupt Enable Register (TDMxRIER), page 19-64.
- TDMx Transmit Interrupt Enable Register (TDMxTIER), page 19-65.
- TDMx Adaptation Sync Distance Register (TDMxASDR), page 19-66.
- TDMx Receive Data Buffers Displacement Register (TDMxRDBDR), page 19-67
- TDMx Transmit Data Buffers Displacement Register (TDMxTDBDR), page 19-67.
- TDMx Receive Number of Buffers (TDMxRNB), page 19-68.
- TDMx Transmitter Number of Buffers (TDMxTNB), page 19-69.
- TDMx Receive Event Register (TDMxRER), page 19-69.
- TDMx Transmit Event Register (TDMxTER), page 19-70.
- TDMx Adaptation Status Register (TDMxASR), page 19-71.
- TDMx Receive Status Register (TDMxRSR), page 19-72.
- TDMx Transmit Status Register (TDMxTSR), page 19-73.
- TDMx Parity Error Register (TDMxPER), page 19-73.

Note: The base addresses for the TDM registers are as follows:

- TDM0 = 0xFFF30000
- TDM1 = 0xFFF34000
- TDM2 = 0xFFF38000
- TDM3 = 0xFFF3C000
- TDM4 = 0xFFF40000
- TDM5 = 0xFFF44000
- TDM6 = 0xFFF48000
- TDM7 = 0xFFF4C000

19.7.1 Configuration Registers

The following sections describe the configuration registers.

19.7.1.1 TDMx General Interface Register (TDMxGIR)

TDM*x*GIR TDM*x* General Interface Register Offset 0x3FF8

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

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

TDM*x*GIR defines the TDM*x* interface operation mode.

Table 19-6. TDM*x*GIR Bit Descriptions

Name	Reset	Description	Settings
— 31–6	0	Reserved. Write to zero for future compatibility.	
SYNC_MODE 5	0	**Sync Mode** Used to synchronize the start of each TDM receiver/transmitter with enabling the TDM0 transmitter. In this mode, the TDM0 transmitter should be the last to operate.	0 Non-sync mode. 1 Sync mode.

Table 19-6. TDM*x*GIR Bit Descriptions (Continued)

Name	Reset	Description	Settings
CTS 4	0	**Common TDM Signals** Defines whether the TDM shares sync and clock signals with other TDM modules. The eight TDMs can share signals as follows: ■ TDM[0–7] do not share signals. (TDMxCTS = 0 for 0 ≤ x ≤ 7) ■ TDM0 and TDM1 share signals, but TDM[2–7] do not share signals with the other TDM modules. (TDMxCTS = 1for 0 ≤ x ≤ 1, TDMyCTS=0 for 2 ≤ y ≤ 7) ■ TDM[0–2] share signals, but TDM[3–7] do not share signals with the other TDM modules. (TDMxCTS=1for 0 ≤ x ≤ 2, TDMyCTS=0 for 3 ≤ y ≤ 7) ■ TDM[0–3] share signals, but TDM[4–7] do not share signals with the other TDM modules. (TDMxCTS=1for 0 ≤ x ≤ 3, TDMyCTS=0 for 4 ≤ y ≤ 7) ■ TDM[0–4] share signals, but TDM[5–7] do not share signals with the other TDM modules. (TDMxCTS=1for 0 ≤ x ≤ 4, TDMyCTS=0 for 5 ≤ y ≤ 7) ■ TDM[0–5] share signals, but TDM[6–7] do not share signals with the other TDM modules. (TDMxCTS=1for 0 ≤ x ≤ 5, TDMyCTS=0 for 6 ≤ y ≤ 7) ■ TDM[0–6] share signals, but TDM[7] do not share signals with the other TDM modules. (TDMxCTS=1for 0 ≤ x ≤ 6, TDM7CTS=0) ■ TDM[0–7] share signals (TDMxCTS=1for 0 ≤ x ≤ 7) **Table 19-7** on page 19-39 describes the functionality of the TDM signals as a function of the **RTSAL** field. **Note:** If the TDM modules share sync and clock signals, then the RFP, TFP,RIR, and TIR registers should be configured the same way for all the TDM modules.	0 The TDM does not share signals with other TDM modules 1 The TDM shares sync and clock signals with other TDM modules. Refer to **Table 19-7**.

Table 19-6. TDM*x*GIR Bit Descriptions (Continued)

Name	Reset	Description	Settings
RTSAL 3–0	0	**Receive and Transmit Sharing and Active Links** Defines the TDM serial interface operating mode. It determines whether the TDM transmit and receive paths are independent or share the same clock and sync. It also determines whether the TDM receive and transmit share the data links. Bits 2 and 3 determine the receive and transmit sharing mode, and bits 1 and 0 determine the number of active data links. **Note:** If RTSAL [3–2]= 01 or 11, some parameters of the receive and transmit path should be the same. The value of the TDMxRFP[RNCF], RCS and RT1 fields should be equal to that of the TDMxTFP[TNCF], TCS, and TT1 fields. The value of the TDMxRIR[RFSE] and TDMxRIR[RSL] fields should be equal to the that of the TDMxTIR[TFSE] and TDMxTIR[TSL] fields, respectively. For details, see **Section 19.2.1**, *Common Signals for the TDM Modules,* on page 19-8. **Note:** Unused signals should not be configured as dedicated signals in the PAR.	0000 The receive and transmit are independent.The TDM receives one data link and transmits one data link. 0001 The receive and transmit are independent. The TDM receives two data links and transmits two data links (valid only if CTS=1). 0010 Reserved 0011 Reserved. 0100 The receive and transmit share the frame clock and frame sync.The TDM receives one data link and transmits one data link. 0101 The receive and transmit share the frame sync and frame clock. The TDM receives two data links and transmits two data links. 0110–1011 Reserved. 1100 The receive and transmit share the frame sync, frame clock, and one full duplex data link. 1101 The receive and transmit share the frame sync, frame clock, and two full duplex data links. 1110 Reserved. 1111 The receive and transmit share the frame sync, frame clock, and four full duplex data links. Refer to **Table 19-8**

Table 19-7. TDM Signal Configuration When TDM Modules Share Signals

RTSAL[3–0] Field Value	Description
0000	Receive clock: TDM1TCLK Transmit clock: TDM0TCLK Receive sync: TDM1TSYN Transmit sync: TDM0TSYN Receive data links: TDMxRDAT Transmit data links: TDMxTDAT Unused signals: TDMxRCLK, TDMxRSYN. **Note:** The x specifies the TDM number of TDMs that share signals. For example if TDM0, TDM1, and TDM2 share signals, then x is equal to 0,1, and 2 and the receive data links are TDM0RDAT, TDM1RDAT, and TDM2RDAT.
0001	Receive clock: TDM1TCLK Transmit clock: TDM0TCLK Receive sync: TDM1TSYN Transmit sync: TDM0TSYN Receive data links: TDMxRDAT, TDMxRSYN. Transmit data links: TDMxTDAT, TDMxRCLk. **Note:** The x specifies the number of the TDM and any one of the shared TDM modules.
0100	Frame clock (receive and transmit share the same clock): TDM0TCLK Frame sync (receive and transmit share the same sync): TDM0TSYN Receive data links: TDMxRDAT Transmit data links: TDMxTDAT Unused signals: TDMxRCLK, TDMxRSYN, TDMyTCLK, TDMyTSYN TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except TDM0. For example, if TDM0 and TDM1 share signals, the unused signals are TDM0RCLK, TDM1RCLK, TDM0RSYN, TDM1RSYN, TDM1TCLK, and TDM1TSYN.
0101	Frame clock (receive and transmit share the same clock): TDM0TCLK Frame sync (receive and transmit share the same sync): TDM0TSYN Receive data links: TDMxRDAT, TDMxRSYN Transmit data links: TDMxTDAT, TDMxRCLK Unused signals: TDMyTCLK, TDMyTSYN TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except TDM0. For example, if TDM0 and TDM1 share signals, the unused signals are TDM1TCLK and TDM1TSYN.
1100	Frame clock (receive and transmit share the same clock): TDM0TCLK Frame sync (receive and transmit share the same sync): TDM0TSYN Full duplex data links (the data link is inout and is used for receive and transmit) TDMxRDAT. Unused signals: TDMyTCLK, TDMyTSYN, TDMxRCLK, TDMxRSYN, and TDMxTDAT. TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except TDM0. For example, if TDM0 and TDM1 share signals, the unused signals are TDM0RCLK, TDM1RCLK, TDM0RSYN, TDM1RSYN, TDM0TDAT, TDM1TDAT, TDM1TCLK, and TDM1TSYN.
1101	Frame clock (receive and transmit share the same clock): TDM0TCLK Frame sync (receive and transmit share the same sync): TDM0TSYN Full duplex data links (the data link is inout it and it is used for receive and transmit): TDMxRDAT, TDMxRSYN Unused signals: TDMyTCLK, TDMyTSYN, TDMxRCLK, TDMxTDAT. TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except of TDM0. for example if TDM0 and TDM1 shared signals then the unused signals are TDM0RCLK, TDM1RCLK, TDM0TDAT, TDM1TDAT, TDM1TCLK, and TDM1TSYN.

Table 19-7. TDM Signal Configuration When TDM Modules Share Signals (Continued)

RTSAL[3–0] Field Value	Description
1111	Frame clock (receive and transmit share the same clock): TDM0TCLK Frame sync (receive and transmit share the same sync): TDM0TSYN Full duplex data links (the data link is inout and is used for receive and transmit): TDMxRDAT,TDMxRSYN,TDMxTDAT,TDMxRCLK Unused signals: TDMyTCLK, TDMyTSYN TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except TDM0. For example, if TDM0 and TDM1 share signals, the unused signals are TDM1TCLK and TDM1TSYN.

Table 19-8. TDM Signal Configuration as a Function of the RTSAL and CTS Fields

No.	CTS	RTSAL [3–0]	TDMxRDAT	TDMxRSYN	TDMxRCLK	TDMxTDAT	TDMxTSYN	TDMxTCLK	Comments
0	0	0000	receive data (RDATA_A)	receive sync	receive clock	transmit data (TDATA_A)	transmit sync	transmit clock	The TDM does not share signals with others TDM modules. Independent mode. One active data link.
	direction		Input	Input	Input	Output	Inout	Input	
1	0	0001	Reserved						
2	0	0010	Reserved						
3	0	0011	Reserved						
4	0	0100	receive data (RDATA_A)	not used	not used	transmit data (TDATA_A)	frame sync	frame clock	The TDM does not share signals with others TDM modules. Receive and transmit share sync and clock signals. One active data link.
	direction		input			Output	Inout	Input	

Table 19-8. TDM Signal Configuration as a Function of the RTSAL and CTS Fields (Continued)

No.	CTS	RTSAL [3–0]	TDMxRDAT	TDMxRSYN	TDMxRCLK	TDMxTDAT	TDMxTSYN	TDMxTCLK	Comments
5	0	0101	receive data (RDATA_A)	receive data (RDATA_B)	transmit data (TDATA_B)	transmit data (TDATA_A)	frame sync	frame clock	The TDM does not share signals with other TDM modules. Receive and transmit share sync and clock signals. Two active data links.
	direction		Input	Input	Output	Output	Inout	Input	
6	0	0110	Reserved						
7	0	0111	Reserved						
8	0	1100	data link (DATA_A)	not used	not used	not used	frame sync	frame clock	The TDM does not share signals with other TDM modules. Receive and transmit share the sync, clock, and data signals. One full duplex active data link.
	direction		Inout				Inout	Input	
9	0	1101	data link (DATA_A)	data link (DATA_B)	not used	not used	frame sync	frame clock	The TDM does not share signals with other TDM modules. Receive and transmit share the sync, clock, and data signals. Two full duplex active data links.
	direction		Inout	Inout			Inout	Input	
10	0	1110	Reserved						

Table 19-8. TDM Signal Configuration as a Function of the RTSAL and CTS Fields (Continued)

No.	CTS	RTSAL [3–0]	TDMxRDAT	TDMxRSYN	TDMxRCLK	TDMxTDAT	TDMxTSYN	TDMxTCLK	Comments
11	0	1111	data link (DATA_A)	data link (DATA_B)	data link (DATA_D)	data link (DATA_C)	frame sync	frame clock	The TDM does not share signals with other TDM modules. Receive and transmit share the sync, clock, and data signals. Four full duplex active data links.
	direction		Inout	Inout	Inout	Inout	Inout	Input	
12	1	0000	receive data (RDATA_A)	not used	not used	transmit data (TDATA_A)	receive sync/ transmit sync/ not used	receive clock/ transmit clock/ not used	The TDM shares receive sync and clock and transmit sync and clock with other TDM modules. Independent mode. One active data link.
	direction		Input			Output	Inout	Input	
13	1	0001	receive data (RDATA_A)	receive data (RDATA_B)	transmit data (TDATA_B)	transmit data (TDATA_A)	receive sync/ transmit sync/ not used	receive clock/ transmit clock/ not used	The TDM shares receive sync and clock and transmit sync and clock with other TDM modules. Independent mode. Two active data links.
	direction		Input	Input	Output	Output	Inout	Input	
14	1	0010	Reserved						
15	1	0011	Reserved						
16	1	0100	receive data (RDATA_A)	not used	not used	transmit data (TDATA_A)	frame sync/ not used	frame clock/ not used	The TDM shares the frame sync and frame clock with other TDM modules. Receive and transmit shared sync and clock signals. One active data link.
	direction		input			Output	Inout	Input	

Table 19-8. TDM Signal Configuration as a Function of the RTSAL and CTS Fields (Continued)

No.	CTS	RTSAL [3–0]	TDMxRDAT	TDMxRSYN	TDMxRCLK	TDMxTDAT	TDMxTSYN	TDMxTCLK	Comments
17	1	0101	receive data RDATA_A	receive data RDATA_B	transmit data (TDATA_B)	transmit data (TDATA_A)	frame sync/ not used	frame clock/ not used	The TDM shares the frame sync and frame clock with other TDM modules. Receive and transmit share the sync and clock signals. Two active data links.
	direction		Input	Input	Output	Output	Inout	Input	
18	10	0110	Reserved						
19	1	0111	Reserved						
20	1	1100	data link (DATA_A)	not used	not used	not used	frame sync/ not used	frame clock/ not used	The TDM shares the frame sync and frame clock with other TDM modules. Receive and transmit share the sync, clock, and data signals. One full duplex active data link.
	direction		Inout				Inout	Input	
21	1	1101	data link (DATA_A)	data link (DATA_B)	not used	not used	frame sync/ not used	frame clock/ not used	The TDM share the frame sync and frame clock with other TDM modules. Receive and transmit share the sync, clock, and data signals. Two full duplex active data links.
	direction		Inout	Inout			Inout	Input	
22	1	1110	Reserved						

Table 19-8. TDM Signal Configuration as a Function of the RTSAL and CTS Fields (Continued)

No.	CTS	RTSAL [3–0]	TDMxRDAT	TDMxRSYN	TDMxRCLK	TDMxTDAT	TDMxTSYN	TDMxTCLK	Comments
23	1	1111	data link (DATA_A)	data link (DATA_B)	data link (DATA_D)	data link (DATA_C)	frame sync/ not used	frame clock/ not used	The TDM shares the frame sync and frame clock with other TDM modules. Receive and transmit share the sync, clock, and data signals. Four full duplex active data links.
	direction		Inout	Inout	Inout	Inout	Inout	Input	
Note: Frame sync specifies that the receiver and transmitter use the same sync. Frame clock specifies that the receiver and transmitter use the same clock. If data link specifies that the direction is inout, the signal is used for receive and transmit.									

19.7.1.2 TDMx Receive Interface Register (TDMxRIR)

TDM*x*RIR TDM*x* Receive Interface Register Offset 0x3FF0

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RFTL	RSTL	—								RFSD		RSL	RDE	RFSE	RRDO
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RIR defines the TDM*x* receiver interface operation.

Table 19-9. TDM*x*RIR Bit Descriptions

Name	Reset	Description	Settings
— 31–17	0	Reserved. Write to zero for future compatibility.	
RBOR 16	1	**Receive Byte Order** Indicates the location of the receive data in the receive external memory buffer. The boot program writes a 1 to this bit.	0 First receive data is at the high address. 1 First receive data is at the low address.
RFTL 15	0	**Receive First Threshold Level** Determines whether the receive first threshold interrupt is pulse or level. For details, see **Section 19.2.6.3**.	0 Receive first threshold interrupt is pulse. 1 Receive first threshold interrupt is level.

Table 19-9. TDM*x*RIR Bit Descriptions (Continued)

Name	Reset	Description	Settings
RSTL 14	0	**Receive Second Threshold Level** Determines whether the receive second threshold interrupt is pulse or level. For details, see **Section 19.2.6.3**.	0 Receive second threshold interrupt is pulse. 1 Receive second threshold interrupt is level.
— 13–6	0	Reserved. Write to zero for future compatibility.	
RFSD 5–4	0	**Receive Frame Sync Delay** With the RDE and the RFSE bits, determines the number of clocks between the receive sync signal and the first data bit of the receive frame.	Refer to **Table 19-10**. **Note:** If the receive channel size is 2 (RCS = 0x1), then the RFSD field value can be only 0 or 1.
RSL 3	0	**Receive Sync Level** Determines the polarity of the receive sync signal. For details, see **Figure 19-21**.	0 Receive sync is active on logic 1. 1 Receive sync is active on logic 0.
RDE 2	0	**Receive Data Edge** Determines whether the receive data signal is sampled on the rising or falling edge of the receive clock. For details see **Section 19.2.4.2**.	0 The receive data signal is sampled on the rising edge of the receive clock. 1 The receive data signal is sampled on the falling edge of the receive clock.
RFSE 1	0	**Receive Frame Sync Edge** Determines whether the receive frame sync signal is sampled on the rising or falling edge of the receive clock. For details, see **Section 19.2.4.2**.	0 The receive frame sync signal is sampled with the rising edge of the receive clock. 1 The receive frame sync signal is sampled on the falling edge of the receive clock.
RRDO 0	0	**Receive Reversed Data Order** For examples, see **Section 19.2.4.2**.	0 The first bit of a received channel is stored as the most significant bit in the internal memory. 1 The first bit of a received channel is stored as the least significant bit in the internal memory

Table 19-10. Received Data Delay for Receive Frame Sync

Frame Sync Delay	Frame Sync Edge	Data Edge	Receive Clocks[1]
00	0	0	0.0
00	0	1	0.5
00	1	0	0.5
00	1	1	0.0
01	0	0	1.0
01	0	1	1.5
01	1	0	1.5
01	1	1	1.0
10	0	0	2.0
10	0	1	2.5
10	1	0	2.5
10	1	1	2.0
11	0	0	3.0
11	0	1	3.5
11	1	0	3.5
11	1	1	3.0

Note: Receive clocks is the number of receive clocks between the sample of the receive frame sync and the sample of first data bit of the received frame.

19.7.1.3 TDMx Transmit Interface Register (TDMxTIR)

TDM*x*TIR TDM*x* Transmit Interface Register Offset 0x3FE8

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TFTL	TSTL	TSO	TAO	SOL	SOE	—				TFSD		TSL	TDE	TFSE	TRDO
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDMxTIR defines the TDM *x* transmitter interface operation.

Table 19-11. TDM*x*TIR Bit Descriptions

Name	Reset	Description	Settings
— 31–17	0	Reserved. Write to zero for future compatibility.	
TBOR 16	1	**Transmit Byte Order** Indicates how to transmit data residing in the transmit external memory buffer. The boot program writes a 1 to this bit.	0 Transmit high address first. 1 Transmit low address first.
TFTL 15	0	**Transmit First Threshold Level** Determines whether the Transmit first threshold interrupt is pulse or level. For details, see **Section 19.2.6.3.**	0 Transmit first threshold interrupt is pulse. 1 Transmit first threshold interrupt is level.

Table 19-11. TDM*x*TIR Bit Descriptions (Continued)

Name	Reset	Description	Settings
TSTL 14	0	**Transmit Second Threshold Level** Determines whether the Transmit second threshold interrupt is pulse or level. For details, see **Section 19.2.6.3**.	0 Transmit second threshold interrupt is pulse. 1 Transmit second threshold interrupt is level.
TSO 13	0	**Transmit Sync Output** Determines whether the transmit sync is driven out by the TDM transmitter or it input to the TDM transmitter. For details, see **Section 19.2.4.1**	0 Transmit sync is input. 1 Transmit sync is output.
TAO 12	0	**Transmit Always Output** Determines whether the TDM transmitter drives TDMxDAT for the inactive channels.	0 The TDM transmitter does not drive the TDMxDAT for inactive channels. 1 The TDM transmitter drives the TDMxDAT regardless of whether the channel is active.
SOL 11	0	**Sync Out Length** Indicates whether the TDMxTSYN is asserted for one cycle of TDMxTCLK or is asserted for the duration of the first channel in the frame. For details, see **Section 19.2.4.1**.	0 The sync_out width is one bit. 1 The sync_out width is equal to the channel width.
SOE 10	0	**Sync Out Edge** Determines whether the sync out signal is driven out with the rising or falling edge of the transmit clock. For details, see **Section 19.2.4.1**.	0 The transmit frame sync signal is driven out on the rising edge of the transmit clock. 1 The transmit frame sync signal is driven out on the falling edge of the transmit clock.
— 9–6	0	Reserved. Write to zero for future compatibility.	
TFSD 5–4	0	**Transmit Frame Sync Delay** With the TDE and the TFSE bits, determines the number of clocks between the transmit sync signal and the first data bit of the transmit frame. For examples, see **Section 19.2.4.2**.	Refer to Table 19-12 on page 47. **Note:** If the transmit channel size is 2 (TCS = 0x1) then the TFSD field value can be only 0 or 1.
TSL 3	0	**Transmit Sync Level** Determines the polarity of the transmit sync signal. For details, see **Section 19.2.4.2**.	0 Transmit sync is active on logic 1. 1 Transmit sync is active on logic 0.
TDE 2	0	**Transmit Data Edge** Determines whether the transmit data is driven out on the rising or falling edge of the transmit clock. For details, see **Section 19.2.4.2**.	0 The transmit data is driven out on the rising edge of the transmit clock. 1 The transmit data is driven out on the falling edge of the transmit clock.
TFSE 1	0	**Transmit Frame Sync Edge** Determines whether the transmit frame sync signal is sampled with the rising or falling edge of the transmit clock. For details, see **Section 19.2.4.2**.	0 The transmit frame sync signal is sampled with the rising edge of the transmit clock. 1 The transmit frame sync signal is sampled with the falling edge of the transmit clock.
TRDO 0	0	**Transmit Reversed Data Order** For examples, see **Section 19.2.4.4**.	0 The most significant bit of the memory is sent out at the first transmit data bit. 1 The least significant bit of the memory is sent out at the first transmit data bit.

Table 19-12. Transmit Data Delay for Transmit Frame Sync

Frame Sync Delay	Frame Sync Edge	Data Edge	Transmit Clocks[1]
00	0	0	–1 [2]
00	0	1	–0.5 [2]
00	1	0	–0.5 [2]

Table 19-12. Transmit Data Delay for Transmit Frame Sync (Continued)

Frame Sync Delay	Frame Sync Edge	Data Edge	Transmit Clocks[1]
00	1	1	–1 [2]
01	0	0	0
01	0	1	0.5
01	1	0	0.5
01	1	1	0
10	0	0	1
10	0	1	1.5
10	1	0	1.5
10	1	1	1
11	0	0	2
11	0	1	2.5
11	1	0	2.5
11	1	1	2

Notes: **1.** Transmit clocks is the number of transmit clocks between the first transmit frame sync sample and the first data bit of the frame that is driven out.

2. The field value is negative because the data is driven out before the transmit frame sync sample.

19.7.1.4 TDMx Receive Frame Parameters (TDMxRFP)

TDM*x*RFP TDM*x* Receive Frame Parameters Offset 0x3FE0

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—					RCDBL			—		RCS				RT1	RUBM
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RFP defines the TDM*x* receive frame parameters.

Table 19-13. TDM*x*RFP Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
RNCF 23–16	0	**Receive Number of Channels in a TDM Frame** Specifies the total number of channels that are received in the TDM modules. One TDM frame can contain 2–256 channels at a granularity of two. **Notes:** **1.** RNCF[8-15] = (number of channels that received on one active link) × (number of active data links) – 1, the number of active data links is specified in the RTSAL field. **2.** If RCDBL field is clear, then the minimum number of channels is limit. The minimum receive number of channels is 128 / (receive channel size) + 2. For example, if the channel size is 4 bits, then the receive TDM frame should contain at least 34 channels. **Table 19-14** describes the RNCF valid value as a function of the RTSAL field (Receive and Transmit Sharing and Active Links). For details, see **Section 19.2.5**.	0x00 Reserved. 0x01 2 received channels. 0x02 Reserved. 0x03 4 received channels. 0x04 Reserved. . . . 0xFD 254 received channels. 0xFE Reserved. 0xFF 256 received channels. **Note:** The even values are reserved.
— 15–11	0	Reserved. Write to zero for future compatibility.	
RCDBL 10–8	0	**Receive Channel Data Bits Latency** Defines the maximum amount of receive channel bits stored in the TDM local memory before they are transferred for processing. RCDBL determines the maximum data latency in the following way: Maximum data latency= (RCDBL)/RCS × (receive frame duration). For details, see **Section 19.2.6**. **Notes:** **1.** The maximum data latency is the latency at the worst case when the bus is very loaded. Typically the latency it much smaller. **2.** RCS is field at RFP register defines the channel size. **3.** The minimum number of receive channel is limited if the RCDBL field is clear.The minimum receive number of channels is 128/(receive channel size) + 2.	000 Maximum 64 channel bits. 001 Maximum 128 channel bits. 010 Maximum 256 channel bits. 011 Maximum 512 channel bits. 100 Maximum 1024 channel bits. 101 Maximum 2048 channel bits. 110 Reserved. 111 Reserved.
— 7–6	0	Reserved. Write to zero for future compatibility.	
RCS 5–2	0	**Receive Channel Size** Determines the receiver channel size – 1. For details, see **Section 19.2.6**.	0000 Reserved. 0001 The receiver channel size is 2 bits. 0010 Reserved. 0011 The receiver channel size is 4 bits. 0100– 0110 Reserved. 0111 Receiver channel size is 8 bits. 1000– 1110 Reserved. 1111 Receiver channel size is 16 bits.

Table 19-13. TDM*x*RFP Bit Descriptions (Continued)

Name	Reset	Description	Settings
RT1 1	0	**Receive T1 frame** Determines whether the receive frame is T1 frame or non T1. **Note:** In T1 mode the channel size must be 8 bits (RCS = 0x7) and the number of channels must be 24 × (number of links). For example, if the number of link is 2 (RTSAL[1–0] = 01), the number of channels should be 48 (RNCF = 0x2F). For details, see **Section 19.2**.	0 The receive frame is non T1 frame. 1 The receive frame is T1 frame.
RUBM 0	0	**Receive Unified Buffer Mode** Indicates that all the received data is directed to one buffer. When RUBM is set, the number of active links must be 1 (RTSAL = 0b0000 or RTSAL = 0100). The channel parameters of all active channels are located in the TDMxRCPR0 (See page 19-62). For details, see **Section 19.2.6.4**. **Note:** When this bit is set, the TDMxRIR[RRDO] bit should be cleared.	0 Each channel is written to a different data buffer in the internal MBus. 1 All the channels are written to the same data buffer in the internal MBus.

Table 19-14 describes the RNCF valid value as a function of the RTSAL[0–1] field.

Table 19-14. RNCF[7–0] Valid Values

RTSAL[1–0]	Number of Active Links	RNCF[7–0] Suffix	Valid Value of RNCF
00	1	xxxxxxx1	The total number of channels must have a granularity of two.
01	2	xxxxxx11	The total number of channels must have a granularity of four.
10	Reserved.		
11	4	xxxxx111	The total number of channels must have a granularity of eight.

19.7.1.5 TDMx Transmit Frame Parameters (TDMxTFP)

TDM*x*TFP TDM*x* Transmit Frame Parameters Offset 0x3FD8

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—					TCDBL			—		TCS				TT1	TUBM
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TFP defines the TDM*x* transmit frame parameters.

Table 19-15. TDM*x*TFP Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
TNCF 23–16	0	**Transmit Number of Channels in a TDM Frame** Specifies the total number of channels that are transmitted in the TDM modules. One TDM frame contains 2–256 channels. **Notes:** **1.** TNCF[8–15] = (number of channels that transmit on one active link) × (number of active data links) – 1. the number of active data links is specified in the RTSAL field. **2.** If TCDBL field is cleared, the minimum number of channels is limit. The minimum transmit number of channels is 128 / (transmit channel size) + 2. for example if the transmit channel size is 16 bits then the transmit TDM frame should contain at least 10 channels. The number of active data links is specified in the RTSAL field. **Table 19-16** describes the TNCF valid value as a function of the TDMxGIR[RTSAL] field (Receive and Transmit Sharing and Active Links). For details, see **Section 19.2.5**.	0x00 Reserved. 0x01 2 Transmit channels. 0x02 Reserved. 0x03 4 Transmit channels. 0x04 Reserved. . . . 0xFE Reserved. 0xFF 256 Transmit channels. The even values are reserved.
— 15–11	0	Reserved. Write to zero for future compatibility.	

Table 19-15. TDM*x*TFP Bit Descriptions (Continued)

Name	Reset	Description	Settings
TCDBL 10–8	0	**Transmit Channel Data Bits Latency** Defines the maximum transmit channel bits that can be stored in the TDM local memory before it transfers output. TCDBL determines the maximum data latency in the following way: Maximum data latency = (TCDBL) / TCS × (transmit frame duration). See **Section 19.2.6**. **Notes:** **1.** The maximum data latency is the latency at the worst case when the bus is very loaded. Typically, actual latency is much smaller. **2.** TDMxTFP[TCS] defines the transmit channel size. **3.** The minimum number of transmit channel is limit if the RCDBL field is clear. The minimum transmit number of channels is 128 / (transmit channel size) + 2.For example see **TNCF** field.	000 Maximum 64 channel bits. 001 Maximum 128 channel bits. 010 Maximum 256 channel bits. 011 Maximum 512 channel bits. 100 Maximum 1024 channel bits. 101 Maximum 2048 channel bits. 110 Reserved. 111 Reserved.
— 7–6	0	Reserved. Write to zero for future compatibility.	
TCS 5–2	0	**Transmit Channel Size** Determines the transmitter channel size – 1. For details, see **Section 19.2.5**.	0000 Reserved 0001 The transmitter channel size is 2 bits. 0010 Reserved 0011 The transmitter channel size is 4 bits. 0100 Reserved. 0101 Reserved. 0110 Reserved. 0111 The transmitter channel size is 8 bits. 1000–1110 Reserved 1111 The transmitter channel size is 16 bits.
TT1 1	0	**Transmit T1 Frame** Determines whether the TDM transmitter drives a T1 frame or non T1 frame. **Note:** In T1 mode, the channel size must be 8 (TCS = 0x7) and the number of channels 24 × (number of links). For example, if the number of links is 1 (RTSAL[1–0] = 00, the number of channels should be 24 (TNCF = 0x17). For details, see **Section 19.2**.	0 Transmit frame is non T1 frame. 1 Transmit frame is T1 frame.
TUBM 0	0	**Transmit Unified Buffer Mode** Indicates that all the transmit data is transferred from one buffer. When TUBM is set, the number of active links must be 1 (RTSAL = 0b0000 or 0b0100). The parameters for all transmit channels are located in TDMxTCPR0. For details, see **Section 19.2.6.4**. **Note:** When this bit is set, the TDMxTIR[TRDO] bit should be cleared.	0 Each channel is read from a different data buffer in the internal MBus. 1 All the channels are read from the same data buffer in the internal MBus.

Table 19-16 describes the TNCF valid value as function of the RTSAL field.

Table 19-16. TNCF[7–0] Valid Values

RTSAL[2–0]	Number of Active Links	TNCF[7–0] Suffix	Valid Value of TNCF
00	1	xxxxxxx1	The total number of channels must have a granularity of two.
01	2	xxxxxx11	The total number of channels must have a granularity of four.
10	Reserved		
11	4	xxxxx111	The total number of channels must have a granularity of eight.

19.7.1.6 TDMx Receive Data Buffer Size (TDMxRDBS)

TDM*x*RDBS TDM*x* Receive Data Buffer Size Offset 0x3FD0

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

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

TDM*x*RDBS defines the receive data buffers size in bytes.The buffers for A/µ-law channels are double size.

Table 19-17. TDM*x*RDBS Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
RDBS 23–0	0	**Receive Data Buffers Size** Receive data buffers size equals the receive data buffer size in bytes minus 1. The buffer size is aligned to 8 bytes, so bits 0–2 must be set to “111”. For details, see **Section 19.2.6.1**, *Data Buffer Size and A/m-law Channels,* on page 19-23. **Note:** The minimum buffer size is 16 bytes.	0x00000F to 0xFFFFFF

19.7.1.7 TDMx Transmit Data Buffer Size (TDMxTDBS)

TDM*x*TDBS TDM*x* Transmit Data Buffer Size Offset 0x3FC8

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

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

TDM*x*TDBS defines the transmit data buffer size in bytes. The buffers for A/μ channels are double size.

Table 19-18. TDM*x*TDBS Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
TDBS 23–0	0	**Transmit Data Buffers Size** Transmit data buffers size equals the transmit data buffer size in bytes minus 1. The buffer size is aligned to 8 bytes, so bits 29–31 must be set to “111.” For details, see **Section 19.2.6.1**. **Note:** The minimum buffer size is 16 bytes.	0x00000F to 0xFFFFFF

19.7.1.8 TDMx Receive Global Base Address

TDM*x*RGBA TDM*x* Receive Global Base Address Offset 0x3FC0

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

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

TDM*x*RGBA determines the 16 most significant bits global base address of the receiver data buffers. The actual address of each receive data buffer is RCDBA + (RGBA << 16). See **Section 19.2.6.2**.

Table 19-19. TDMxRGBA Bit Descriptions

Name	Reset	Description
— 31–16	0	Reserved. Write to zero for future compatibility.

Table 19-19. TDMxRGBA Bit Descriptions

Name	Reset	Description
RGBA 15–0	0	**Receive Global Base Address** Determines the global base address of the receiver data buffers. It is added to the channel data buffer address and to the current receive displacement to generate the actual data address.

19.7.1.9 TDMx Transmit Global Base Address

TDM*x*TGBA TDM*x* Transmit Global Base Address Offset 0x3FB8

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

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

TDM*x*TGBA determines the 16 most significant bits global base address of the transmitter data buffers. The actual address of each transmit data buffer is TCDBA + (TGBA << 16). See **Section 19.2.6.2**.

Table 19-20. TDM*x*TGBA Bit Descriptions

Name	Reset	Description
— 31–16	0	Reserved. Write to zero for future compatibility.
TGBA 15–0	0	**Transmit Global Base Address** Determines the global base address of the transmit data buffers. It is added to channel data buffer address and to the current transmit displacement to generate the actual address.

19.7.1.10 TDMx Transmit Force Register (TDMxTFR)

TDM*x*TFR TDM*x* Transmit Force Register 0x3F10

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

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

TDM*x*TFR sets the priority upgrade value for the transmit channels.

Table 19-21. TDMxTFR Bit Descriptions

Name	Reset	Description
— 31–16	0	Reserved. Write to zero for future compatibility.
PUV 15–0	0	**Priority Upgrade Value** Determines the threshold value for which the TDM transmitter upgrades its system priority. The value is used to avoid an underrun condition. If the value is 0x0000 (reset value), the priority is not upgraded and remains low. The maximum value, which causes the priority always to be high, depends on the value stored in TDMxTFP[TCDBL]. If TDMxTFP[TCDBL] = 0x000, achieve the maximum value is achieved by setting PUV = TDMxTFP[TNCF]. If TDMxTFP[TCDBL] is not 0x000, set the PUV using: PUV = 64/TDMxTFP[TCS] × TDMxTNB[TNB]. See page 19-51 for TDMxTFP details and page 19-69 for TDMxTNB details.

19.7.1.11 TDMx Receive Force Register (TDMxRFR)

TDM*x*RFR TDM*x* Receive Force Register0x3F18

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

TDM*x*RFR sets the priority upgrade value for the transmit channels.

Table 19-22. TDMxRFR Bit Descriptions

Name	Reset	Description
— 31–16	0	Reserved. Write to zero for future compatibility.
PUV 15–0	0	**Priority Upgrade Value** Determines the threshold value for which the TDM receiver upgrades its system priority. The value is used to avoid an overrun condition. If the value is 0x0000 (reset value), the priority is not upgraded and remains low. The maximum value, which causes the priority always to be high, depends on the value stored in TDMxRFPx[RCDBL]. If TDMxRFP[RCDBL] = 0x000, achieve the maximum value by setting PUV = TDMxRFP[RNCF]. If TDMxRFP[RCDBL] is not 0x000, set the PUV using PUV = 64/TDMxRFP[RCS] × RNBx[RNB]. See page 19-48 for TDMxRFP details and page 19-68 for TDMxRNB details.

19.7.1.12 TDMx Parity Control Register (TDMxPCR)

TDMxPCR TDMx Parity Control Register Offset 0x3F00

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

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

TDMxPCR controls the parity check operation.

Table 19-23. TDMxPCR Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.	
PC 2	0	**Parity Check** Enables the parity check mechanism. When the bit is set, you can write to TDMxRCPRn[27–24] or TDMxTCPRn[27–24].	0 Parity check is disabled. 1 Parity check is enabled
PIE 1	0	**Parity Interrupt Enable** Enables/disables the parity interrupt.	0 Parity interrupt is disabled. 1 Parity interrupt is enabled
PIL 0	0	**Parity Interrupt Level** Selects the trigger type for the parity interrupt.	0 Parity interrupt is edge-triggered. 1 Parity interrupt is level-triggered.

19.7.2 Control Registers

The following sections describe the TDM control registers.

19.7.2.1 TDMx Adaptation Control Register

TDMxACR TDMx Adaptation Control Register Offset 0x3FB0

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

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

TDMxACR controls the activation/deactivation of the TDMx adaptation machine. The propagation of the enable/disable to the adaptation machine is delayed because is clocked by the serial clock.

Table 19-25. TDMxACR Bit Descriptions

Name	Reset	Description	Settings
— 31–2	0	Reserved. Write to zero for future compatibility.	
AME 1	0	**Adaptation Machine Enable** Determines whether the adaptation machine is enabled or disabled.	0 Adaptation machine is disabled. 1 Adaptation machine is enabled
LTS 0	0	**Learn Transmit Sync** Determines whether the adaptation machine learns the transmit sync or the receive sync.	0 Adaptation machine learn the receive sync. 1 Adaptation machine learn the transmit sync.

19.7.2.2 TDMx Receive Control Register

TDM*x*RCR TDM*x* Receive Control Register Offset 0x3FA8

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

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

TDM*x*RCR controls the activation/deactivation of the TDM*x* Receiver. Receiver activation is valid only when the RENS bit is clear.

Table 19-26. TDM*x*RCR Bit Descriptions

Name	Reset	Description	Settings
— 31–1	0	Reserved. Write to zero for future compatibility.	
REN 0	0	**Receive Enable** Determines whether the receive TDM is enabled or disabled. **Note:** Setting this bit is the last step in initializing the receiver.	0 Receiver is disabled. 1 Receiver is enabled.

19.7.2.3 TDMx Transmit Control Register (TDMxTCR)

TDM*x*TCR TDM*x* Transmit Control Register Offset 0x3FA0

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

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

TDM*x*TCR controls the activation/deactivation of the TDM*x* Transmitter. Transmitter activation is valid only when the TENS bit is clear.

Table 19-27. TDM*x*TCR Bit Descriptions

Name	Reset	Description	Settings
— 31–1	0	Reserved. Write to zero for future compatibility.	
TEN 0	0	**Transmit Enable** Determines whether the transmit TDM is enabled or disabled. Setting this bit is the last step in initializing the transmitter.	0 Transmitter is disabled. 1 Transmitter is enabled.

19.7.2.4 TDMx Receive Data Buffer First Threshold (TDMxRDBFT)

TDM*x*RDBFT TDM*x* Receive Data Buffer First Threshold Offset 0x3F98

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

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

TDM*x*RDBFT determines the first threshold of the receive data buffers. When the receive buffers are filled up to the first threshold defined by this field—the TDM*x* Receive Data Buffers Displacement Register (TDM*x*RDBDR) = TDM*x* Receive Data Buffers First Threshold (TDM*x*RDBFT) + 8—the RFTE bit in the TDMx Receive Event Register (TDM*x* RER) is set. If the associated enable bit is also set, an interrupt is generated. This register can be updated at any time, even when the TDM*x* receiver is enabled. For details, see **Section 19.2.6.3**.

Table 19-28. TDM*x*RDBFT Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
RDBFT 23–0	0	**Receive Data Buffer First Threshold** Determines the location of the first threshold in the receive data buffers. The register value has a granularity of 8 bytes; that is, the three LSBits are always clear.	0x000000 to (RDBS – 7)

19.7.2.5 TDMx Transmit Data Buffer First Threshold

TDM*x*TDBFT TDM*x* Transmit Data Buffer First Threshold Offset 0x3F90

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

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

TDM*x*TDBFT determines the first threshold of the transmit data buffers. When the transmit buffers are emptied up to the first threshold defined by this field—the TDM*x* Transmit Data Buffers Displacement Register (TDM*x*TDBDR) = the TDM*x* Transmit Data Buffers First Threshold (TDM*x*TDBFT) + 8—the TFTE bit in the TDM*x* Transmit Event Register (TDM*x*TER) is set. If the associated enable bit is also set, an interrupt is generated. This register can be updated any time, even when the TDMx transmitter is enabled. For details, see **Section 19.2.6.3.**

Table 19-29. TDM*x*TDBFT Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
TDBFT 23–0	0	**Transmit Data Buffer First Threshold** Determines the location of the first threshold in the transmit data buffers. The register value has a granularity of eight bytes; that is, the three LSBits are always clear.	0x000000 to (TDBS – 7)

19.7.2.6 TDMx Receive Data Buffer Second Threshold

TDM*x*RDBST TDM*x* Receive Data Buffer Second Threshold Offset 0x3F88

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

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

TDM*x*RDBST determines the second threshold of the receive data buffers. When the receive buffers are filled up to the second threshold defined by this field—the Receive Data Buffers Displacement Register (TDM*x*RDBDR) = the Receive Data Buffers Second Threshold (TDM*x*RDBST) + 8—the RSTE bit in the TDMx Receive Event Register (TDM*x*RER) is set. If the associated enable bit is also set, an interrupt is generated. This register can be updated any time, even when the TDM*x* receiver is enabled. For details, see **Section 19.2.6.3**.

Table 19-30. TDM*x*RDBFT Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
RDBST 23–0	0	**Receive Data Buffer Second Threshold** Determines the location of the second threshold in the receive data buffers. The register value has a granularity of eight bytes; that is, the three LSBits are always clear.	0x000000 to (RDBS – 7)

19.7.2.7 TDMx Transmit Data Buffer Second Threshold

TDM*x*TDBST TDM*x* Transmit Data Buffer Second Threshold Offset 0x3F80

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

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

TDM*x*TDBST determines the second threshold of the transmit data buffers. When the transmit buffers are emptied up to the second threshold defined by this field—the Transmit Data Buffers Displacement Register (TDM*x*TDBDR) = the Transmit Data Buffers Second Threshold (TDM*x*TDBST) + 8—then the TSTE bit in the TDM*x* Transmit Register (TDM*x*TER) is set. If

the associated enable bit is also set, an interrupt is generated. This register can be updated at any time, even when the TDM*x* transmitter is enabled. For details, see **Section 19.2.6.3**.

Table 19-31. TDM*x*TDBST Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
TDBST 23–0	0	**Transmit Data Buffer Second Threshold** Determines the location of the second threshold in the transmit data buffers. The register value has a granularity of 8 bytes; that is, the three LSBits are always clear.	0x000000 to (TDBS – 7)

19.7.2.8 TDMx Receive Channel Parameter Register n

TDM*x*RCPRn TDM*x* Receive Channel Parameter Register n Offset 0x1000 + n*0x4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RACT	RCONV		—	Reserved				RCDBA							
Type	R/W															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

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

Note: The value n is reported in decimal. Convert the value to hexadecimal before multiplying to compute the offset value for a specific register location.

TDM*x*RCPRn determines the parameters for channel 0 to channel 255. The TDM*x*RCPRn[RACT] bit can be changed at any time during the receiver operation. All other fields can only be changed when TDM*x*RCPRn[RACT] is cleared. The read/write access to TDM*x*RCPRn registers can done only to 32 bits, write or read of byte or word is not valid. The register reset value is unknown.

Note: All TDM*x*RCPRn with an index number (n) less than or equal to the TDM*x*RFP[RNCF] bit (see page 19-48) should be valid when setting the corresponding TDM*x*RCR[REN] bit (see page 19-58).

The TDMxRCPRn registers are implemented using a compiled memory, and include support of parity mechanisms that allows detection and correction of one soft error using an interrupt (see TDMxPCR on page 19-57)**.**

Table 19-32. TDM*x*RCPRn Bit Descriptions

Name	Reset	Description	Settings
RACT 31	—	**Receive Channel Active** Set when the receive channel *n* is active.	0 The channel is non-active. 1 The channel is active.

Table 19-32. TDM*x*RCPRn Bit Descriptions (Continued)

Name	Reset	Description	Settings
RCONV 30–29	—	**Receive Channel Convert** Determines the type of the incoming channel n: Transparent, A-law, or µ-Law.	00 Receive channel n is a transparent channel. 01 Receive channel n is a µ-Law channel. 10 Receive channel n is an A-Law channel. 11 Reserved.
— 28	—	Reserved. Write to zero for future compatibility.	
— 27–24	—	These bits are used for parity protection.	
RCDBA 23–0	—	**Receive Channel Data Buffer Base Address** Determines the offset of the data buffer *n* base address from the Receive Global Base Address (RGBA).The RCDBA value should be 16 byte aligned; that is, the four LSB should be 0. For details, see **Section 19.2.6.2**.	0x000000–0xFFFFF0.

19.7.2.9 TDMx Transmit Channel Parameter Register n

TDM*x*TCPRn TDM*x* Transmit Channel Parameter Register n Offset 0x2800 + n*0x4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TACT	TCONV		—	Reserved				TCDBA							
Type	R/W															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

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

Note: The value n is reported in decimal. Convert the value to hexidecimal before multiplying to compute the offset value for a specific register location.

The TDM*x*TCPRn registers determine the parameters for channel 0 to channel 255. The registers can change any time during the transmitter operation. The TDM*x*TCPRn[TACT] bit can be changed at any time during the transmitter operation. All other fields can only be changed when TDM*x*TCPRn[TACT] is cleared. The read/write access to TDM*x*TCPRn registers can done only as a 32-bit access. A write or read of a byte or a word is not valid.The register reset value is indeterminate.

Note: All TDM*x*TCPRn with an index number (n) less than or equal to the TDM*x*TFP[TNCF] bit (see page 19-51) should be valid when setting the corresponding TDM*x*TCR[TEN] bit (see page 19-59).

The TDMxTCPRn registers are implemented using a compiled memory, and include support of a parity mechanism that allows detection and correction of one soft error using an interrupt (see TDMxPCR on page 19-57).

Table 19-33. TDM*x*TCPRn Bit Descriptions

Name	Reset	Description	Settings
TACT 31	—	**Transmit Channel Active** Set when the transmit channel n is active.	0 The channel is non-active. 1 The channel is active.
TCONV 30–29	—	**Transmit Channel Convert** Determines the type of the transmit channel n: Transparent, A-law, or µ-Law.	00 Transmit channel *n* is a transparent channel. 01 Transmit channel *n* is a µ-Law channel. 10 Transmit channel *n* is an A-Law channel. 11 Reserved.
— 28	—	Reserved. Write to zero for future compatibility.	
— 27–24	—	These bits are used for parity protection.	
TCDBA 23–0	—	**Transmit Channel Data Buffer Base Address** Determines the offset of the transmit data buffer *n* base address from the Transmit Global Base Address (TGBA). The TCDBA value should be 16 byte aligned; that is, the four LSB should be clear. For details, see **Section 19.2.6.2**.	0x000000–0xFFFFF0.

19.7.2.10 TDMx Receive Interrupt Enable Register (TDMXRIER)

TDM*x*RIER TDM*x* Receive Interrupt Enable Register Offset 0x3F78

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—												RSEEE	OLBEE	RFTEE	RSTEE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RIER has the same bit format as the TDM*x*RER registers. If an RIER bit is clear, the corresponding event in the TDM*x*RER registers is masked (see page 19-69).

Table 19-34. TDM*x*RIER Bit Descriptions

Name	Reset	Description	Settings
— 31–4	0	Reserved. Write to zero for future compatibility.	
RSEEE 3	0	**Receive Sync Error Event Enable** Enable assertion of the receive error interrupt when the Receive Sync Error (RSE) bit is set (see page 19-69).	0 Receive sync error is masked. 1 Receive sync error is enabled.

Table 19-34. TDM*x*RIER Bit Descriptions (Continued)

Name	Reset	Description	Settings
OLBEE 2	0	**Overrun Local Buffer Event Enable** Enable assertion of an interrupt when the Overrun Local Buffer Event (OLBE) bit is set (see page 19-69).	0 Overrun Local buffer event is masked. 1 Overrun Local buffer event is enabled.
RFTEE 1	0	**Receive First Threshold Event Enable** Enable assertion of the receive first threshold interrupt when the Receive First threshold Event (RFTE) bit is set (see page 19-69).	0 Receive first threshold interrupt is disabled. 1 Receive first threshold interrupt is enabled.
RSTEE 0	0	**Receive Second Threshold Event Enabled** Enable assertion of the receive second threshold interrupt when the Receive Second Threshold Event (RSTE) bit is set (see page 19-69).	0 Receive second threshold interrupt is disabled. 1 Receive second threshold interrupt is enabled

19.7.2.11 TDMx Transmit Interrupt Enable Register (TDMxTIER)

TDM*x*TIER TDM*x* Transmit Interrupt Enable Register Offset 0x3F70

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—												TSEIE	ULBEE	TFTEE	TSTEE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TIER has the same bit format as the TDM*x*TER registers. If a TDM*x*TIER bit is clear, the corresponding event in the TDM*x*TER is masked (see page 19-70).

Table 19-35. TDM*x*TIER Bit Descriptions

Name	Reset	Description	Settings
— 31–4	0	Reserved. Write to zero for future compatibility.	
TSEIE 3	0	**Transmit Sync Error Event Enabled** Enable assertion of the transmit error interrupt when the Transmit Sync Error (TSE) bit is set. See page 19-70.	0 Transmit sync error interrupt is disabled. 1 Transmit sync error interrupt is enabled.
ULBEE 2	0	**Underrun Local Buffer Event Enabled** Enable assertion of an interrupt when the Underrun Local Buffer Event (ULBE) bit is set. See page 19-70.	0 Underrun Local buffer event is masked. 1 Underrun Local buffer event is enabled.
TFTEE 1	0	**Transmit First Threshold Event Enabled** Enable assertion of the transmit first threshold interrupt when the Transmit First Threshold Event (TSTE) bit is set. See page 19-70.	0 Transmit first threshold interrupt is disabled. 1 Transmit first threshold interrupt is enabled.

Table 19-35. TDM*x*TIER Bit Descriptions

Name	Reset	Description	Settings
TSTEE 0	0	**Transmit Second Threshold Event Enabled** Enable assertion of the transmit second threshold interrupt when the Transmit second Threshold Event (TSTE) bit is set.	0 Transmit second threshold interrupt is disabled. 1 Transmit second threshold interrupt is enabled.

19.7.3 Status Registers

The following sections describe the TDM status registers.

19.7.3.1 TDMx Adaptation Sync Distance Register (TDMxASDR)

TDMxASDR TDMx Adaptation Sync Distance Register Offset 0x3F68

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

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

TDMxASDR indicates the number of receive/transmit bits between the last two consecutive receive/transmit sync events. The register value updates each time the TDMxASR[AMS] bit is set.

Table 19-36. TDMxASDR Bit Descriptions

Name	Reset	Description	Settings
— 31–11	0	Reserved. Write to zero for future compatibility.	
ASD 10–0	0	Adaptation Sync Distance Indicate the number of bits between the last two consecutive sync events. If the TDMxACR[LTS] bit is set, the ASD field indicates the number of transmit bits between the last two transmit sync events. If the LTS bit is clear, the value indicates the number of receive bits between the last two receive sync events.	0x000 The number of bits between the last two sync events is 1. 0x001 The number of bits between the last two sync events is 2. . . . 0x7FF The number of bits between the last two sync events is 2048.

19.7.3.2 TDMx Receive Data Buffers Displacement Register (TDMxRDBDR)

TDM*x*RDBDR TDM*x* Receive Data Buffers Displacement Register Offset 0x3F60

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

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

TDM*x*RDBDR points to the current displacement in the receive data buffers where the received data should be written by the TDM DMA. For details, see **Section 19.2.6.2**, *Data Buffer Address*, on page 19-24.

Table 19-37. TDM*x*RDBDR Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
RDBD 23–0	0	**Receive Data Buffer Displacement** Points to the current displacement of the received data in the data buffers. The value is unified to all the transparent channels and is doubled for A/μ law channels.	0 to (RDBS – 7) = Receive Data Buffer Size.

19.7.3.3 TDMx Transmit Data Buffer Displacement Register (TDMxTDBDR)

TDM*x*TDBDR TDM*x* Transmit Data Buffer Displacement Register Offset 0x3F58

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

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

TDM*x*TDBDR points to the current displacement in the transmit data buffers of the data that should be read by the TDM DMA. For details, see **Section 19.2.6.2**, *Data Buffer Address*, on page 19-24.

Table 19-38. TDMxTDBDR Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
TDBD 23–0	0	**Transmit Data Buffer Displacement** Points to the current displacement of the transmit data in the transmit data buffers. The value is unified to all the transparent channels and is doubled for A/μ law channels.	The register value can range from 0x000000 to the Transmit Data Buffer (TDBS – 7).

19.7.3.4 TDMx Receive Number of Buffers (TDMxRNB)

TDM*x*RNB TDM*x* Receive Number of Buffers Offset 0x3F50

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

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

TDM*x*RNB holds the number of buffers in the TDM receive local buffer. Using this register, you can calculate the location of all the bytes belonging to any channel in the TDM local memory.

Table 19-39. TDMxRNB Bit Descriptions

Name	Reset	Description	Settings
— 31–5	0	Reserved. Write to zero for future compatibility.	
RNB 4–0	0	**Receive Number of Buffers** Holds the number of buffers in the TDM receive local buffer. For details, see **Section 19.2.5**, *TDM Local Memory*, on page 19-22. **Notes:** **1.** The number of receive buffers equals RNB + 1. **2.** If TDMxRFP[RUBM] = 1, the RNB value is determined by the value of TDMxRFP[RCDBL].	0x00 1 buffer. 0x01 2 buffers. 0x03 4 buffers. 0x07 8 buffers. 0x0F 16 buffers. 0x1F 32 buffers. The other values are reserved.

19.7.3.5 TDMx Transmitter Number of Buffers (TDMxTNB)

TDM*x*TNB TDM*x* Transmitter Number of Buffers Offset 0x3F48

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

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

TDMxTNB holds the number of transmit buffers in the TDM transmit local buffer.

Table 19-40. TDMxTNB Bit Descriptions

Name	Reset	Description	Settings
— 31–5	0	Reserved. Write to zero for future compatibility.	
TNB 4–0	0	**Transmit Number of Buffers** Holds the number of buffers in the TDM transmit local buffer. **Notes:** **1.** The number of transmit buffers equals TNB + 1. **2.** If TDMxTFP[TUBM] = 1, the TNB value is determined by the value of TDMxTFP[TCDBL].	0x00 1 buffer. 0x01 2 buffers. 0x03 4 buffers. 0x07 8 buffers. 0x0F 16 buffers. 0x1F 32 buffers. The other values are reserved.

19.7.3.6 TDMx Receive Event Register (TDMxRER)

TDM*x*RER TDM*x* Receive Event Register Offset 0x3F40

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—											RFSTI	RSE	OLBE	RFTE	RSTE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RER contains the status of the receive data buffers and general receive events. The register can be read at any time. Bits 3–0 are cleared by writing ones to them; writing zero has no effect.

Table 19-41. TDM*x*RER Bit Descriptions

Name	Reset	Description	Settings
— 31–5	0	Reserved. Write to zero for future compatibility.	

Table 19-41. TDM*x*RER Bit Descriptions (Continued)

Name	Reset	Description	Settings
RFSTI 4	0	**Receive First or Second Threshold Interrupt Indication** Indicates whether the last receive threshold interrupt is the first or second threshold.	0 Second threshold interrupt. 1 First threshold interrupt.
RSE 3	0	**Receive Sync Error** Indicates whether a sync error has occurred. RSE is set when the receive frame synchronization is lost (the synchronization state change from SYNC to HUNT state) because that a frame sync arrive early or it not recognized at the expected position. During operation, this bit indicates errors on the receive signals of the TDM module. For details, see **Section 19.2.4.3**	0 Normal operation. No receive error has occurred. 1 Receive sync error has occurred.
OLBE 2	0	**Overrun Local Buffer Event** Indicates whether an overrun event has occurred in TDM local memory. This error should not occur during normal operation. It indicates that the TDM has not received enough bandwidth on the internal MBus and therefore cannot write the data into the destination memory (data buffer). For details, see **Section 19.2.6**.	0 No overrun event has occurred in the TDM local memory. 1 An overrun event has occurred in the TDM local memory.
RFTE 1	0	**Receive First Threshold Event** This field is set when the first thresholds of all the received data buffers are filled with received data. The first threshold pointer is determined by the Receive Data Buffer First Threshold field (RDBFT). For details, see **Section 19.2.6.3**.	0 No receive first threshold event has occurred. 1 A receive first threshold event has occurred.
RSTE 0	0	**Receive Second Threshold Event** This field is set when the second thresholds of all the receive data buffers are filled with received data. The second threshold pointer is determined by the Receive Data Buffer Second Threshold. (RDBST) field. For details, see **Section 19.2.6**	0 No receive second threshold event has occurred. 1 A receive second threshold event has occurred.

19.7.3.7 TDMx Transmit Event Register (TDMxTER)

TDM*x*TER TDM*x* Transmit Event Register Offset 0x3F38

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—											TFSTI	TSE	ULBE	TFTE	TSTE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TER contains the status of the transmit data buffers and general transmit events.The register can be read at any time. Bits 3–0 are cleared by writing ones to them; writing zero has no effect.

Table 19-42. TDM*x*TER Bit Descriptions

Name	Reset	Description	Settings
— 31–5	0	Reserved. Write to zero for future compatibility.	
TFSTI 4	0	**Transmit First or Second Threshold Interrupt Indication** Indicates whether the last receive threshold interrupt is the first or second threshold.	0 Second threshold interrupt. 1 First threshold interrupt.
TSE 3	0	**Transmit Sync Error** Indicates whether a sync error has occurred. TSE is set when the transmit frame synchronization is lost (the synchronization state change from SYNC to HUNT state) because that a transmit frame sync arrive early or it not recognized at the expected position. During operation, this bit indicates errors on the transmit signals of the TDM module. For details, see **Section 19.2.4.3**	0 Normal operation. No transmit sync error has occurred. 1 A transmit sync error has occurred.
ULBE 2	0	**Underrun Local Buffer Event** Indicates whether an underrun event has occurred in the TDM local buffer. This error should not occur during normal operation. It indicates that the TDM has not received enough bandwidth on the internal MBus and therefore cannot read the data from the data buffers to the TDM local memory. For details, see **Section 19.2.6**.	0 No underrun event has occurred in the TDM local memory. 1 An underrun event has occurred in the TDM local memory.
TFTE 1	0	**Transmit First Threshold Event** Indicates whether a first threshold event has occurred. TFTE is set when the first threshold of all the transmit data buffers is empty. The first threshold pointer is determined by the Transmit First Threshold Register (TDMxTFTR). For details, see **Section 19.2.6.3**.	0 No transmit first threshold event has occurred. 1 A transmit first threshold event has occurred.
TSTE 0	0	**Transmit Second Threshold Event** Indicates whether a transmit second threshold event has occurred. TSTE is set when the second threshold of all the transmit data buffers is empty. The second threshold pointer is determined by the transmit Second Threshold Register (TDMxTSTR). For details, see **Section 19.2.6.3**.	0 No transmit second threshold event has occurred. 1 A transmit second threshold event has occurred.

19.7.3.8 TDMx Adaptation Status Register (TDMxASR)

TDM*x*ASR TDMx Adaptation Status Register Offset 0x3F30

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

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

TDMxASR contains the status of the adaptation machine. The register can be read at any time. Bits are cleared by writing ones to them; writing zero has no effect.

Name	Reset	Description	Settings
— 31–1	0	Reserved. Write to zero for future compatibility.	
AMS 0	0	**Adaptation Machine Status** Indicates the status of the adaptation machine. If the bit is set, new sync arrive and the Adaptation Sync Distance Register (TDMxASDR) loads the new value.	0 No sync arrives and TDMxASDR does not contain a new value. 1 New sync arrives and the TDMxASDR register contains a new value that the SC3400 core should read.

19.7.3.9 TDMx Receive Status Register (TDMxRSR)

TDM*x*RSR TDM*x* Receive Status Register Offset 0x3F28

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—													RSSS		RENS
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RSR contains the receiver statues. It indicates whether the receiver is synchronized on the receive sync and the receiver is enabled or disabled.

Table 19-43. TDM*x*RSR Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.	
RSSS 2–1	0	**Receive Sync Synchronization Status** Indicates the status of the receive sync synchronization. When the synchronization state is SYNC, the serial part synchronized on the received sync and the received data transfer to the buffer in main memory for processing. **Note:** For details, see **Section 19.2.4.3**.	00 HUNT state. 01 WAIT state. 11 PRESYNC state. 10 SYNC state.
RENS 0	0	**Receive Enable Status** Indicates whether all the receiver parts are enabled/disabled. The propagation of the enable/disable may be delayed because of the different clocks domains.	0 The receiver machine is disabled. 1 The receiver machine is enabled.

19.7.3.10 TDMx Transmit Status Register (TDMxTSR)

TDM*x*TSR TDM*x* Transmit Status Register Offset 0x3F20

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—													TSSS		TENS
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TSR contains the status of the transmitter. It indicates whether the transmitter is synchronized on the transmit sync and whether it is enabled or disabled.

Table 19-44. TDM*x*TSR Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.	
TSSS 2–1	0	**Transmit Sync Synchronization Status** Indicates the transmit sync synchronization status. When the synchronization state is SYNC, the serial part is synchronized on the transmit sync and new transit data is driven out. For details, see **Section 19.2.4.3**.	00 HUNT state. 01 WAIT state. 11 PRESYNC state. 10 SYNC state.
TENS 0	0	**Transmit Enable Status** Indicates whether all the transmitter parts are enabled/disabled. The propagation of the enable/disable may be delayed because of the different clock domains.	0 The transmit machine is disabled. 1 The transmit machine is enabled.

19.7.3.11 TDMx Parity Error Register (TDMxPER)

TDM*x*PER TDM*x* Parity Error Register Offset 0x3F08

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

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

TDM*x*PER contains the parity error status information. It indicates whether a parity error has occurred, whether multiple errors have occurred, and the address of the last error. **Table 19-45** lists the bit field definitions.

Table 19-45. TDM*x*PER Bit Descriptions

Name	Reset	Description	Settings
— 31–18	0	Reserved. Write to zero for future compatibility.	
PME 17	0	**Parity Multiple Error** Indicates whether multiple parity errors occurred before clearing the PERR field. Clearing the PERR field clears this bit.	0 < 2 parity errors occurred. 1 2 or more parity errors occurred.
PERR 16	0	**Parity Error** Indicates whether a parity error occurred. The bit is cleared by writing a 1 to it. Writing a zero has no effect. The parity error is calculated in row resolution (2 channel parameters). Thus, even when accessing a channel parameter with no parity error, this bit indicates a parity error if the other channel has a parity error. Moreover, the parity error is indicated even if a channel is non-active.	0 No parity error occurred. 1 Parity error occurred.
— 15–10	0	Reserved. Write to zero for future compatibility.	
PEA 9–0	0	**Parity Error Address** Internal memory address where the last parity error occurred. The field is cleared when the TDMxPER[PERR] field is cleared. Receive parameter addresses fall in the range 0x100–0x17F and transmit parameter addresses fall in the range 0x280–0x2FF. Each address represents two channels. For example, address 0x100 indicates receive channels 0 and 1; address 0x280 indicates transmit channels 0 and 1.	

UART 20

The UART, also known as the serial communication interface (SCI), is a full-duplex port for serial communications with other devices. This interface uses two dedicated signals: transmit data (UTXD) and receive data (URXD) (see **Figure 20-1**). The external connections shared by these signals with GPIO[14,20] are available as general-purpose I/O (GPIO) signals when they are not configured for UART operation (see **Chapter 22**, *GPIO* for details).



Figure 20-1. UART Interface

The UART is accessible, via the MBus, to an external host or to each of the SC3400 cores. The UART generates one interrupt signal that connects to each SC3400 core so that each SC3400 core can service UART interrupts. For details on UART interrupt signal connectivity, refer to **Chapter 17**, *Interrupt Processing*. When accepting an interrupt request, an SC3400 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 SC3400 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 SC3400 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 MSC8144E and remote devices, including other MSC8144E devices. The UART transmitter and receiver operate independently, although they use the same baud-rate generator and same character length. An SC3400 core monitors the status of the UART, writes the data to be transmitted, and processes received data.

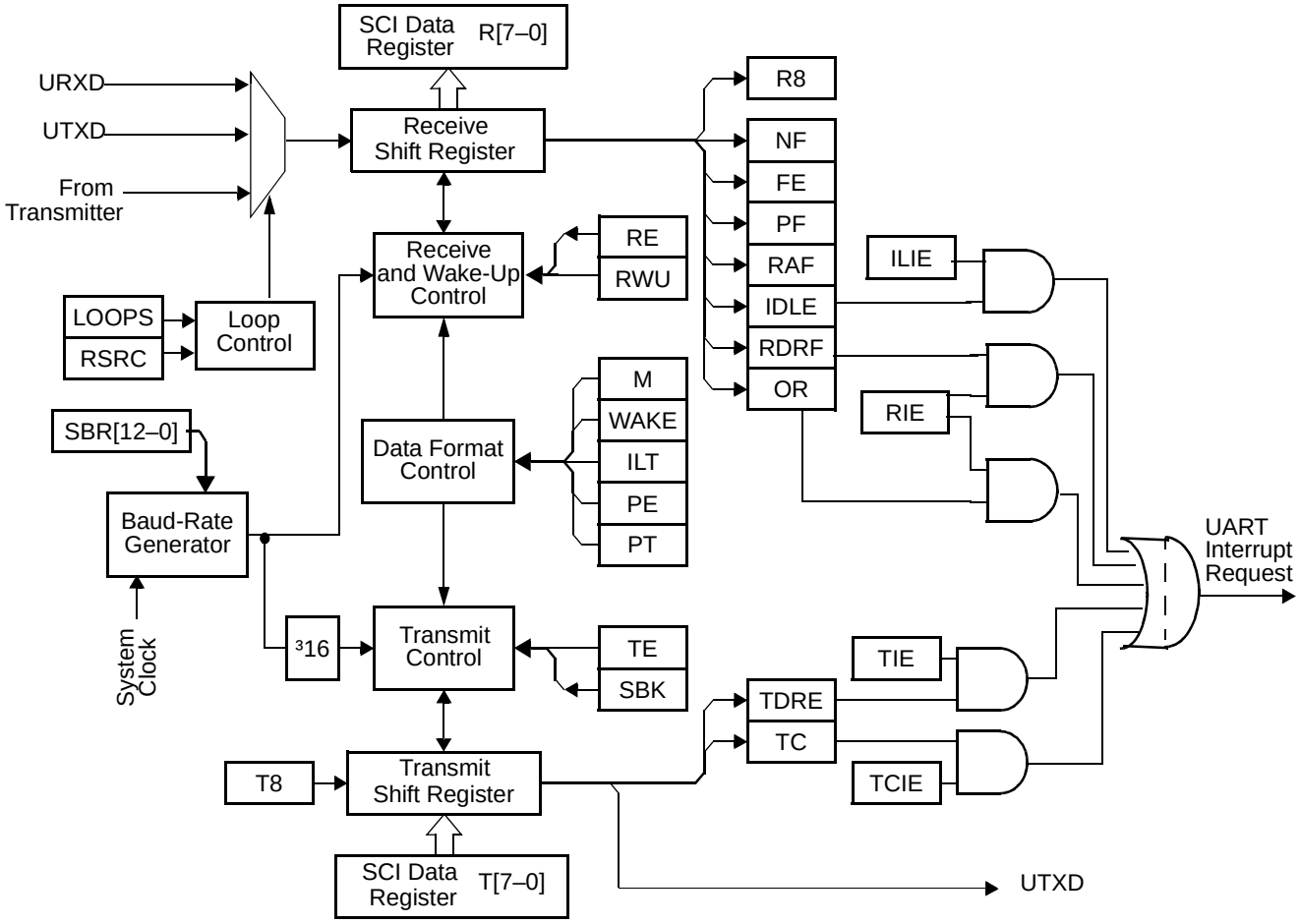
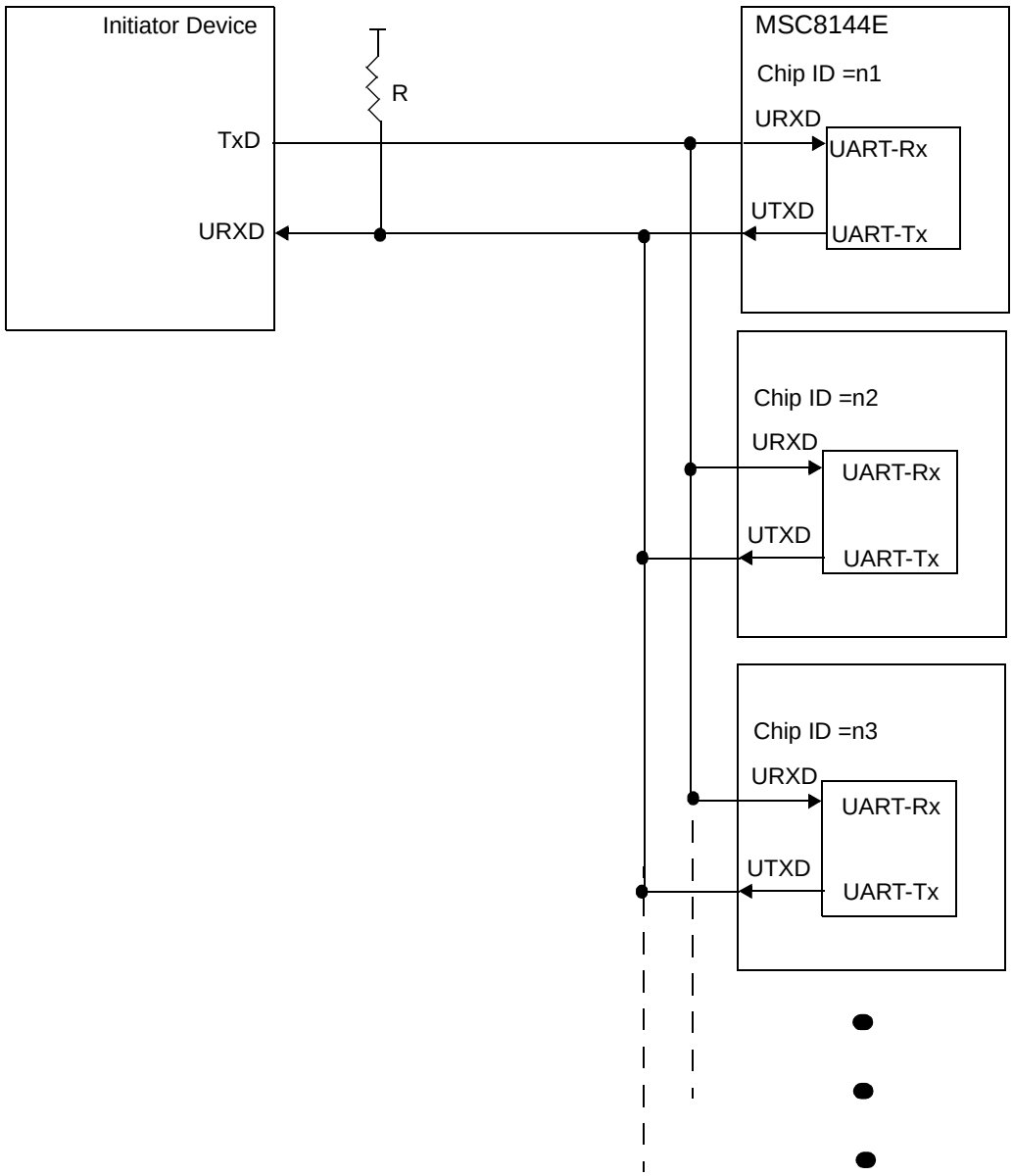


Figure 20-2. UART Block Diagram

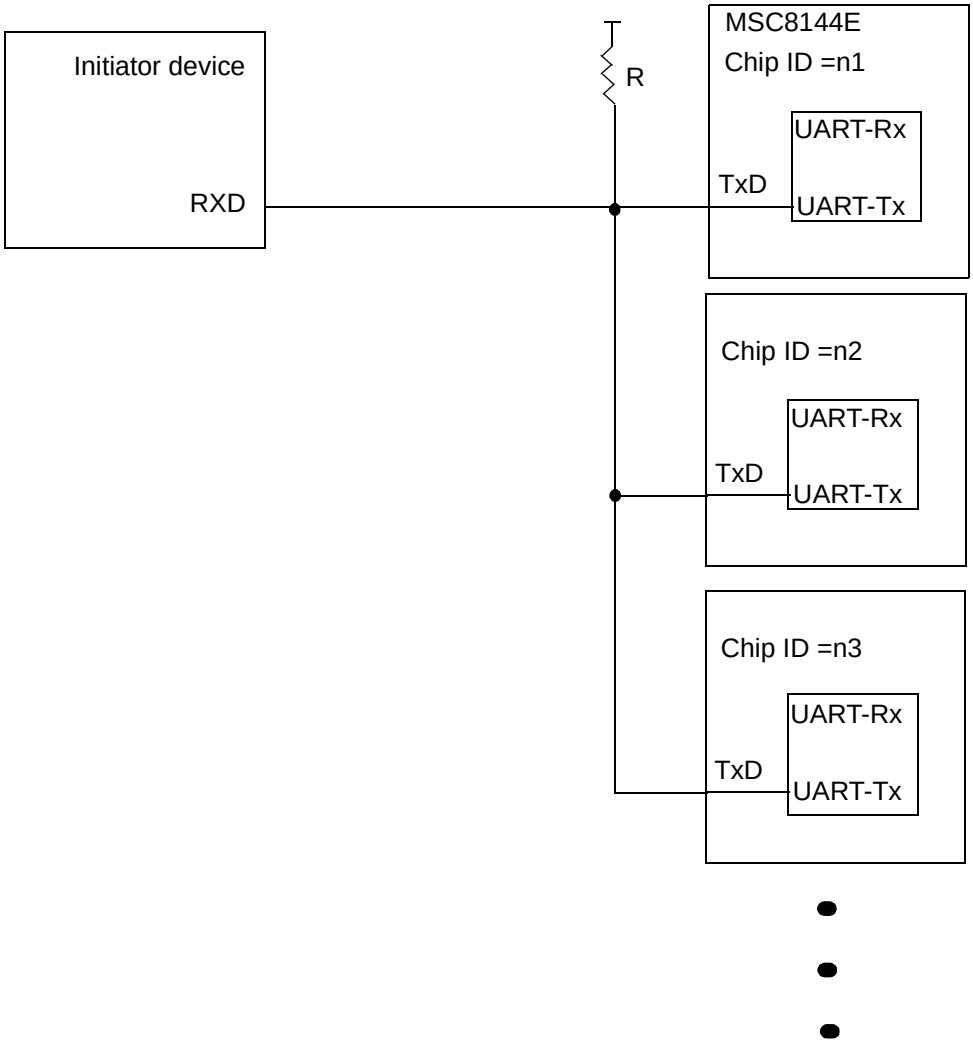
Figure 20-3 shows the full duplex UART system in which the MSC8144E UART transmits and receives simultaneously. A higher-level protocol should handle the full duplex communication to guarantee that no more than one target UART transmits to the URXD signal of the initiator at a given time. Receiver wake-up can obtain such a protocol (see **Section 21.2.7**, *Receiver Wake-Up*). The UART UTXD signal can be configured with full CMOS drive or with open-drain drive (see **Chapter 22**, *GPIO*). In both cases, the external pull-up resistor is needed to avoid floating input at the URXD 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 UTXD signal must be configured with open-drain drive (see **Chapter 22**, *GPIO*) and an external pull-up resistor. For details on single-wire, see **Section 21.4.2**, *Single-Wire Operation*.



Note: The RC value on the MultiPoint UTXD 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**.

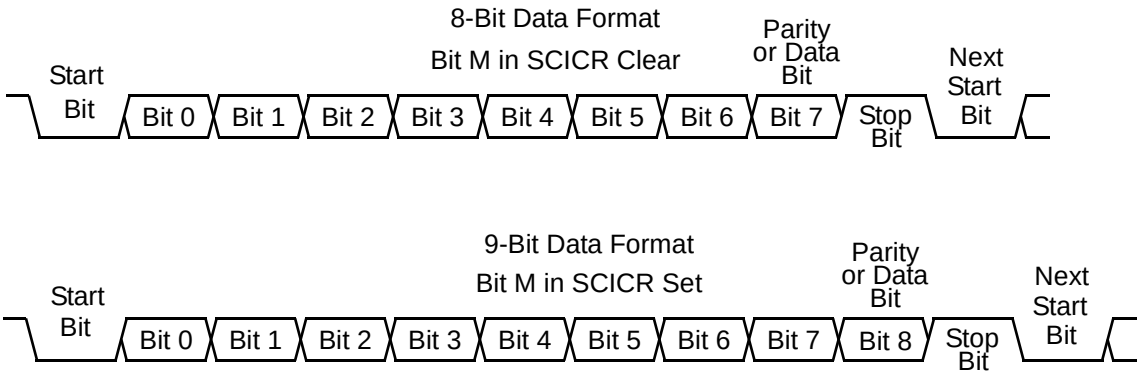


Figure 20-5. UART Data Formats

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 CLASS64 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 CLASS64 clock 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 CLASS64 clock frequency of 100 MHz, using the following formula:

$$UART\ baud\ rate = System\ clock / (16 \times SCIBR[12–0])$$

Table 20-3. Baud Rates (CLASS64 clock = 100 MHz)

Bits SBR	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (Percentage)
14	7,142,857	446,429	460,000	3.04
27	3,703,704	231,481	230,000	0.64
54	1,851,852	115,741	115,000	0.64
98	1,020,408	63,776	64,000	0.35
163	613,497	38,343.6	38,400	0.15
326	306,748	19,171.8	19,200	0.15
651	153,610	9600.6	9600	0.006
1302	76,804.9	4800.3	4800	0.006
2604	38,402.5	2400.15	2400	0.006
5208	19,201.2	1200.08	1200	0.006
Note: The error relates only to the first source error.				

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).

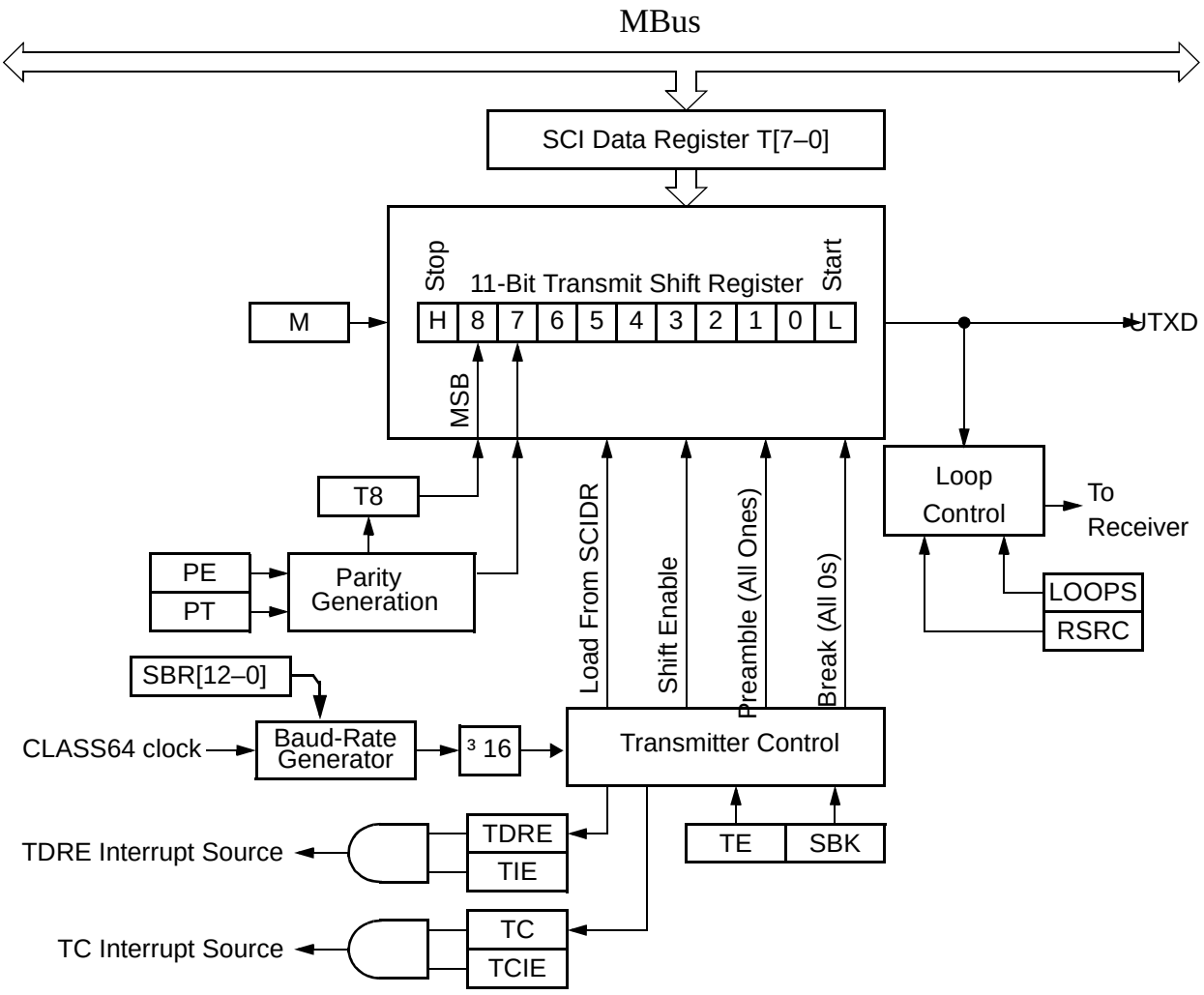


Figure 20-6. Transmitter Block Diagram

20.1.1 Character Transmission

To transmit data, one of the SC3400 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 UTXD 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 UTXD 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 GPIO14 for UART UTXD (see **Chapter 22**, *GPIO*):
      - Select the UART transmit signal for the GPIO14 external connection via the GPIO14 configuration registers.
      - Set the direction bit for the GPIO14 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 UTXD 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 until 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, UTXD goes to the idle condition, logic 1. When software clears the SCICR[TE] bit, the transmitter relinquishes control of UTXD.

Note: If SCIDDR[DDRTX] is set, UTXD is driven by a logic 0 (pulled down). Otherwise, if SCIDDR[DDRTX] is cleared, the UTXD signal is not driven. See **Chapter 22**, *GPIO* for details about configuring this signal.

If software clears SCICR[TE] while a transmission is in progress (TC = 0), the frame in the transmit shift register continues to shift out. Then the transmitter relinquishes control of UTXD 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, UTXD 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 UTXD. Setting SCICR[TE] after the stop bit appears on UTXD 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*.

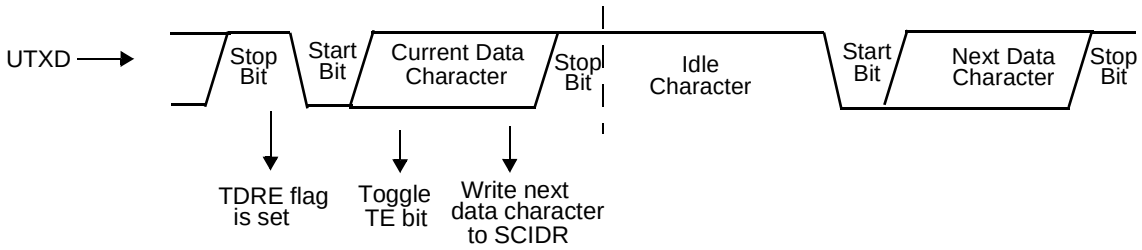


Figure 20-7. Queuing an Idle Character

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 URXD. 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 GPIO20 for UART URXD (see **Chapter 22**, *GPIO*):
   — Select the UART URXD signal as the external connection via the GPIO port 20 configuration registers.
   — Clear the direction bit for GPIO20 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 URXD 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.

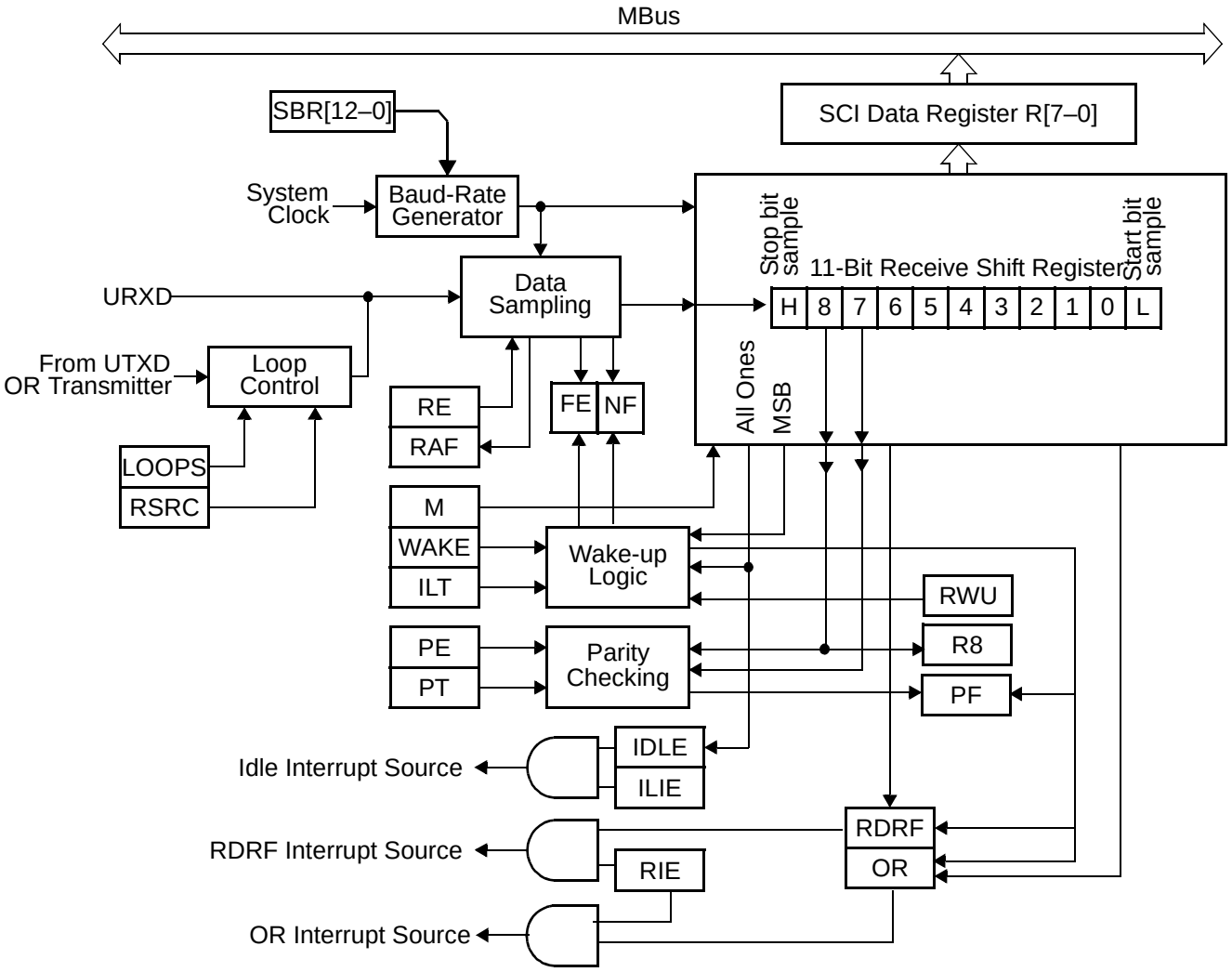


Figure 20-8. UART Receiver Block Diagram

20.2.2 Data Sampling

The receiver samples URXD 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.

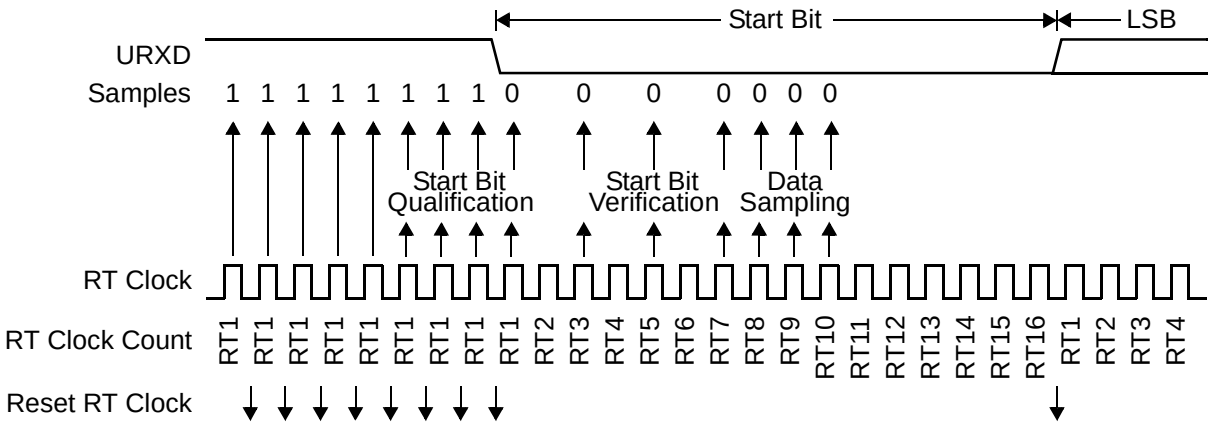


Figure 20-9. Receiver Data Sampling

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.

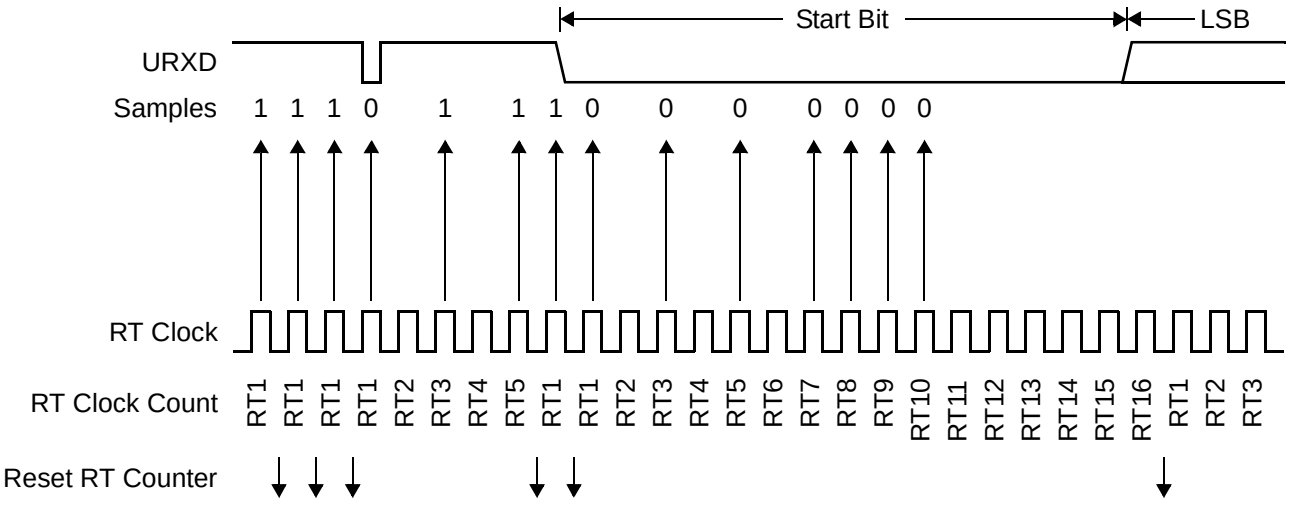


Figure 20-10. Start Bit Search Example 1

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.

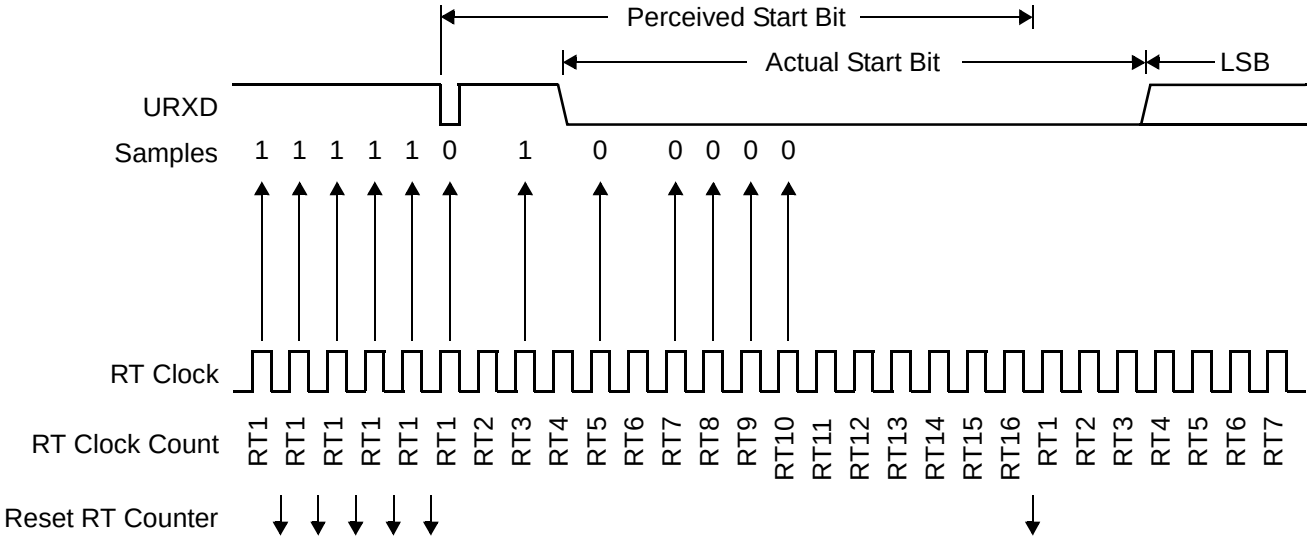


Figure 20-11. Start Bit Search Example 2

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.

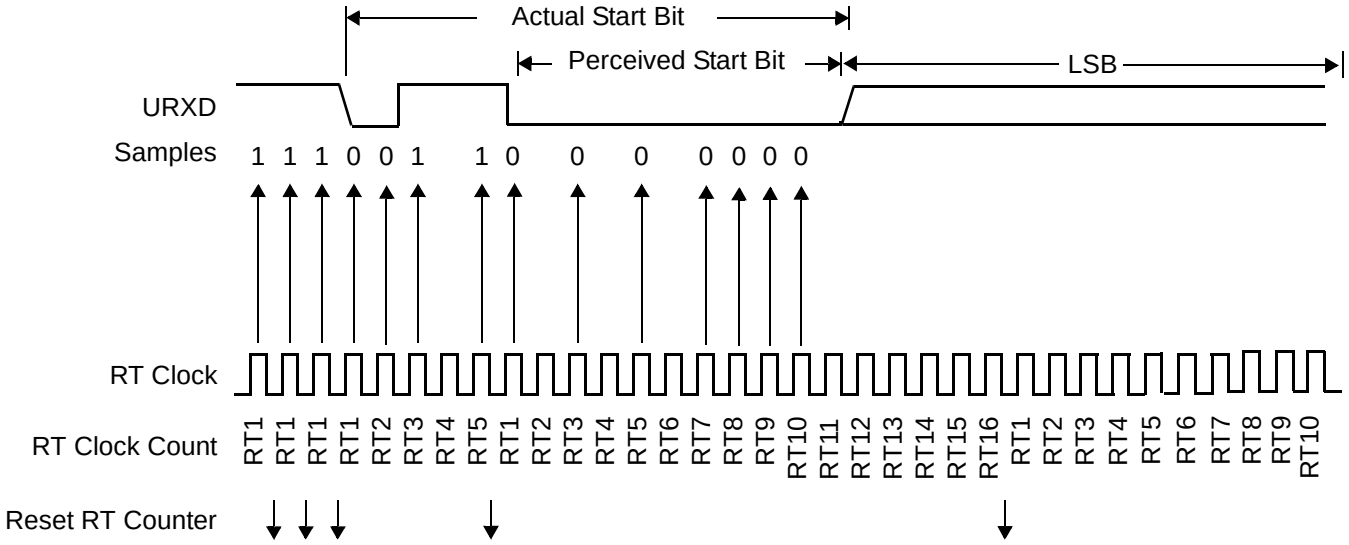


Figure 20-12. Start Bit Search Example 3

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.

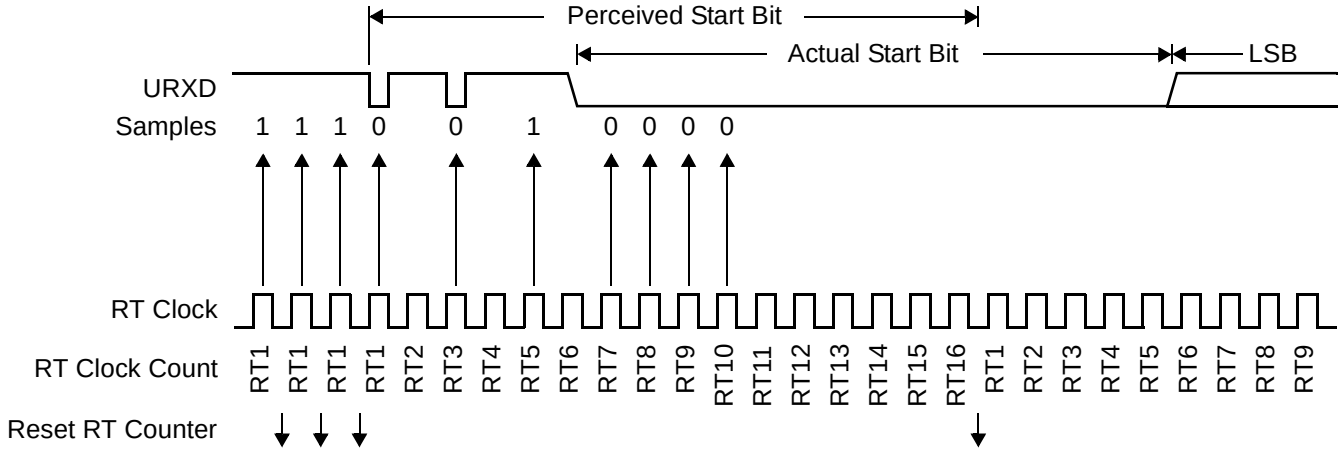


Figure 20-13. Start Bit Search Example 4

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.

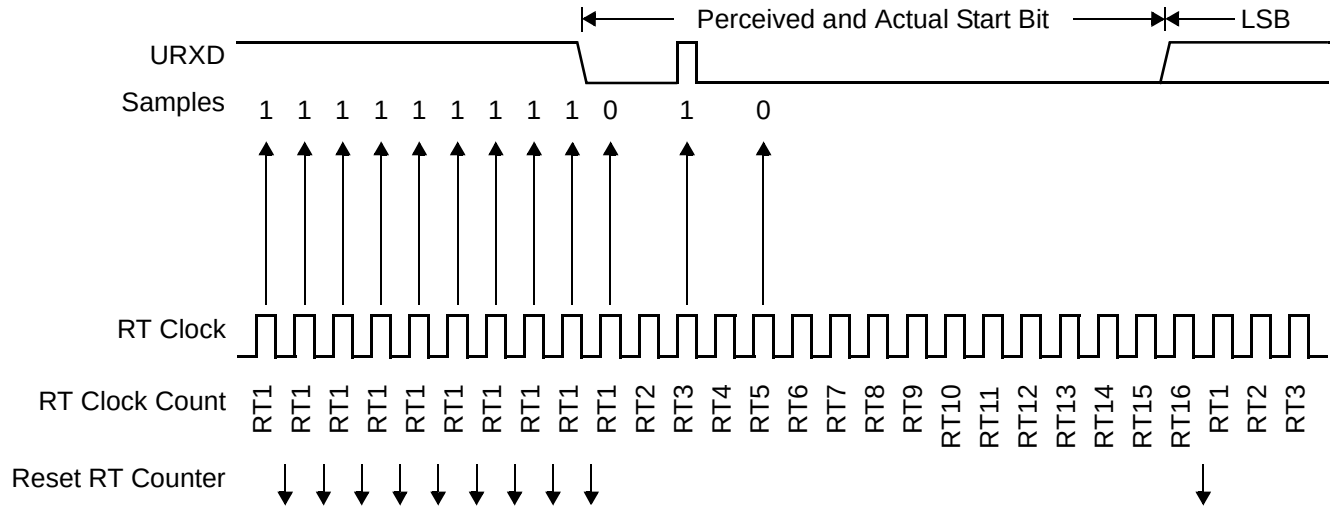


Figure 20-14. Start Bit Search Example 5

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.

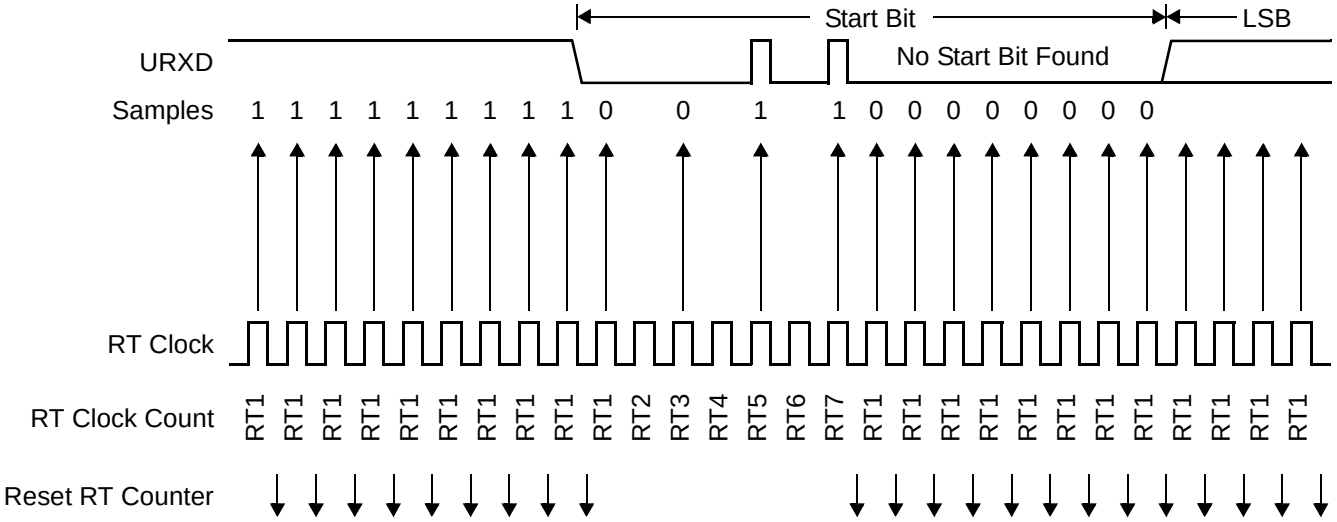


Figure 20-15. Start Bit Search Example 6

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.

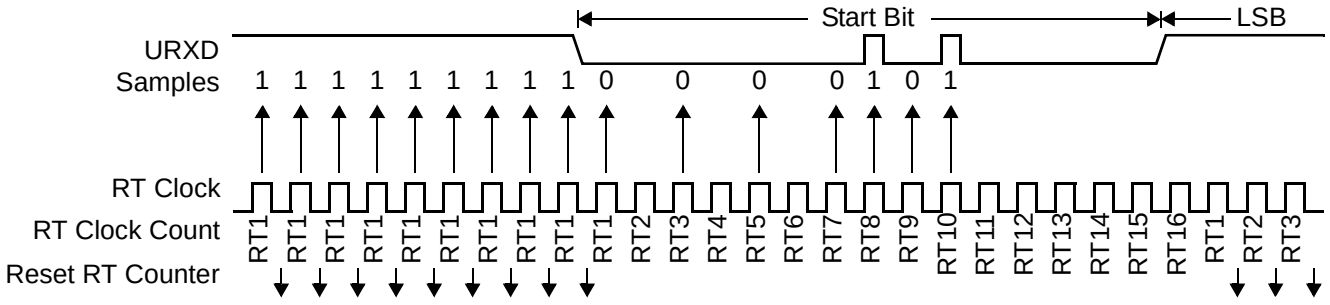


Figure 20-16. Start Bit Search Example 7

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.

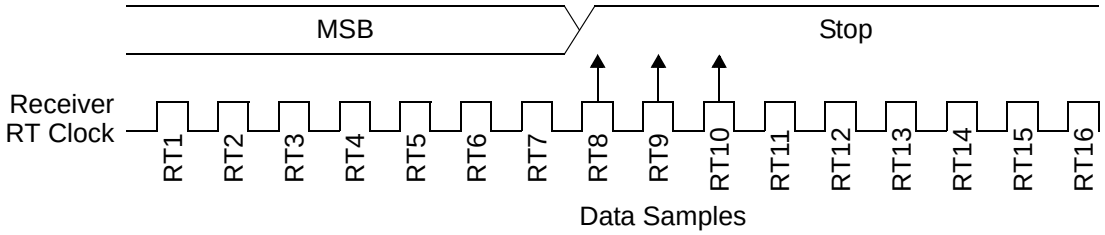


Figure 20-17. Slow Data

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.

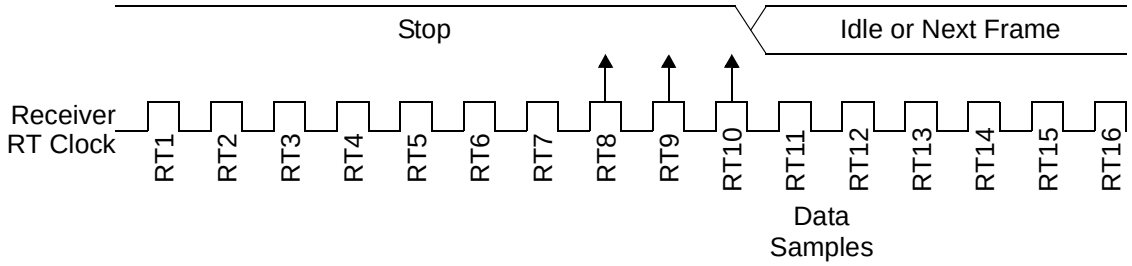


Figure 20-18. Fast Data

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 URXD (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 URXD.

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 URXD 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 URXD.

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 UTXD and URXD 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, URXD is disconnected from the UART and is available as a GPIO signal (see **Chapter 22**, *GPIO*). The UART uses UTXD for both receiving and transmitting data. Setting the data direction bit for UTXD, SCIDDR[DDRTX], configures UTXD as the output for transmitted data. Clearing the data direction bit, SCIDDR[DDRTX], disables the transmitter to drive UTXD.

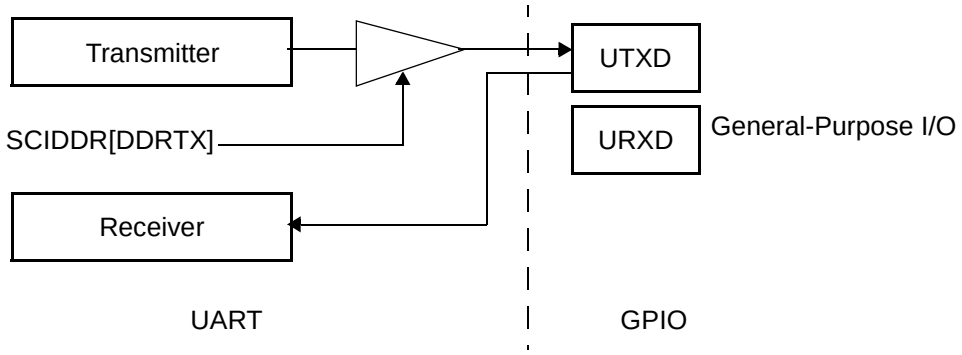


Figure 20-19. Single-Wire Operation (LOOPS = 1, RSRC = 1)

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 URXD to the receiver. Setting the SCICR[RSRC] bit connects the receiver input to the output of UTXD. Both the transmitter and receiver must be enabled (SCICR[TE] = 1 and SCICR[RE] = 1). You can configure UTXD (see **Chapter 22**, *GPIO*) for full CMOS drive or for open-drain drive. The configuration bit controls UTXD in both normal operation and single-wire operation. The configuration bit also allows the UTXD outputs to be tied together in a multiple-transmitter system, which allows the UTXD 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 URXD signal is disconnected from the external connection which then becomes available for use as a GPIO signal. Clearing the data direction bit of UTXD 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 URXD to the external output connection. Clearing the SCICR[RSRC] bit connects the transmitter output to the receiver input. Both the transmitter and receiver must be enabled (SCICR[TE] = 1 and SCICR[RE] = 1) for loop operation.

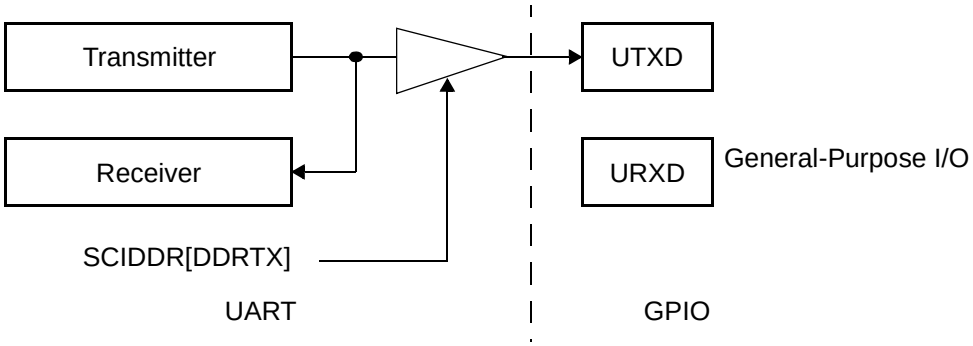


Figure 20-20. Loop Operation (LOOPS = 1, RSRC = 0)

20.4.4 Stop Mode

The UART stops its clock to provide reduced power consumption when the GCR1[UART_STC] bit is set (see **Section 8.2.1**, *General Configuration Register 1 (GCR1)*, on page 8-2). When the UART enters Stop mode, the states of the UART registers are unaffected. The UART registers cannot be accessed during Stop mode. When the UART_STC bit is cleared, UART operation resumes. Entering Stop mode during a transmission or reception results in invalid data. Therefore, disable the receiver and transmitter (SCICR[TE] = 0, SCICR[RE] = 0) before entering Stop mode.

20.4.5 Receiver Standby Mode

Refer to **Section 21.2.7**, *Receiver Wake-Up*.

20.5 Interrupt Operation

Table 20-7 lists the five interrupts generated by the UART to communicate with an SC3400 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: 0xFFF7F000.

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 = SCI CLASS64 clock/(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 URXD 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 UTXD is set and LOOPS = 1, the transmitter output drives UTXD. If the data direction bit is clear and LOOPS =1, the SCI transmitter does not drive UTXD.	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 UTXD. 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 URXD (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, UTXD is not driven by the SCI transmitter
1	0	1	Loop mode, UTXD is driven by the SCI transmitter
1	1	0	Single-wire mode UTXD acting as an input for the received data. The external connection that URXD shares can be configured as a GPIO.
1	1	1	Single-wire mode with UTXD acting as an output for the transmitted data. The transmitted data is also internally connected to the receiver input. The external connection that URXD shares can be configured as a GPIO.
Note: See **Chapter 22**, *GPIO* for details on configuring the signal multiplexing.			

20.6.3 SCI Status Register (SCISR)

SCISR SCI Status Register Offset 0x10

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF	—							RAF
Type	R								R/W							R
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SCISR can be read any time. A write has no meaning or effect.

Table 20-11. SCISR Bit Descriptions

Name	Reset	Description	Settings
— 31–16	0	Reserved. Write to zero for future compatibility.	
TDRE 15	1	**Transmit Data Register Empty Flag** Set when the transmit shift register receives a character from the SCI data register. When TDRE is 1, the transmit data register (SCIDR) is empty and can receive a new value to transmit. This flag can generate an interrupt request (refer to **Section 20.5**). Clear TDRE by reading TDRE and then writing to T[7–0] in the SCIDR.	1 Character transferred to transmit shift register; transmit data register empty. 0 No character transferred to transmit shift register.
TC 14	1	**Transmit Complete Flag** Set low when there is a transmission in progress or when a preamble or break character is loaded. TC is set high when the TDRE flag is set and no data, preamble, or break character is being transmitted. When TC is set, UTXD 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, UTXD 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, UTXD is driven by logic 0. 0 UTXD 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 UTXD signal is configured for UART operation.			

Timers 21

The MSC8144E device includes 3 types of timers:

- *Device-level timers*. Each MSC8144E device contains a total of 16 device timers. Each can be used by any of the DSP cores within MSC8144E as well as by an external host. See **Section 21.1** for details.
- *SC3400 DSP core platform timers*. Each of the SC3400 DSP core subsystems contain two timers used by the specific DSP core for any required operating system purpose. For details, see the *MSC8144 DSP Core Subsystem Reference Manual* available with a signed non-disclosure agreement. Contact you Freescale representative or distributor for details.
- *Software watchdog timers.* The MSC8144E device includes 5 software watchdog timers. Each of the software watchdog timers can be used by any of the cores within MSC8144E as well as by an external host. For details, see **Section 21.3**.

21.1 Device-Level Timers

There are four identical quad timer modules in the MSC8144E device. Each quad timer module contains four identical timer groups that serve as frequency dividers, clock generators, and event counters. Each 16-bit timer group contains a prescaler, a counter, a load register, a hold register, a capture register, two compare registers, and two status and control registers. The timers interface with the MSC8144E inputs/outputs as listed in **Table 21-1**. This document uses the term *timer* to refer to the four internal timers in each module and quad timer to refer to each of the timer modules. The term channel is used in register naming for clarity.

Table 21-1. Device-Level Timers Connectivity

Timer Module	Timer Channel	External Input	External Output
0	0	TIMER0	—
	1	TIMER1	TIMER0
	2	CLKIN	—
	3	TDM0TCLK	TIMER1
1	0	TIMER0	—
	1	TIMER2	—
	2	CLKIN	—
	3	TDM0TCLK	TIMER2
2	0	TIMER0	—
	1	TIMER3	—
	2	CLKIN	—
	3	TDM0TCLK	TIMER3

Table 21-1. Device-Level Timers Connectivity (Continued)

Timer Module	Timer Channel	External Input	External Output
3	0	TIMER0	—
	1	TIMER4	—
	2	CLKIN	—
	3	TDM0TCLK	TIMER4

Note: The external inputs list the external signal line connected to the specified timer input. The external outputs connect to the specified timer output. Most of the timer outputs do not connect to an external signal line. Even for cases in which a connection is indicated, the output is not valid unless the output is enabled by the TMRnSCTL[OEN] bit for the timer (see **Section 21.4.1.2**). Inputs and outputs for the TIMERn signal lines are multiplexed.

21.1.1 Features

Features of the timers include:

- Counters support the following operations:
  - Cascade.
  - Preloading.
  - Count once or continuously.
  - Share input pins.
  - Do capture and compare.
  - Count up or down.
- Count modulo is programmable.
- Maximum count rate is the CLASS64 clock rate when the timer input signals are not in use.
- Maximum count rate is half the CLASS64 clock rate when the timer input signals are in use.
- Each counter has a separate prescaler.

21.1.2 Timer Module Architecture

Each quad timer module contains four timers. The block diagram of one timer within a quad timer module is shown in **Figure 21-1**. As the figure shows, the primary clock selector contains a prescaler, primary clock multiplex, and an optional invert. It selects and optionally inverts a clock source for the primary clock. The primary clock can be selected from any of the following:

- Normal clocking:
  - CLASS64 clock.
  - CLASS64 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 21-19) and then set the TMRxSCTL[FORC] bit (page 21-19).

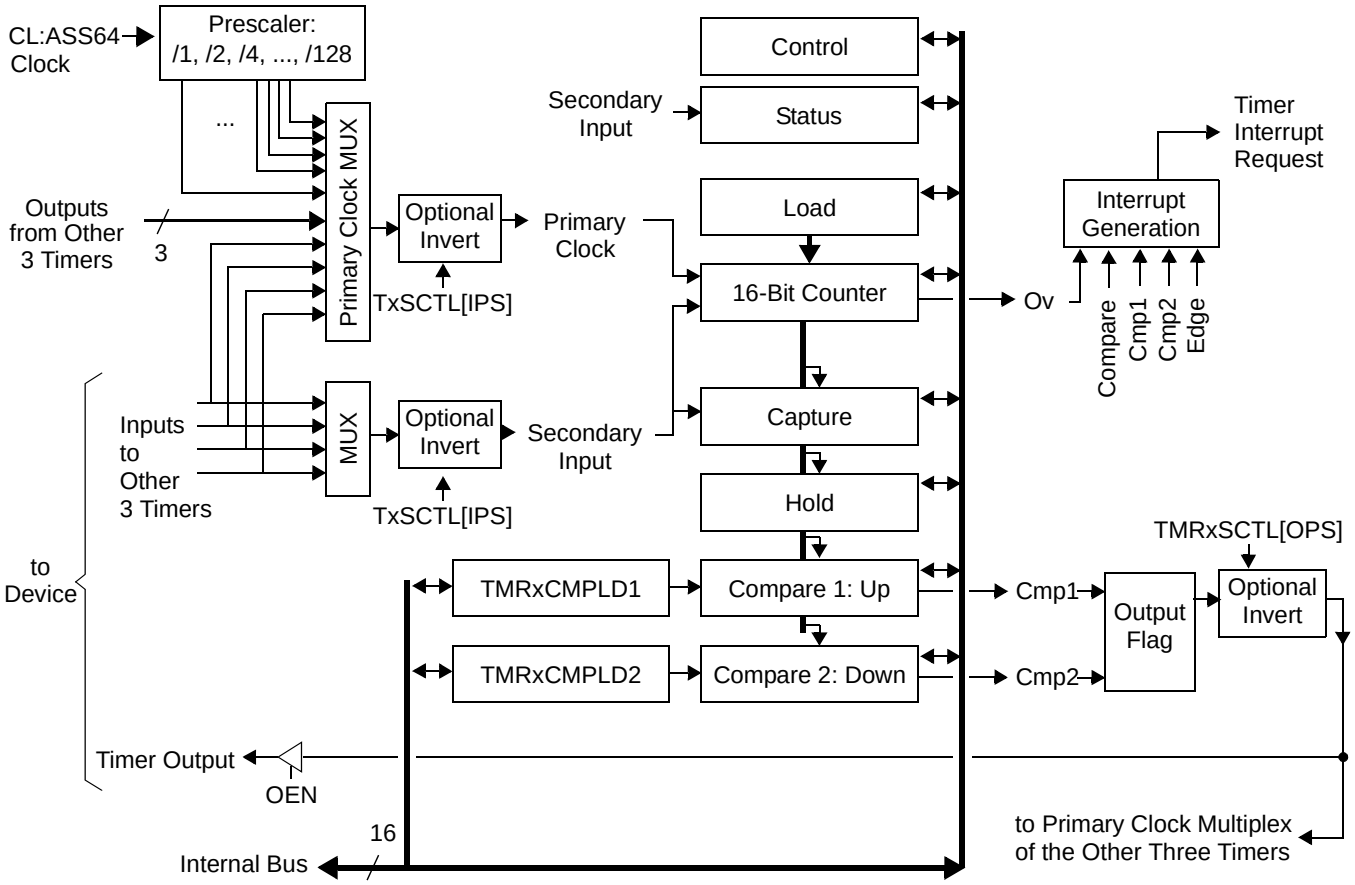


Figure 21-1. Timer Module Block Diagram — One of Four Timers

21.1.3 Setting Up Counters for Cascaded Operation

To create a counter larger than 16 bits, the first timer is programmed for the desired configuration and count mode (it is *not* programmed in Cascade mode). This first timer also programs the TMRxCTL[PCS] field to select the desired primary clock. All other timers in the cascade are simply programmed with their count mode set to Cascade mode (TMRxCTL[CM] = 111). They also program the TMRxCTL[PCS] field using the appropriate timer N output so that each can receive their clocks from the previous timer in the chain. The first timer in the chain must never be programmed in cascade mode. Also, the first timer in the chain must not choose one of the outputs from the timers for its primary clock. All timers in the cascade follow the counting mode of the first timer in the chain. In Cascade mode, a special high-speed signal path is used, bypassing the timer output flag logic to ensure that the cascaded channels operate as a single synchronous counter. You can connect timers using the other (non-cascade) timer modes and selecting the outputs of other timers as a clock source. In this case, the timers operate in a *ripple* mode, in which higher-order counters transition a clock later than in a purely synchronous design. This is *not* the typical use for cascaded counters.

21.1.3.1 Operation of the Cascaded Timer

If the first timer in a cascaded chain is counting up and it encounters a compare event, the timer connected to it is incremented. If the first timer in the chain is counting down and it encounters a compare event, the timer is decremented. You can correctly read all 16-bit portions of a cascaded timer as follows using the TMRxHOLD registers:

1. Read any 16-bit portion of the cascaded timer from its TMRxCNTR register. You can do this at any time.
2. When any TMRxCNTR register in the module is read, all other timers simultaneously load their values into their hold registers.
3. Read the 16-bit portions of all other timers in the cascade from their TMRxHOLD registers.

21.1.3.2 Cascading Restrictions

To ensure that there are no feedback loops in a cascade, there are restrictions on which timers can be cascaded. The timer with the lowest number must always be the first in the cascade, the timer with the second lowest number must be second, and so on. The timer with the highest number must always be last in the cascade. **Table 21-2** summarizes the cascading restrictions.

Table 21-2. Restrictions On Cascading Timers

Timer Number	Valid Cascade Inputs	Legal Values for Cascading using TMRxCTL[PCS]	Description
Timer 0	None	None	Timer 0 can only be the first timer in a cascaded timer. It cannot receive another timer's output for cascaded operation. Timer 0 must always be the first timer in the cascade.
Timer 1	Timer 0 output	0100: Timer 0	Timer 1 can be cascaded with Timer 0, with Timer 0 as the first timer in the chain.
Timer 2	Timer 0 output Timer 1 output	0100: Timer 0 0101: Timer 1	Timer 2 can be cascaded with Timer 0 or Timer 1 when Timer 2 is not the first timer in the cascade.
Timer 3	Timer 0 output Timer 1 output Timer 2 output	0100: Timer 0 0101: Timer 1 0110: Timer 2	Timer 3 can be cascaded with Timer 0, Timer 1, or Timer 3. Timer 3 must always be the last timer in a cascade.

21.1.4 Timer Operating Modes

The timer operates in two modes:

- Count the CLASS64 clock or external events via the timer input using the primary clock.
- Count the CLASS64 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 21-17). If a timer is programmed to count to a specific value and then stop, the TMRCTL[CM] bit is cleared when the count terminates. **Table 21-3** summarizes the counting modes.

Table 21-3. Summary of Timer Counting Modes

Counting Mode	CM Bits	Description	Primary Clock	Secondary Clock
Disabled	000	Timer not active.	—	—
Count	001	Counts the rising edges of the selected clock source (falling edges if TxSCTL[IPS] is set). This mode is useful for generating periodic interrupts for timing purposes or for counting external events.	Clock*	—
Dual-Edge Count	010	Counts both edges of a timer Input signal. This mode is useful for counting the changes in the external environment. When this mode is selected, TMRxCTL[PCS] must not be set to any value between 1000 and 1111; that is, it must not set to the input clock or any scaled version of the input clock.	Clock	—

Table 21-3. Summary of Timer Counting Modes

Counting Mode	CM Bits	Description	Primary Clock	Secondary Clock
Gated Count	011	Counts primary clock edges while the secondary input is high (low if TxSCTL[IPS] is set). This mode is used to time the duration of external events when the primary clock is set to the input clock and the secondary input is set to use one of the timer input signals. It can also be used to count the number of external events that occur on one of the timer input signals, set as the primary clock, while a second timer input signal, connected to the secondary Input signal, is asserted.	Clock	Gate*
Quadrature Count	100	Counts using quadrature encoded signals. The quadrature signals are square waves, 90 degrees out of phase. The decoding of quadrature signal provides both count and direction information. A timing diagram illustrating the basic operation of a quadrature incremental position encoder is provided in **Figure 21-2.**	Quadrature signal	Quadrature signal
Signed Count	101	Counts the primary clock source while a secondary input provides the count direction (up or down) for each recognized count.	Clock to count	Count direction
Triggered Count	110	Counts the primary clock source only after a rising edge is detected on the secondary input (falling edge if TxSCTL[IPS] is set). The counting continues until a compare event occurs or another positive input transition is detected. If a second input transition occurs before a terminal count is reached, counting stops. Subsequent odd-numbered edges of the secondary input restart the counting, and even numbered edges stop counting. This process continues until a compare event occurs.	Clock to count	Enable/disable timer*
Cascade Count	111	Cascades multiple timers. Cascade mode is used for creating timers larger than 16-bits. Up to four timers may be cascaded together to create a 64-bit wide timer. The Cascaded Timer mode is synchronous. See **Section 21.1.4.2**.	Clock to count	Triggers timer
Note: * This input can be inverted by the TxSCTL[IPS] bit.				

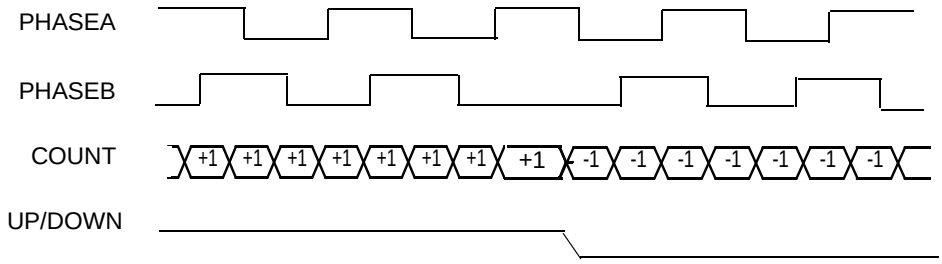


Figure 21-2. Quadrature Incremental Position Encoder

Other count modes derived as special cases of the modes described in **Table 21-3** are described in the remainder of this section.

21.1.4.1 One-Shot Mode

One-Shot mode is a variation on Triggered Count mode if the timer is set up as follows:

- TMRxCTL[CM] = 110 to count the rising and falling edges of the primary source (see **Table 21-5**).
- The Count Length bit, TMRxCTL[LEN] = 1.
- Output Flag mode, TMRxCTL[OFLM] = 101 to select set on compare, cleared on secondary input signal edge.
- The Count Once bit, TMRxCTL[ONCE] = 1 to count till a compare and then stop.

An external event causes the timer to count. When terminal count is reached, the timer output flag is asserted. This delayed output assertion can be used to provide timing delays.

21.1.4.2 Pulse Output Mode

In Pulse Output mode, a variation on Count mode, the timer outputs a stream of pulses with the same frequency as the selected clock source (cannot be the CLASS64 clock/1) if the timer is set up as follows:

- TMRxCTL[CM] = 001 to count the rising edges of the primary source. (see **Table 21-5***TMR[0–3]SCTL[0–3] Bit Descriptions*, on page 21>-19).
- The Output Flag Mode, TMRxCTL[OFLM], = 111 to enable gated clock output while the timer is active.
- The Count Once bit, TMRxCTL[ONCE], = 1 to count till a compare and then stop.

The number of output pulses is equal to the compare value minus the initial value. The primary count source must be set to one of the timer outputs for gated clock output mode.

21.1.4.3 Fixed Frequency PWM Mode

Fixed Frequency Pulse Width Modulated (PWM) mode is a subset of Count mode. The timer is set up as follows:

- TMRxCTL[CM] = 001 to count the rising edges of the primary source.
- The Count Length bit, TMRxCTL[LEN], = 0 so that the timer continues counting past the compare value (binary roll-over).
- The Count Once bit, TMRxCTL[ONCE], = 0 to count repeatedly.
- The Output Flag Mode, TMRxCTL[OFLM], = 110 so that the output flag is set when a compare occurs.

The timer output yields a PWM signal with:

- a frequency equal to the count clock frequency divided by 65,536.
- a pulse width duty cycle equal to the compare value divided by 65,536.

21.1.4.4 Variable Frequency PWM Mode

The timer output yields a PWM signal with a frequency and pulse width determined by the values programmed into the TMRxCMP1 and TMRxCMP2 registers and the input clock frequency if the timer is set up as follows:

- TMRxCTL[CM] = 001 TMRxCTL[CM] = 001 to count the rising edges of the primary source (see **Table 21-5**).
- The Count Length bit, TMRxCTL[LEN], = 1 so that the timer counts to the compare value and then reinitializes.
- The Count Once bit, TMRxCTL[ONCE], = 0 to count repeatedly.
- The Output Flag Mode, TMRxCTL[OFLM], = 100 to toggle the timer output flag using alternating compare registers.

This method of PWM generation has the advantage of allowing almost any desired PWM frequency and/or constant on or off periods. The TMRxCMPLD1 and TMRxCMPLD2 registers are especially useful for this mode because they give you time to calculate values for the next PWM cycle during the PWM current cycle.

To set up the timer to run in Variable Frequency PWM mode with compare preload, use the set up described here for the desired timer. During set-up, update the TMRxCTL register last because the timer starts counting if the count mode changes to any value other than 000. Set up the Timer Control (TMRxCTL) register bits as follows:

- Count Mode (CM) = 001 to count the rising edges of the primary source.
- Primary Count Source (PCS) = 1000 to specify the best granularity for waveform timing; prescaler CLASS64 clock/1.
- Secondary Count Source (SCS) = Any value because the bits are ignored in this mode.
- Count Once (ONCE) = 0 to count repeatedly.
- Count Length (LEN) = 1 so that the timer counts till it reaches a compare and then reinitializes the timer register.
- Direction (DIR) = Count up (0) or count down (1). The compare register values must be chosen carefully to account for roll-under and so on.
- External Initialization (EIN) = 0 so that another timer cannot force a reinitialization of this timer. However, you can set this bit if you need the functionality.
- Output Mode (OFLM) = 100 to toggle the timer output flag using alternating compare registers.

Set up the Timer Status and Control Register (TMRxSCTL) bits as follows:

- Output Polarity Select (OPS) = Your choice, true (0) or inverted (1).
- Output Enable (OEN) = 1 to enable the timer output to be put on an external pin. Set this bit as needed.
- Ensure that the rest of the TMRxSCTL bits are cleared. Interrupts are enabled in the Timer Comparator Status and Control Register (TMRxCOMSC) instead of in this register.

Set up the Timer Comparator Status and Control Register (TMRxCOMSC) bits as follows:

- Timer Compare 2 Interrupt Enable (TCF2EN) = 1 to allow an interrupt to be issued when TCF2 is set).
- Timer Compare1 Interrupt Enable (TCF1EN) = 0 so that an interrupt cannot be issued when TCF1 is set.
- Timer Compare 1 Interrupt Source (TCF1) = 0 to clear the timer compare 1 interrupt source flag. This bit is set when a successful comparison of the timer and the TMRxCMP1 register occurs.
- Timer Compare 2 Interrupt Source (TCF2) = 0 to clear the timer compare 2 interrupt source flag. This bit is set when a successful comparison of the timer and the TMRxCMP2 register occurs.
- Compare Load Control 1 (CL1) = 10 to load the compare register when TCF2 is set.
- Compare Load Control 2 (CL2) = 01 to load the compare register when TCF1 is set.

To service the TCF2 interrupts generated by the Timer, the interrupt controller must be configured to enable the interrupts for the timer being used. Additionally, you must write an interrupt service routine to do at least the following:

- Clear the TCF2 and TCF1 flags.
- Calculate and write new values for TMRxCMPLD1[15–0] and TMRxCMPLD2[15–0].

Figure 21-3 shows the timing for the compare preload cycle, which begins when a compare event on TMRxCMP2 causes TCF2 to be set. TMRxCMP1 is loaded with the value in the TMRxCMPLD1 one internal bus clock later. In addition, the timer asserts an interrupt, and the interrupt service routine executes while both comparator load registers are updated with new values. When TCF1 is set, TMRxCMP2 is loaded with the value of the CLV2 bits in TMRxCMPLD2. During the subsequent TCF2 event, TMRxCMP1 is loaded with the value of the TMRxCMPLD1[CLV1] bits. The cycle starts over again as an interrupt is asserted and the interrupt service routine clears TCF1 and TCF2 and calculates new values for TMRxCMPLD1 and TMRxCMPLD2.

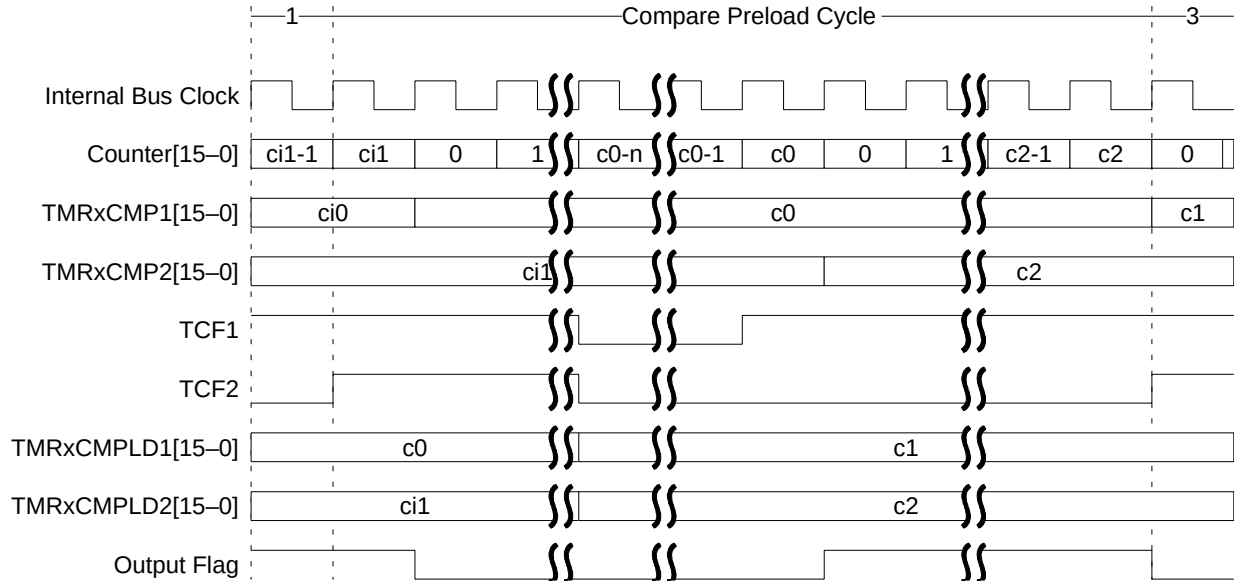


Figure 21-3. Compare Preload Timing

21.1.5 Timer Compare Functionality

The compare registers (TMRxCMP1 and TMRxCMP2) provide a bidirectional modulo count capability. TMRxCMP1 is used when the timer is counting *up*. Program it with the desired maximum count value or to 0xFFFF to indicate the maximum un signed value prior to roll-over. TMRxCMP2 is used when the timer is counting *down*. Program it with the maximum negative count value or to 0x0000 to indicate the minimum unsigned value prior to roll-under. The only exception occurs when the timer is operating with alternating compare registers.

When TMRxCTL[OFLM] = 100, alternating values of TMRxCMP1 and TMRxCMP2 are used to generate successful compares, and the output flag toggles while using alternating compare registers. For example, when TMRxCTL[OFLM] = 100, the timer is programmed to count upwards. It counts until the TMRxCMP1 value is reached, reinitializes, then counts until the TMRxCMP2 value is reached, reinitializes, then counts until the TMRxCMP1 value is reached, and so on. In this Variable Frequency PWM mode, the TMRxCMP2 value defines the desired pulse width of the *on-time*, and the TMRxCMP1 register defines the *off-time*. The Variable Frequency PWM mode is defined for positive counting only. See **Section 21.1.4.4**, *Variable Frequency PWM Mode*, on page 21-8.

Use caution when changing TMRxCMP1 and TMRxCMP2 while the timer is active. If the timer has already passed the new value, it counts to 0xFFFF or 0x0000, rolls over/under, and then begins counting toward the new value. The check is for Count = TMRxCMPx, not Count > = TMRxCMP1 or Count < = TMRxCMP2). Use of the preload registers addresses this problem.

21.1.5.1 Compare Preload Registers

The TMRxCMPLD1, TMRxCMPLD2 and TMRxCOMSC registers offer a high degree of flexibility for loading compare registers with user-defined values on different compare events. To ensure correct functionality, use the loading method described in this section.

The compare preload feature speeds updating of the compare registers. The compare preload feature allows you to calculate new compare values and store them into the comparator preload registers. When a compare event occurs, the new compare values in the comparator preload registers are directly written to the compare registers, eliminating the use of software to do this.

The compare preload feature is used in variable frequency PWM mode. See **Section 21.1.4.4**, *Variable Frequency PWM Mode*, on page 21-8. The TMRxCMP1 register determines the pulse width for the logic low part of the timer output, and TMRxCMP2 determines the pulse width for the logic high part of the timer output. The period of the waveform is determined by the TMRxCMP1 and TMRxCMP2 values and the frequency of the primary clock source.
See **Figure 21-4**.

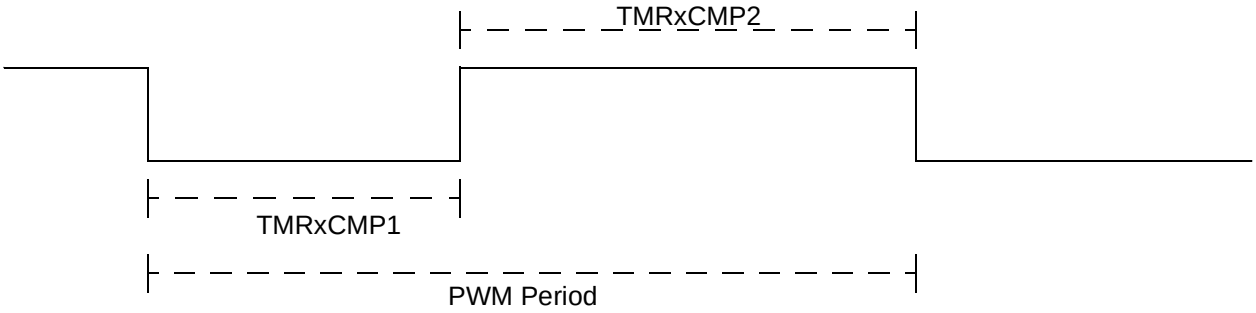


Figure 21-4. Variable PWM Waveform

To update the duty cycle or period of the waveform, update the TMRxCMP1 and TMRxCMP2 values using the compare preload feature.

21.1.5.2 Capture Register Use

The capture register, TMRxCAP (page 21-23), stores a copy of the timer value when an input edge (positive, negative, or both) on the secondary input signal is detected. The capture mode, programmable via TMRxSCTL[CM] (page 21-19), is one of the following:

- CM = 00. Disabled.
- CM = 01. Load the capture register on the *rising* edge of the signal.
- CM = 10. Load the capture register on the *falling* edge of the signal.
- CM = 11. Load the capture register on *either* edge of the signal.

When a capture event occurs, there are no further updates of TMRxCAP until the input edge flag (IEF) is cleared by writing a value of 0 to the TMRxSCTL[IEF] bit (page 21-19).

21.1.5.3 Broadcast from an Initiator Timer

Any timer can be assigned as an initiator. An initiator compares signal can be broadcast to the other timers within the module. The other timers can be configured as follows to reinitialize their timers and/or force their output to predetermined values when an initiator timer compare event occurs:

- Select one timer as the initiator timer by setting the TMRxSCTL[MSTR] bit.
- Program the other timers to perform an action when a compare event occurs on the initiator timer as follows:
  - The other timer is reinitialized if its TMRxCTL[EIN] bit is set.
  - The other timer forces its output flag signal if its TMRxSCTL[EEOF] bit is set.

21.1.6 Resets and Interrupts

The timers reset conditions are shown in **Chapter 5**, *Reset*. This reset forces all registers to their reset state and clears the output flag signal if it is asserted. The timer is turned off until the settings in the control register are changed. Each timer in a quad timer module can be programmed for interrupts. The available types of interrupts are as follows:

- Timer compare
- Timer compare 1
- Timer compare 2
- Timer overflow
- Timer input edge

Each of these different types is ORed together within each timer to generate a single interrupt request signal to the interrupt controller.

21.1.6.1 Timer Compare Interrupts

Interrupt requests are generated when a successful compare occurs between a timer and its compare registers while the Timer Compare Flag Interrupt Enable bit, TMRxSCTL[TCFIE], is set. These interrupt requests are cleared by writing a zero to the appropriate TMRxSCTL[TCF] bit. When a timer compare interrupt is set in the TMRxSCTL and the compare preload registers are available, one of the following two interrupts is also asserted:

- Timer compare 1 interrupt
- Timer compare 2 interrupt

Timer compare 1 interrupts are generated when a successful compare occurs between a timer and its TMRxCMP1 register while the Timer Compare 1 Interrupt Enable (TCF1EN) is set in the TMRxCOMSC register. These interrupts are cleared by writing a zero to the TMRxCOMSC[TCF1] bit. Timer compare 2 interrupts are generated when a successful compare

occurs between a timer and its TMRxCMP2 register while the Timer Compare 2 Interrupt Enable (TCF2EN) bit is set in the TMRxCOMSC register. These interrupts are cleared by writing a zero to the TCF2 bit in the TMRxCOMSC.

21.1.6.2 Timer Overflow Interrupts

Timer overflow interrupts are generated when a timer rolls over its maximum value while the TCFIE bit is set in the TMRxSCTL register. These interrupts are cleared by writing a zero to the Timer Overflow Flag (TOF) bit of the appropriate TMRxSCTL.

21.1.6.3 Timer Input Edge Interrupts

Timer input edge interrupts are generated by a transition of the input signal (either positive or negative, depending on TMRxSCTL[IPS] setting) while the Input Edge Flag Interrupt Enable (IEFIE) bit is set in the TMRxSCTL. These interrupts are cleared by writing a zero to the appropriate TMRxSCTL[IEF] bit.

21.2 SC3400 DSP Core Subsystem Timers

For a detailed description of the core subsystem timers, see the *MSC8144 DSP Core Subsystem Reference Manual.*

21.3 Software Watchdog Timers

Since the MSC8144E device contains four cores, there are total of five software watchdog timers (WDTs), one per core and one for an external host. However, you can allocate the WDTs in any manner to meet your system requirements. The five software WDTs are identical.

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 SC3400 core subsystems in the MSC8144E 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 non-maskable 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.

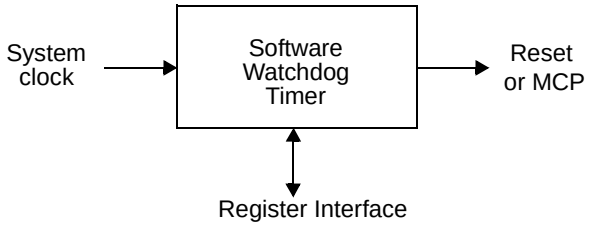


Figure 21-1. Software Watchdog Timer High Level Block Diagram

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 ~21.47 s maximum software time-out delay for 200 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 SWCRR[SWPR] bit that controls the divide-by-65536 of WDT counter.

  — Prescale mode (SWCRR[SWPR] = 1)

  The WDT clock is divided by 65536.

  — Non-prescale mode (SWCRR[SWPR] = 0)

  The WDT clock is not prescaled.

21.3.3 Software WDT Servicing

The software watchdog timer service sequence consists of the following two steps:

- Write 0x556C to the System Watchdog Service Register (SWSRR)
- Write 0xAA39 to SWSRR

The service sequence reloads the watchdog timer and the timing process begins again. If a value other than 0x556C or 0xAA39 is written to the SWSRR, the entire sequence must start over. Although the writes must occur in the correct order before a time-out, any number of instructions can be executed between the writes. This allows interrupts and exceptions to occur between the two writes when necessary. **Figure 21-5** shows a state diagram for the watchdog timer.

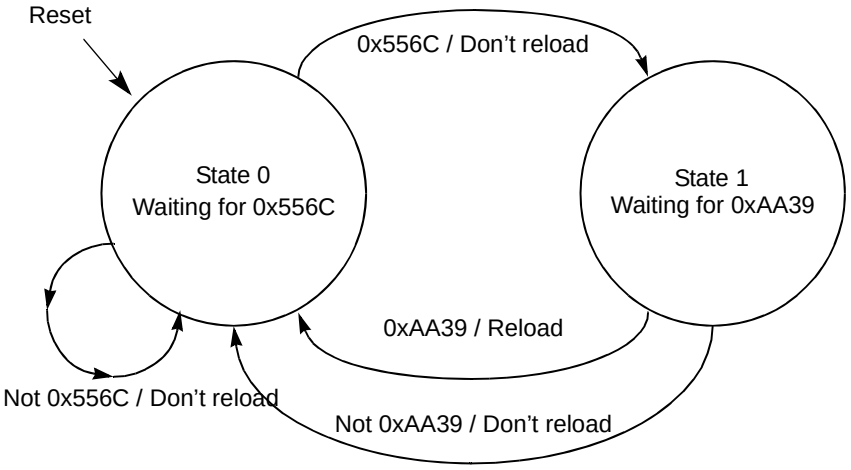


Figure 21-5. Software Watchdog Timer Service State Diagram

Although most software disciplines permit or even encourage the watchdog concept, some systems require a selection of time-out periods. For this reason, the software watchdog timer must provide a selectable range for the time-out period. **Figure 21-6** shows how to handle this need.

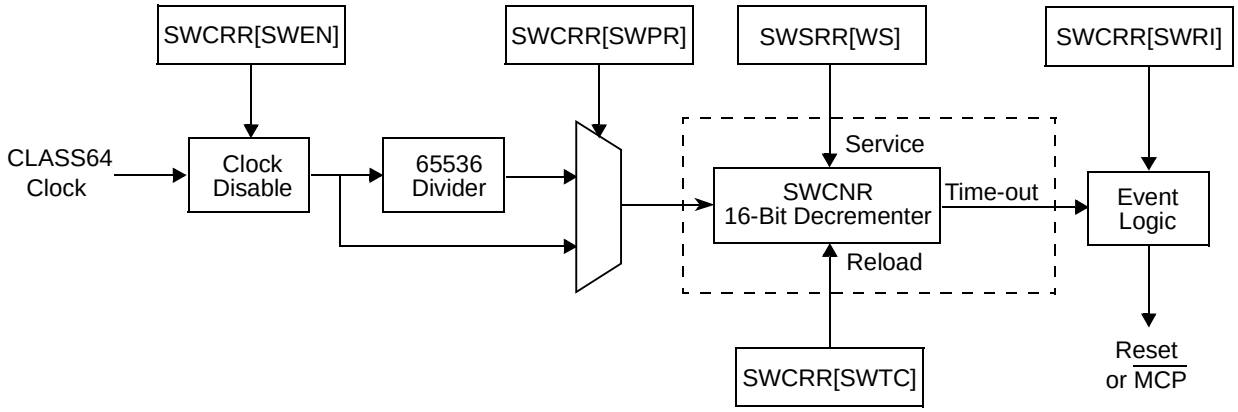


Figure 21-6. Software Watchdog Timer Functional Block Diagram

In **Figure 21-6**, the range is determined by SWCRR[SWTC]. The value in SWTC is then loaded into a 16-bit decrementer clocked by the CLASS64 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, SC3400 DSP core platform level, and software watchdog timers are described in separate subsections.

21.4.1 Device-Level Timers

This section describes the device-level timers. For a complete listing of all registers in all modules with their memory locations, see **Chapter 9**, *Memory Map*. The device-level timer registers are listed as follows, along with the pages on which the registers are discussed:

- Timer Channel Control Registers (TMR[0–3]CTL[0–3], page 21-17.
- Timer Channel Status and Control Registers (TMR[0–3]SCTL[0–3]), page 21-19.
- Timer Channel Compare Register 1 (TMR[0–3]CMP1[0–3]), page 21-21.

- Timer Channel Compare Register 2 (TMR[0–3]CMP2[0–3]), page 21-21.
- Timer Channel Compare Load Register 1 (TMR[0–3]CMPLD1[0–3]), page 21-22.
- Timer Channel Compare Load Register 2 (TMR[0–3]CMPLD2[0–3]), page 21-22.
- Timer Channel Comparator Status and Control Registers (TMR[0–3]COMSC[0–3]), page 21-22.
- Timer Channel Capture Register (TMR[0–3]CAP[0–3]), page 21-22.
- Timer Channel Load Register (TMR[0–3]LOAD[0–3]), page 21-24.
- Timer Channel Hold Registers (TMR[0–3]HOLD[0–3]), page 21-24.
- Timer Channel Counter Register (TMR[0–3]CNTR[0–3]), page 21-24.

Note: The base addresses for the device level timer modules are as follows:

- Timer 0 = 0xFFF26000
- Timer 1 = 0xFFF26100
- Timer 2 = 0xFFF26200
- Timer 3 =- 0xFFF26300

21.4.1.1 Timer Channel Control Registers (TMRnCTLx)

TMR[0–3]CTL[0–3] Timer Control Registers Offset 0x18 + x*0x40

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CM			PCS				SC		ONCE	LEN	DIR	EIN	OFLM		
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-4. TMR[0–3]CTL[0–3] Bit Descriptions

Name	Reset	Description	Settings	
CM 15–13	0	**Count Mode** Control the basic counting behavior of the counter. Rising edges are counted only when TxSCTL[IPS] = 0. Falling edges are counted only when TxSCTL[IPS] = 1. When count mode 010 is selected, the PCS bits must not be set to 1000–1111. When count mode 111 is selected, the PCS bits must be set to one of the “Timer N output” selections.	000 001 010 011 100 101 110 111	No operation. Disabled. Count rising edges of the primary source. Count rising and falling edges of the primary source. Count rising edges of the primary source while the secondary input is high active. Quadrature count mode, uses primary clock and secondary input. Count rising edges of the primary clock; secondary input specifies direction (1 = minus). Edge of the secondary input triggers primary count until a compare occurs. Cascaded timer mode (up/down).

Table 21-4. TMR[0–3]CTL[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
PCS 12–9	0	**Primary Count Source** Select the primary count source. A timer selecting its own output for input is not a legal choice. The result is no counting.	0000 Timer 0 input signal. 0001 Timer 1 input signal. 0010 Timer 2 input signal. 0011 Timer 3 input signal. 0100 Timer 0 output for cascaded timer operation. 0101 Timer 1 output for cascaded timer operation. 0110 Timer 2 output for cascaded timer operation. 0111 Timer 3 output for cascaded timer operation. 1000 Prescaler (CLASS64 clock/1). 1001 Prescaler (CLASS64 clock/2). 1010 Prescaler (CLASS64 clock/4). 1011 Prescaler (CLASS64 clock/ 8). 1100 Prescaler (CLASS64 clock/16). 1101 Prescaler (CLASS64 clock/32). 1110 Prescaler (CLASS64 clock/64). 1111 Prescaler (CLASS64 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 TMRxCMP1 value. If counting down, a successful compare occurs when a timer reaches TMRxCMP2 value.	0 Count repeatedly. 1 Count to the compare value and then stop.
LEN 5	0	**Count Length** Determines whether the timer counts to the compare value and then reinitializes itself, or the timer continues counting past the compare value (binary roll-over). If counting up, a successful compare occurs when the timer reaches the TMRxCMP1 value. If counting down, a successful compare occurs when the timer reaches the TMRxCMP2 value.	0 Roll-over. 1 Count to the compare value and then reinitialize.
DIR 4	0	**Count Direction** Selects either the normal count up direction, or the reverse down direction.	0 Count up. 1 Count down.

Table 21-4. TMR[0–3]CTL[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
EIN 3	0	**External Initialization** Enables another timer within the same module to force the reinitialization of this timer when the other timer has an active compare event. Details on Broadcast mode are presented in **Section 21.1.5.3**, *Broadcast from an Initiator Timer,* on page 21-12.	0 External timers can not force a reinitialization of this timer. 1 External timers may force a reinitialization of this timer.
OFLM 2–0	0	**Output Mode** Determine the mode of operation for the timer output signal. For all of these modes except 000 and 111, the output flag is not toggled when the timer reaches the compare value but instead when the timer advances one value beyond. For example, for a compare value of 7, it toggles on the transition from 7 to 8, not 6 to 7. Unexpected results may occur if the Output mode field is set to use alternating compare registers (mode 100) and the ONCE bit is set.	000 Asserted while timer is active. 001 Clear timer output on successful compare. 010 Set timer output on a successful compare. 011 Toggle the timer output flag when a successful compare occurs. 100 Toggle the timer output flag using alternating compare registers. 101 Set on compare, cleared on secondary input signal's edge. 110 Set on compare, cleared on timer rollover. 111 Enable gated clock output while the timer is active.

21.4.1.2 Timer Channel Status and Control Registers (TMRnSCTLx)

TMR[0–3]SCTL[0–3] Timer Channel Status and Control Register Offset 0x1C + x*0x40

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CM		MSTR	EEOF	VAL	FORC	OPS	OEN
Type	R/W							R	R/W					W	R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-5. TMR[0–3]SCTL[0–3] Bit Descriptions

Name	Reset	Description	Settings
TCF 15	0	**Timer Compare Flag** Set when a successful compare occurs. Clear the bit by writing 0 to it.	0 No successful compare. 1 Successful compare.
TCFIE 14	0	**Timer Compare Flag Interrupt Enable** Enables interrupts when the TCF bit is set.	0 No interrupt. 1 Interrupt.
TOF 13	0	**Timer Overflow Flag** Set when the timer rolls over its maximum value 0xFFFF or 0x0000, depending on count direction. Clear the bit by writing 0 to it.	0 No overflow. 1 Overflow. The timer has reached its maximum or minimum value.
TOFIE 12	0	**Timer Overflow Flag Interrupt Enable** Enables interrupts when the TOF bit is set.	0 No interrupt. 1 Interrupt.

Table 21-5. TMR[0–3]SCTL[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
IEF 11	0	**Input Edge Flag** Set when a positive input transition occurs while the timer is enabled. Clear the bit by writing a 0 to it. Setting the input polarity select (TxSCTL[IPS]) bit enables the detection of negative input edge transitions. Also, the control register secondary count source determines which external input pin is monitored by the detection circuitry.	0 No action. 1 Positive input transition while timer enabled.
IEFIE 10	0	**Input Edge Flag Interrupt Enable** Enables interrupts when the IEF bit is set.	0 No action. 1 Interrupts enabled.
IPS 9	0	**Input Polarity Select** Inverts the input signal polarity.	0 No action. 1 Invert signal polarity.
INPUT 8	0	**Secondary Input Signal** Reflects the current state of the secondary Input signal.	00 Capture function disabled. 01 Load capture register on rising edge of the secondary count source input. 10 Load capture register on falling edge of the secondary count source input. 11 Load capture register on any edge of the secondary count source input.
CM 7–6	0	**Input Capture Mode** Specifies the operation of the capture register as well as the operation of the input edge flag.	00 Capture function is disabled. 01 Load capture register on the rising edge of the secondary count source input. 10 Load capture register on the falling edge of the secondary count source input. 11 Load capture register on any edge of the secondary count source input.
MSTR 5	0	**Initiator Mode** Enables the compare function output to broadcast to the other timers in the module. This identifies a timer as the initiator timer in Broadcast mode. This signal is used to reinitialize the other timers and/or force their outputs. For details on Broadcast mode, see **Section 21.1.5.3**, *Broadcast from an Initiator Timer,* on page 21-12.	0 No action. 1 Broadcast mode.
EEOF 4	0	**Enable External Output Force** Enables the compare from another timer configured as the initiator to force the state of this timer output signal. For details on Broadcast mode, see **Section 21.1.5.3**, *Broadcast from an Initiator Timer,* on page 21-12.	o No action. 1 Other timer can force this timer's output flag signal.
VAL 3	0	**Forced Output Flag Value** Determines the value of the timer output flag signal when a software-triggered FORCE command occurs.	

Table 21-5. TMR[0–3]SCTL[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
FORC 2	0	**Force Output** Forces the current value of the VAL bit to be written to the timer output. Always read this bit as a 0. The VAL and FORCE bits can be written simultaneously in a single write operation. Write to the FORCE bit only if the timer is disabled. Setting this bit while the timer is enabled may yield unpredictable results.	0 No action. 1 Forces the current value of the VAL bit to be written to timer output.
OPS 1	0	**Output Polarity Select** Determines the polarity of the output signal.	0 True polarity. 1 Inverted polarity.
OEN 0	0	**Output Enable** Enables the timer output. The OPS bit determines the polarity of the output.	0 Timer output not enabled. 1 Timer output enabled.

21.4.1.3 Timer Channel Compare 1 Registers (TMRnCMP1x)

TMR[0–3]CMP1[0–3] Timer Channel Compare 1 Registers Offset x*0x40

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

TMR[0–3]CMP1[0–3] store the comparison value (CV) for comparison with the timer value.

21.4.1.4 Timer Channel Compare 2 Registers (TMRnCMP2x)

TMR[0–3]CMP2[0–3] Timer Channel Compare 2 Registers Offset 0x04 + x*0x40

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

TMR[0–3]CMP2[0–3] store the comparison value (CV) for comparison with the timer value.

21.4.1.5 Timer Channel Compare Load 1 Registers (TMRnCMPLD1x)

TMR[0–3]CMPLD1[0–3] Timer Channel Compare Load 1 Registers Offset 0x20 + x*0x40

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CLV1															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR[0–3]CMPLD1[0–3] store the preload value for the TMR[0–3]CMP1[0–3].

21.4.1.6 Timer Channel Compare Load 2 Registers (TMRnCMPLD2x)

TMR[0–3]CMPLD2[0–3] Timer Channel Compare Load 2 Registers Offset 0x24 + x*0x40

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CLV2															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR[0–3]CMPLD2[0–3] store the preload value for the TMR[0–3]CMP2[0–3].

21.4.1.7 Timer Channel Comparator Status and Control Registers (TMRnCOMSCx)

TMR[0–3]COMSC[0–3] Timer Channel Comparator Status and Control Registers Offset 0x28 + x*0x40

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—								TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
Type	R								R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR[0–3]COMSC[0–3] store the preload values used for the TMR[0–3]CMP2[0–3] registers.

Table 21-6. TMR[0–3]COMSC[0–3] Bit Descriptions

Name	Reset	Description	Settings
— 15–8	0	Reserved. Write to 0 for future compatibility.	
TCF2EN 7	0	**Timer Compare 2 Interrupt Enable** Enables the compare 2 interrupt. An interrupt is issued when both this bit and the TCF2 bit are set.	0 No action. 1 Enable compare 2 interrupt.

Table 21-6. TMR[0–3]COMSC[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
TCF1EN 6	0	**Timer Compare 1 Interrupt Enable** Enables the compare 1 interrupt. An interrupt is issued when both this bit and the TCF1 bit are set.	0 No action. 1 Enable compare 1 interrupt.
TCF2 5	0	**Timer Compare 2 Interrupt Source** Indicates a successful comparison of the timer and the TMRxCMP2. This bit is sticky and remains set until it is explicitly cleared by writing a zero to this bit location.	0 Normal operation. 1 Successful compare 2.
TCF1 4	0	**Timer Compare 1 Interrupt Source** Indicates a successful comparison of the timer and the TMRxCMP1. This bit is sticky and remains set until it is explicitly cleared by writing a zero to this bit location.	0 Normal operation. 1 Successful compare 1.
CL2 3–2	0	**Compare Load Control 2** Control when TMRxCMP2 is preloaded with the value from TMRxCMPLD2.	00 Never preload. 01 Load upon successful compare with the value in TMRxCMP1. 10 Load upon successful compare with the value in TMRxCMP2. 11 Reserved.
CL1 1–0	0	**Compare Load Control 2** Control when TMRxCMP1 is preloaded with the value from TMRxCMPLD1.	00 Never preload. 01 Load upon successful compare with the value in TMRxCMP1. 10 Load upon successful compare with the value in TMRxCMP2. 11 Reserved.

21.4.1.8 Timer Channel Capture Registers (TMRnCAPx)

TMR[0–3]CAP[0–3] Timer Channel Capture Registers Offset 0x08 + x*0x40

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

TMR[0–3]CAP[0–3] store the values captured from the timers.

21.4.1.9 Timer Channel Load Registers (TMRnLOADx)

TMR[0–3]LOAD[0–3] Timer Channel Load Registers Offset 0x0C + x*0x40

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

TMR[0–3]LOAD[0–3] store the value used to load the timer.

21.4.1.10 Timer Channel Hold Registers (TMRnHOLDx)

TMR[0–3]HOLD[0–3] Timer Channel Hold Registers Offset 0x10 + x*0x40

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

TMR[0–3]HOLD[0–3] store the value whenever any timer is read.

21.4.1.11 Timer Channel Counter Registers (TMRnCNTRx)

TMR[0–3]CNTR[0–3] Timer Channel Counter Registers Offset 0x14 + x*0x40

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

TMR[0–3]CNTR[0–3] are counters.

21.4.2 SC3400 DSP Core Subsystem Timers

For a detailed information about programming the core subsystem timers, see the *MSC8144 DSP Core Subsystem Reference Manual.*

21.4.3 Software Watchdog Timers

This section describes the software WDT registers, which include:

- System Watchdog Control Register 0–4 (SWCRR[0–4]), see page 21-25.
- System Watchdog Count Register 0–4 (SWCNR[0–4]), see page 21-26.
- System Watchdog Service Register 0–4 (SWSRR[0–4]), see page 21-27.

Note: The watchdog timers use the following base addresses:

- System Watchdog Timer 0 = 0xFFF25000
- System Watchdog Timer 1 = 0xFFF25100
- System Watchdog Timer 2 = 0xFFF25200
- System Watchdog Timer 3 = 0xFFF25300
- System Watchdog Timer 4 = 0xFFF25400

21.4.3.1 System Watchdog Control Register 0–4 (SWCRR[0–4])

SWCRR[0–4] System Watchdog Control Register 0–4 Offset 0x04

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

SWCRR[0–4] control the software watchdog timer period and configure WDT operation. SWCRR can be read at any time but can be written only once after system reset. **Table 21-7** defines the SWCRR[0–4] bit fields.

Table 21-7. SWCRR[0–4] 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.	

Table 21-7. SWCRR[0–4] Bit Descriptions

Name	Reset	Description	Settings
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	1	**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	1	**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 divided by 65536.

21.4.3.2 System Watchdog Count Register 0–4 (SWCNR[0–4])

SWCNR[0–4] System Watchdog Count Register 0–4 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[0–4] provide visibility to the WDT counter values. Writes to SWCNR have no effect and terminate without transfer error exception.

Table 21-8. SWCNR[0–4] 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 0–4 (SWSRR[0–4])

SWSRR[0–4] System Watchdog Service Register 0–4 Offset 0x0E

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

When the WDT is enabled, writing 0x556C followed by 0xAA39 to the SWSRR register before the WDT times out prevents a reset. If SWSRR is not serviced before the time-out, the watchdog timer sends a signal to the reset or interrupt controller module. Both writes must occur before the time-out in the order listed, but any number of instructions can be executed between the two writes. Reset initializes the SWSRR[WS] field to 0x0000. SWSRR can be written at any time, but returns all zeros when read. **Table 21-9** defines the bit fields of SWSRR.

Table 21-9. SWSRR[0–4] Bit Descriptions

Name	Reset	Description
WS 15–0	0	**Software Watchdog Service Field** Write 0x556C followed by 0xAA39 to this register to prevent software watchdog timer time-out. SWSRR[WS] can be written at any time, but returns all zeros when read. Writing any other value resets the servicing sequence.

GPIO 22

The MSC8144E device has 32 general-purpose I/O (GPIO) ports, 28 of which are multiplexed as either GPIO ports or dedicated peripheral interface ports. In addition, sixteen of the GPIOs can be configured as external maskable interrupt inputs. As GPIOs, each port is configured as an input or output (with a register for data output that is read or written at any time). If configured as output, the GPIO ports can also be configured as open-drain (that is, configured in an active low wired-OR configuration on the board). In this mode, an output drives a zero voltage but goes to tri-state when driving a high voltage. GPIO ports do not have internal pull-up resistors. The dedicated MSC8144E peripheral functions multiplexed with the GPIO ports are grouped to maximize the usefulness of the ports in the greatest number of MSC8144E applications.

Note: To understand the port assignment capability described in this chapter, you must first understand the Time-Division Multiplexing (TDM), timers, UART, I^2C, PCI, Ethernet, and ATM UTOPIA peripherals.

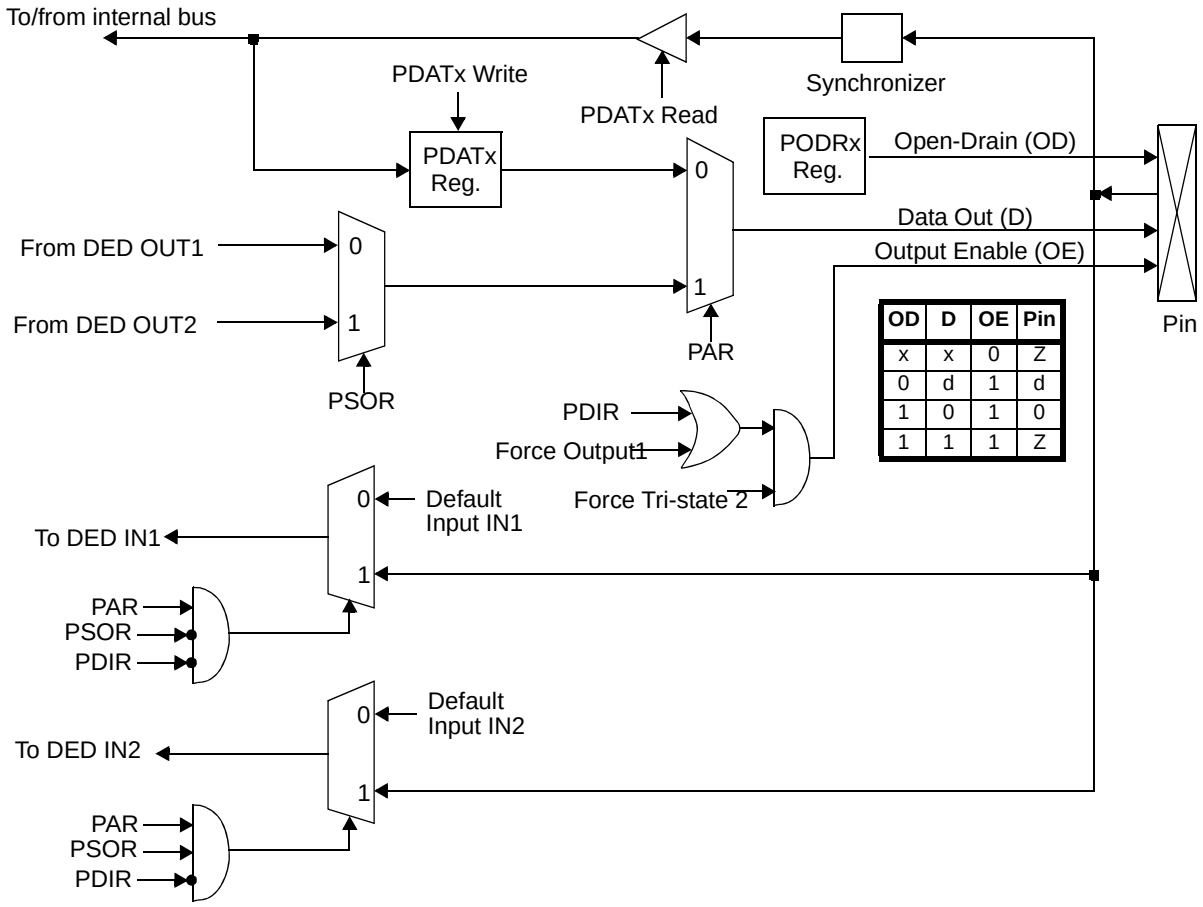
22.1 Features

Following are the key features of the GPIO ports:

- 32 GPIO ports.
- All ports are bidirectional.
- Most ports have alternate on-device peripheral functions.
- All 32 ports are set as GPIO inputs at system reset.
- 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.
- Eighteen of the GPIOs can function as PCI signals.
- Ten of the GPIOS can function as TDM signals.
- Five of the GPIOs can function as timer signals.
- Two of the GPIOs can function as a UART port.
- Two of the GPIOs can function as an I^2C interface.
- One GPIO can function as an Ethernet signal.
- One GPIO can function as a UTOPIA signal.

22.2 GPIO Block Diagram

Figure 22-2 shows a GPIO functional block diagram.

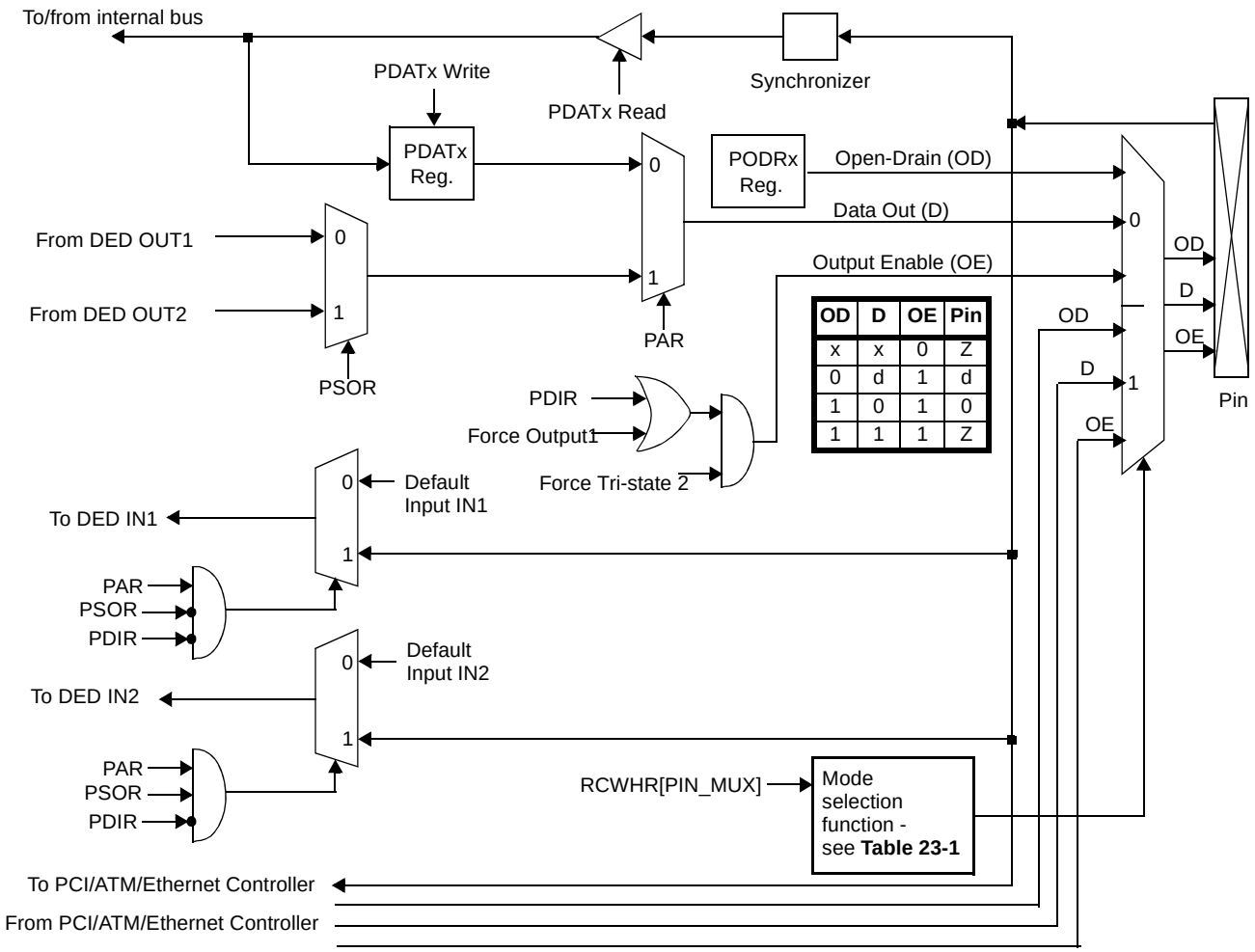


Notes: 1. Force Output may be asserted high by dedicated peripheral 2 direction control, only when PAR = 1 and PSOR = 1 (selects this peripheral) and PDIR = 0 (input). It is used for bidirectional operation allowing the peripheral to dynamically control the port direction.
2. Force Tri-state may be asserted 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 each the ports as inputs using the General Input Enable (GIER) register. See **8.2.9**, *GPIO Input Enable Register (GIER)* for details.			

Figure 22-1. Port Functional Operation



OD	D	OE	Pin
x	x	0	Z
0	d	1	d
1	0	1	0
1	1	1	Z

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

Figure 22-2. Port Functional Operation

22.3 I/O Mode Functionality of GPIO

Whether a port operates as a GPIO/dedicated input/output or PCI/UTOPIA/Ethernet input/output depends on the settings of RCWRH[PIN_MUX], as described in **Table 22-1**.

Table 22-1. GPIO/Dedicated Functionality Versus I/O Mode Multiplexing

GPIO	I/O Mode (PIN_MUX field lowest four bits)							
	Mode 0 (0000)	**Mode 1 (0001)**	**Mode 2 (0010)**	**Mode 3 (0011)**	**Mode 4 (0100)**	**Mode 5 (0101)**	**Mode 6 (0110)**	**Mode 7 (0111)**
0	GPIO							
1	GPIO							
2	GPIO							
3	GPIO							
4	GPIO			PCI		GPIO		
5	GPIO			PCI		GPIO		
6	GPIO			PCI		GPIO		
7	GPIO			PCI		GPIO		
8	GPIO			PCI		GPIO		
9	GPIO			PCI		GPIO		
10	GPIO			PCI		GPIO		
11	GPIO			PCI		GPIO		
12	GPIO			PCI		GPIO		
13	GPIO							
14	GPIO							
15	GPIO							
16	GPIO							
17	UTOPIA		GPIO	UTOPIA	PCI	UTOPIA		
18	GPIO				PCI	GPIO		UTOPIA
19	GPIO				PCI	GPIO		UTOPIA
20	GPIO				PCI	GPIO		UTOPIA
21	GPIO							
22	GPIO							
23	GPIO							
24	GPIO							
25	GPIO	Ethernet	PCI			GPIO	Ethernet	GPIO
26	GPIO							
27	GPIO							
28	GPIO			PCI		GPIO		
29	GPIO		PCI			GPIO		
30	GPIO		PCI			GPIO		
31	GPIO		PCI			GPIO		

Note: GPIO includes any signal configured by the GPIO configuration registers, including GPI, GPO, IRQ, and other dedicated functions as listed in **Table 22-2**.

When the selected signal is not controlled by the GPIO configuration registers:

- Its direction and driving mode are controlled by the PCI, ATM, or Ethernet controller regardless of PDIR*x* and PODR*x* values. See **Chapter 15**, *PCI*, **Chapter 19**, *Asynchronous Transfer Mode (ATM) Controller*, and **Chapter 20**, *Ethernet Controller*.
- Data written to PDAT*x* is stored in the output register, but it is prevented from reaching the external port.
- A read of PDAT*x* returns the data at the external port, independently of whether the port is defined as input or output in Ethernet controller.
- Default values are supplied to internal peripheral inputs connected to this GPIO port.

22.4 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, configuring all ports as GPIO inputs.

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, 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 22-2**. 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 22-2 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 22-2** as primary and secondary input.

Table 22-2. 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	—	—	0	—	—	0
1	—	—	0	—	—	0
2	—	—	0	—	—	0
3	—	—	1	—	—	0
4	—	$\overline{\text{IRQ10}}$	1	TDM6RCLK	TDM6RCLK	0
5	—	$\overline{\text{IRQ11}}$	1	TDM6RDAT	TDM6RDAT	0
6	—	$\overline{\text{IRQ12}}$	1	TDM6RSYN	TDM6RSYN	0
7	—	$\overline{\text{IRQ13}}$	1	TDM6TDAT	TDM6TDAT	0
8	—	$\overline{\text{IRQ14}}$	1	TDM6TSYN	TDM6TSYN	0
9	—	—	1	TDM5RDAT	TDM5RDAT	0
10	—	—	1	TDM5RSYN	TDM5RSYN	0
11	—	—	1	TDM5TDAT	TDM5TDAT	0
12	—	—	1	TDM5TSYN	TDM5TSYN	0
13	QE_BRGC0	—	1	TIMER0	TIMER0	0
14	—	$\overline{\text{IRQ8}}$	1	URXD	URXD	1
15	—	$\overline{\text{IRQ9}}$	1	UTXD	UTXD	0
16	—	$\overline{\text{IRQ0}}$	1	QE_BRGC1	—	0
17	—	—	1	TIMER1	TIMER1	0
18	—	—	1	TIMER2	TIMER2	0
19	—	—	1	TIMER3	TIMER3	0
20	—	—	1	TIMER4	TIMER4	0
21	—	$\overline{\text{IRQ1}}$	1	—	SPICLK	0
22	—	$\overline{\text{IRQ4}}$	1	—	SPIMOSI	0
23	—	$\overline{\text{IRQ5}}$	1	—	SPIMISO	0
24	—	$\overline{\text{IRQ6}}$	1	—	$\overline{\text{SPISEL}}$	1
25	—	$\overline{\text{IRQ15}}$	1	—	GE1_RX_ER	0
26	—	—	0	SCL	SCL	0
27	—	—	0	SDA	SDA	0
28	—	—	0	TDM5RCLK	TDM5RCLK	0
29	—	$\overline{\text{IRQ7}}$	1	—	—	0
30	—	$\overline{\text{IRQ2}}$	1	—	—	0
31	—	$\overline{\text{IRQ3}}$	1	—	—	0

Note: You must enable the port to use it as an input. See **8.2.9**, *GPIO Input Enable Register (GIER)* for details.

22.5 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 registers:

- Pin Open-Drain Register (PODR), **page 22-7**
- Pin Data Register (PDAT), **page 22-8**
- Pin Data Direction Registers (PDIR), **page 22-9**
- 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.

22.5.1 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. When a GPIO port has Ethernet functionality (see **Table 22-1**), PODR*x* does not influence its driving mode.

Table 22-3. 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.

22.5.2 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, 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, 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, the state of the actual pin is read.

When a GPIO port has Ethernet functionality (see **Table 22-1**), data written to PDAT*x* is stored in the output register, but it is prevented from reaching the external port. Read of PDAT*x* returns the data at the external port, independently of whether the port is defined as input or output in Ethernet Controller.

22.5.3 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 22-4. PDIR Bit Descriptions

Name	Reset	Description	Settings
DR 0–31	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 8.2.9**, *GPIO Input Enable Register (GIER)*, on page 8-11 for details.			

22.5.4 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 22-5. PAR Bit Descriptions

Name	Reset	Description	Bit Settings
DD[31–0] 0–31	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. When a GPIO port has Ethernet functionality (see **Table 22-1**), its PAR bit should be set to 0.	0 GPIO. The peripheral functions of the pin are not used. 1 Dedicated peripheral function.

22.5.5 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 22-6. PSOR Bit Descriptions

Name	Reset	Description	Settings
SO[31–0] 0–31	0	**Special-Option** Determines whether an external connection configured for a dedicated function (PAR[DD] = 1) uses option 1 or option 2. See **Table 22-2**.	0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.

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.

Hardware Semaphores 23

The MSC8144E hardware semaphores (HS) hold eight coded hardware semaphores. A coded hardware semaphore provides a simple way to achieve a “lock” operation via a single write access, eliminating the need for such sophisticated read-modify-write mechanisms as the reservation. Using the hardware semaphores, external hosts and on-device bus initiators can protect internal and external shared resources and ensure coherency in any sequence of operations on these resources. Each coded hardware semaphore is an eight-bit register with a selective write mechanism, as follows (see **Figure 23-1**):

- The semaphore is *free* if its value is zero.
- The semaphore is *locked* if its value is non-zero and its value is the *lock code*. Each processor/task that needs the lock capability of the semaphore must have a unique non-zero eight-bit code for locking the semaphore for correct protocol operation.
- A write of a non-zero value (lock code) is successful only if the current value of the semaphore is zero (free). This write is defined as a successful *lock* operation, and the written value is the *lock code*.
- A write of a non-zero value (lock code) is ignored if the current value of the semaphore is non-zero (locked). This write is defined as a failed *lock* operation, since the coded semaphore is considered locked with the non-zero code it holds.
- To maintain the protocol coherency, only the initiator that successfully locked the semaphore is allowed (and obligated) to free it. A write of zero is always successful and is defined as a *free* operation.

When a processor/task must lock an unlocked semaphore to its unique code, it writes its unique lock code to the semaphore register. It then reads back the semaphore value, and if it matches its lock code, the lock operation is successful. A value mismatch (a different non-zero code or zero) indicates to the initiator that the lock operation failed and must be retried. After a successful lock operation, the initiator can do any coherent operation associated with this semaphore and then it must free the semaphore (write zero).

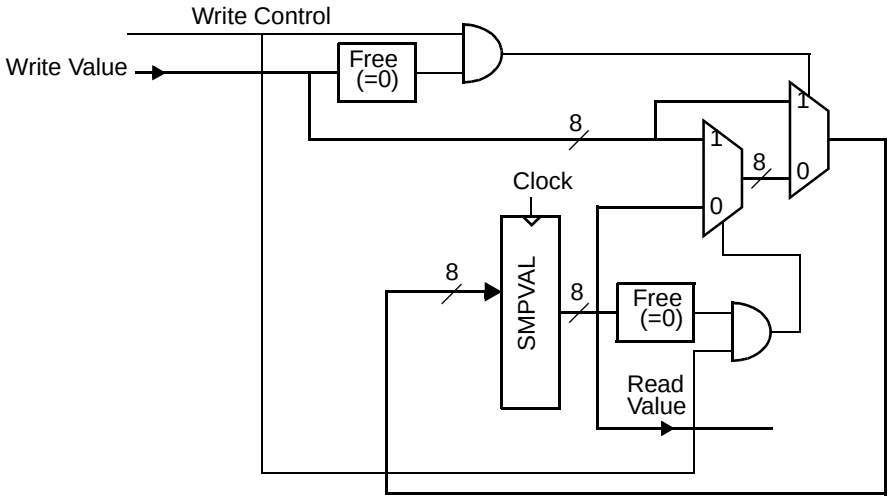


Figure 23-1. Hardware Semaphore Block Diagram

The hardware semaphore registers reside in the CCSR address space and have a constant offset. They are accessed through the MBus. The addresses of the hardware semaphore registers are presented in **Chapter 9**, *Memory Map*. The registers use a base address of 0xFFF27100.

HSMPR[0–7] Hardware Semaphore Register n Offset n*0x8

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

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

Table 23-1. HSMPRx Bit Descriptions

Name	Reset	Description	Settings
— 31–8	0	Reserved. Write to zero for future compatibility.	
SMPVAL 7–0	0	**Semaphore Value** The eight-bit coded semaphore value. It holds the current semaphore value. A non-zero value indicates the current lock code.	0 Semaphore is free. It is writable to any value. ≠ 0 Semaphore is locked with lock code indicated by its current value. It is writable only to zero.

I²C 24

The inter-integrated circuit (IIC or I²C) bus is a two-wire—serial data (SDA) and serial clock (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 MSC8144E device, the maximum SCL frequency is 400 kHz. The I2C standard specification requires pull-up resistors on SDA and SCL. **Figure 24-1** is a block diagram of the I²C block.

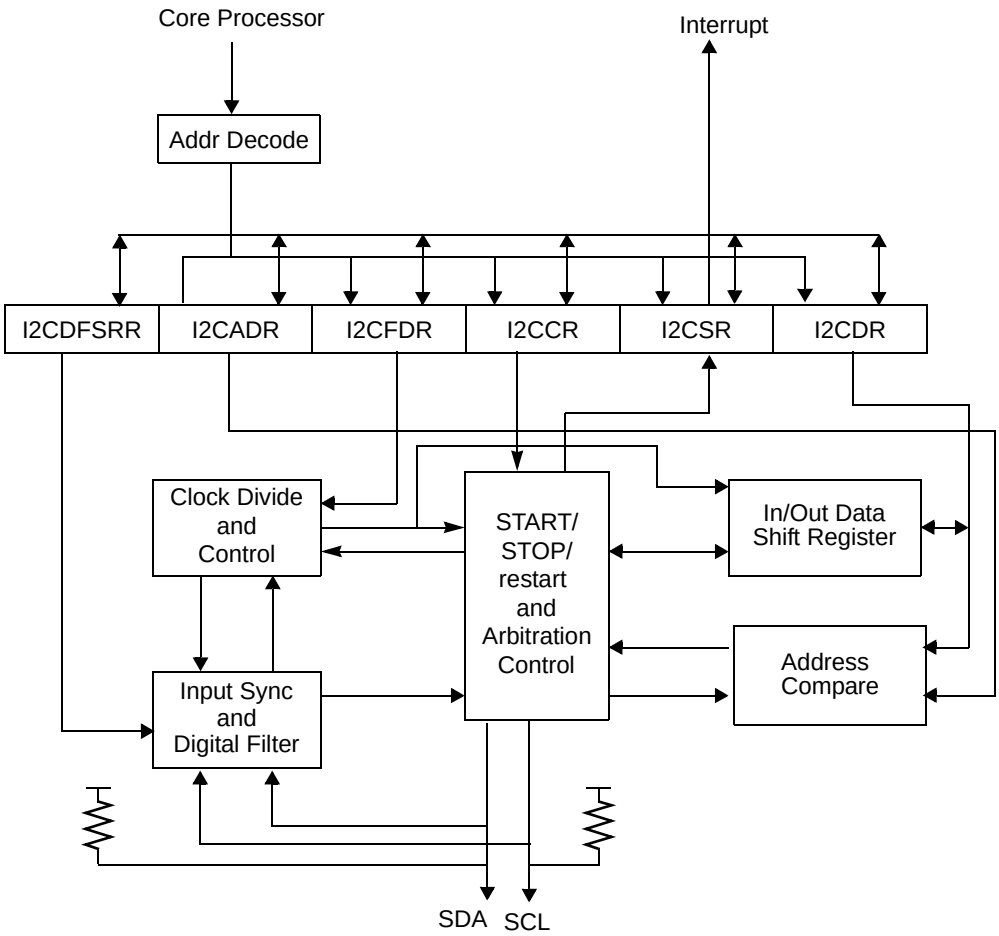


Figure 24-1. I²C Controller Block Diagram

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

24.1 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 SDA 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 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*.

24.2 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

24.2.1 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

24.2.2 Input Synchronization

The input synchronization block synchronizes the input SCL and SDA signals to the CLASS64 clock and detects transitions of these signals.

24.2.3 Digital Input Filter

The SCL and 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.

24.2.4 Transaction Monitoring

The different conditions of the I^2C data transfers are monitored as follows:

- START conditions are detected when an SDA fall occurs while SCL is high.
- STOP conditions are detected when and SDA rise occurs while 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.

24.2.5 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 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 SDA samples low).
- A start condition is attempted when the requesting device is not the bus owner
- Unexpected STOP condition detected

24.2.6 Transfer Control

The I^2C contains logic that controls the output to the serial data (SDA) and serial clock (SCL) lines of the I^2C. The SCL output is pulled low as determined by the internal clock generated in the clock module. The SDA output can only change at the midpoint of a low cycle of the SCL, unless performing a START, STOP, or restart condition. Otherwise, the SDA output is held constant. The 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 SCL signal corresponds to the internal 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

24.2.7 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 SDA line.

24.2.8 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

24.3 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.

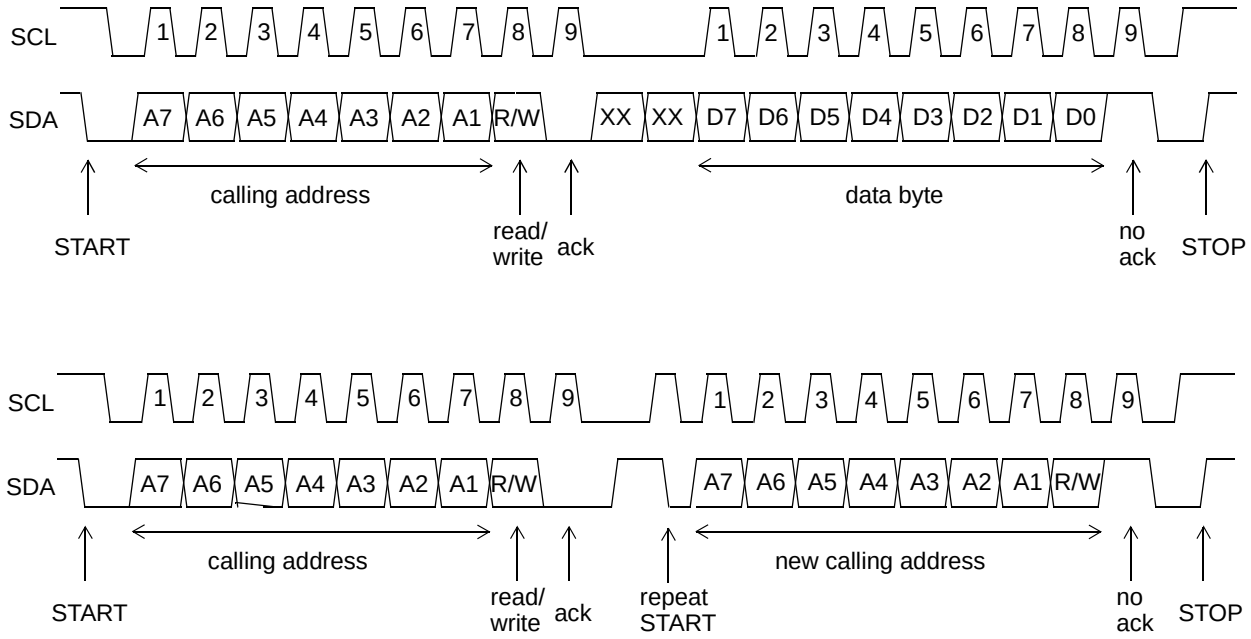


Figure 24-2. I^2C Interface Transaction Protocol

24.3.1 START Condition

When the I^2C bus is not engaged (both SDA and 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 SDA while 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].

24.3.2 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 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 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 SCL is low and must be held stable while SCL is high, as shown in **Figure 24-2**. There is one clock pulse on SCL for each data bit, and the most significant bit (msb) is transmitted first. Each byte of data must be followed by an acknowledge bit, which is signaled from the receiving device by pulling the 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 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 SDA line for the initiator to generate a STOP or a START condition.

24.3.3 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.

24.3.4 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 SDA signal while 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 24.3.3**, 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.

24.3.5 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 SDA while another initiator transmits a logic 0. The losing initiators immediately switch to target-receive mode and stop driving the 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:

- SDA sampled as low when the initiator drives a high during address or data-transmit cycle.
- 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 SDA signal is low and 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 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.

24.3.6 Clock Synchronization

Due to the wire AND logic on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices begin counting their low period when the initiator drives the SCL line low. After a device has driven SCL low, it holds the SCL line low until the clock high state is reached. However, the change of low-to-high in a device clock may not change the state of the SCL line if another device is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time. When all devices concerned have counted off their low period, the synchronized SCL line is released and pulled high. Then there is no difference between the device clocks and the state of the SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the SCL line low again.

24.3.7 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Target devices can hold the 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 SCL line.

24.3.8 Clock Stretching

Targets can use the clock synchronization mechanism to slow down the transfer bit rate. After the initiator has driven the SCL line low, the target can drive SCL low for the required period and then release it. If the target SCL low period is greater than the initiator SCL low period, then the resulting SCL bus signal low period is stretched.

24.4 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.

24.4.1 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 22**, *GPIO* for details.
- Update I2CFDR[FDR] and select the required division ratio to obtain the SCL frequency from the CLASS64 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.

24.4.2 Generation of START

After initialization, the following sequence can be used to generate START:

- If the device is connected to a multi initiator 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.

24.4.3 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 24.4.10** 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 24.4.10, “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 24.4.10** for details.

24.4.4 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 24.4.10, “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.

24.4.5 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].

24.4.6 Generation of SCL When 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 SCL signal (even though 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 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 SCL so that the device driving 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.
- Wait for a period that is equivalent to at least one SCL cycle. SCL does not toggle until the dummy read executes.
- Read the I2CDR.
- Return the I^2C module to target mode by setting I2CCR to 0x80.
- Wait for a period that is equivalent to at least 9 SCL cycles and verify that the bus is released.

24.4.7 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 SCL low between byte transfers. SCL is released when the I2CDR is accessed in the required mode.

24.4.8 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 SCL so that the initiator can generate a STOP condition. See Section 24.4.10, “Interrupt Service Routine Flowchart.”

24.4.9 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 24.2.5**, for details.

24.4.10 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^2C register read or write to guarantee in-order instruction execution.

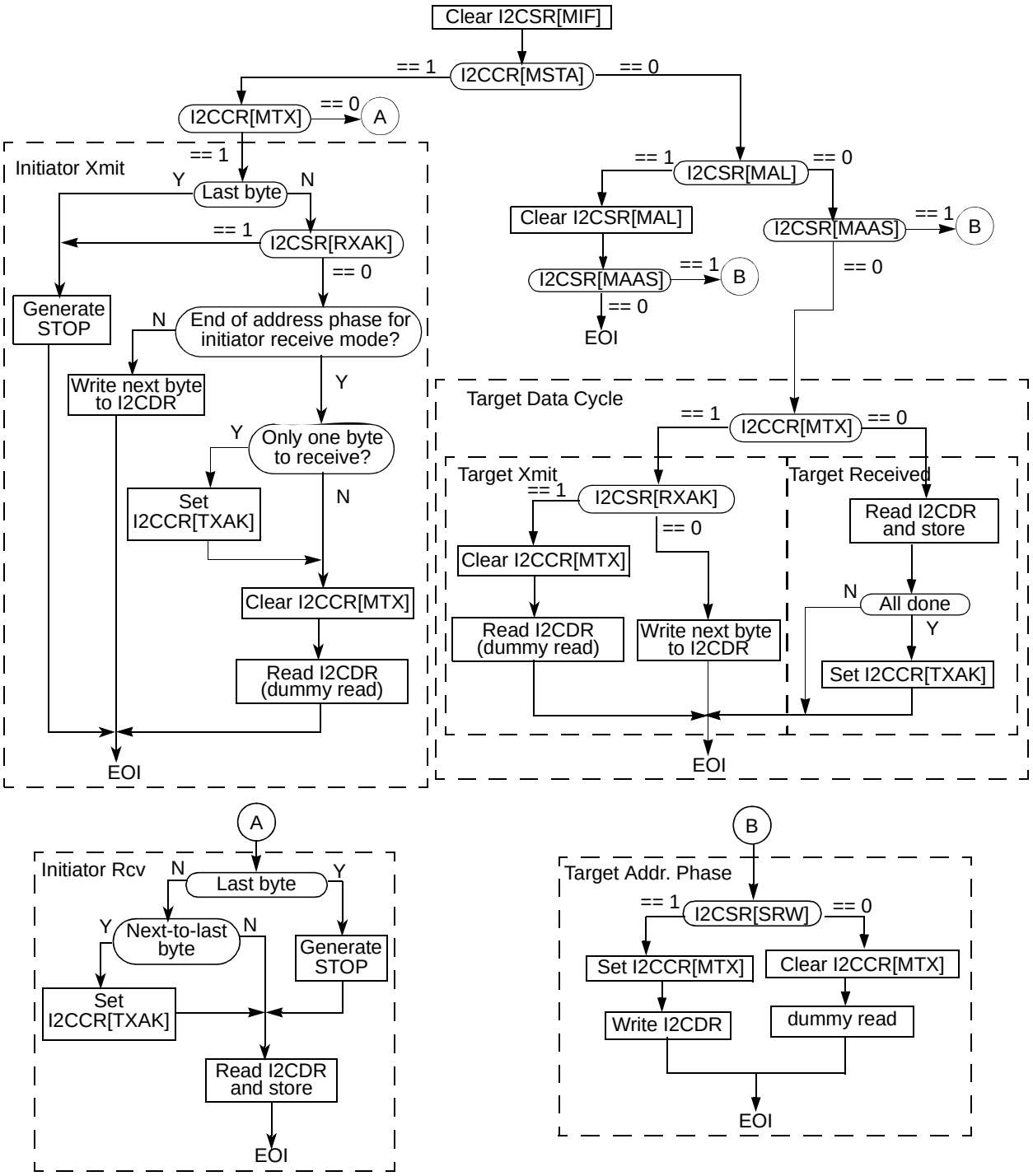


Figure 24-3. Example I^{2}C Interrupt Service Routine Flowchart

24.5 Programming Model

The registers covered in this section are as follows:

- I^{2}C Address Register (I2CADR), page 24-15
- I^{2}C Frequency Divider Register (I2CFDR), page 24-16
- I^{2}C Control Register (I2CCR), page 24-17
- I^{2}C Status Register (I2CSR), page 24-18
- I^{2}C Data Register (I2CDR), page 24-20
- Digital Filter Sampling Rate Register (I2CDFSRR), page 24-20

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 I2C registers use a base address of: 0xFFF24C00.

24.5.1 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.

24.5.2 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 SCL is equal to the CLASS64 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

24.5.3 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 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 (SDA low) is sent to the bus on the 9th clock after receiving one byte of data. 1 No acknowledge signal response is sent (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^2C module to accept broadcast messages at address zero.	0 Broadcast disabled. 1 Broadcast enabled.

24.5.4 I^2C Status Register (I2CSR)

I2CSR I^2C 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^2C 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^2C bus is idle. 1 I^2C 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 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.

24.5.5 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.

24.5.6 Digital Filter Sampling Rate Register (I2CDFSRR)

I2CDFSRR Digital Filter Sampling Rate Register Offset 0x14

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

Table 24-6 describes the I2CDFSRR fields.

Table 24-6. I2CDFSRR Bit Descriptions

Name	Reset	Description
— 7–6	0	Reserved. Write to zero for future compatibility.
DFSR 5–0	010000	**Digital Filter Sampling Rate** To assist in filtering out signal noise, the sample rate is programmed. This field is used to prescale the frequency at which the digital filter takes samples from the I²C bus. The resulting sampling rate is calculated by dividing the system frequency by the non-zero value of DFSR. If I2CDFSRR is set to zero, the I²C bus sample points default to the reset divisor 0x10. The value of DFSR should be defined by the system noise and the I2CFDR[FDR] value. DFSR must be less than six times the division factor defined by I2CFDR[FDR].

Debugging, Profiling, and Performance Monitoring 25

The MSC8144E 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 L2 ICache, DMA controller, and RapidIO controller.

25.1 TAP, Boundary Scan, and OCE

The dedicated user-accessible test access port (TAP) is designed to be compatible with the **IEEE** Std. 1149.1 test access port and boundary scan architecture. Problems associated with testing high-density circuit boards led to development of this standard under the sponsorship of the test technology committee of **IEEE** and the joint test action group (JTAG). The MSC8144E supports circuit-board test strategies based on this standard. This section covers aspects of JTAG that are specific to the MSC8144E. It includes the items that the standard requires to be defined, with additional information specific to the MSC8144E device. For details on the standard, refer to the **IEEE** Std. 1149.1 documentation.

The JTAG port also provides access to the on-chip emulator (OCE) module, a dedicated block for debugging applications. Therefore, this section includes information on registers and functionality of the OCE module that are specific to the MSC8144E. For details on the OCE module functionality, see the *OCE Architecture Manual*.

The SC3400 core OCE module interfaces with the SC3400 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 SC3400 core-based devices. Special

circuits and dedicated signals on the SC3400 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.

25.1.1 Overview

The MSC8144E 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 MSC8144E JTAG can do the following:

- Perform boundary scan operations to check circuit-board electrical continuity.
- Bypass the MSC8144E for a given circuit-board test by effectively reducing the Boundary Scan Register (BSR) to a single cell.
- Sample the MSC8144E 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 MSC8144E-based device.
- Force test data onto the outputs of an MSC8144E-based device while replacing its BSR in the serial data path with a single-bit register.

Note: Precautions must be taken to ensure that the **IEEE** Std. 1149.1-like test logic does not interfere with non-test operation.

To access the JTAG registers, shift the appropriate command into the JTAG instruction register and then shift the required value into the register. See **Section 25.1.3** for a discussion of the JTAG instructions. **Figure 25-1** shows the MSC8144E 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 MSC8144E 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.

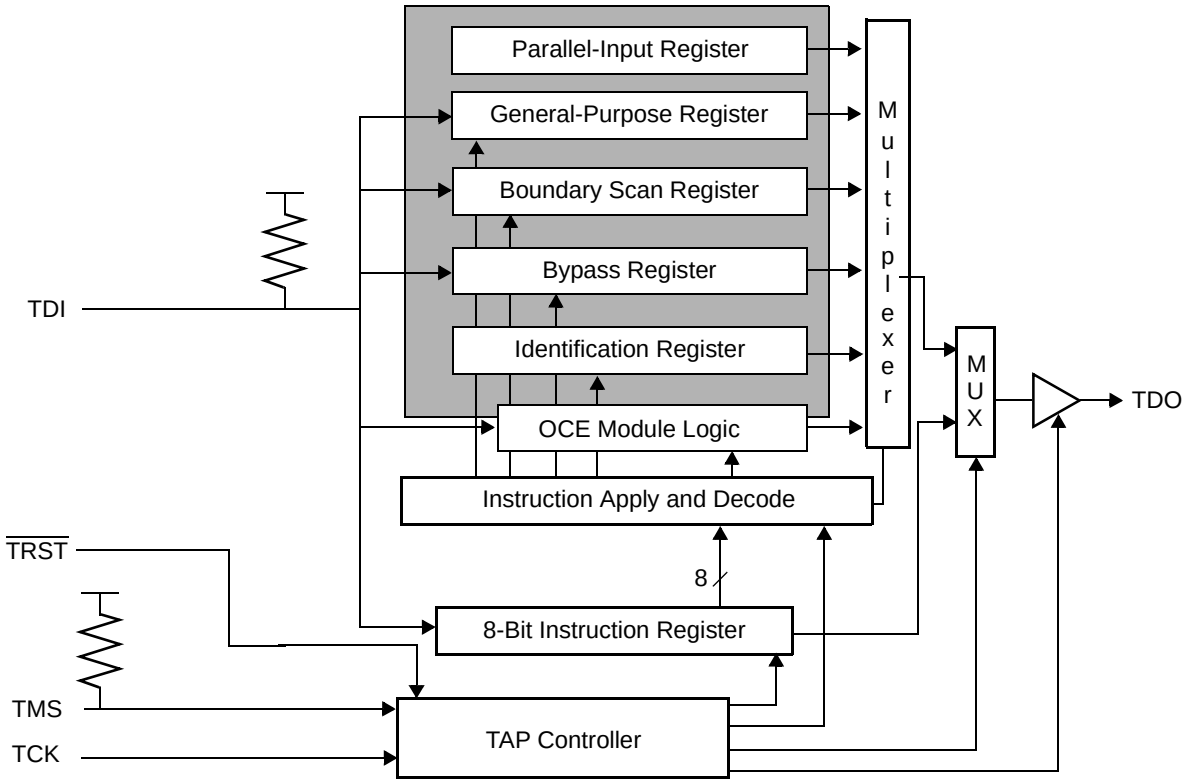


Figure 25-1. Test Logic Block Diagram

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.
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 that provides initialization of the TAP controller and other logic required by the standard.

25.1.2 TAP Controller

The TAP controller interprets the sequence of logical values on the TMS signal. This synchronous state machine controls the operation of the JTAG logic. The value adjacent to each arc in **Figure 25-2** represents the value of the TMS signal sampled on the rising edge of the TCK signal. For a description of the TAP controller states, refer to the **IEEE** Std. 1149.1 documentation.

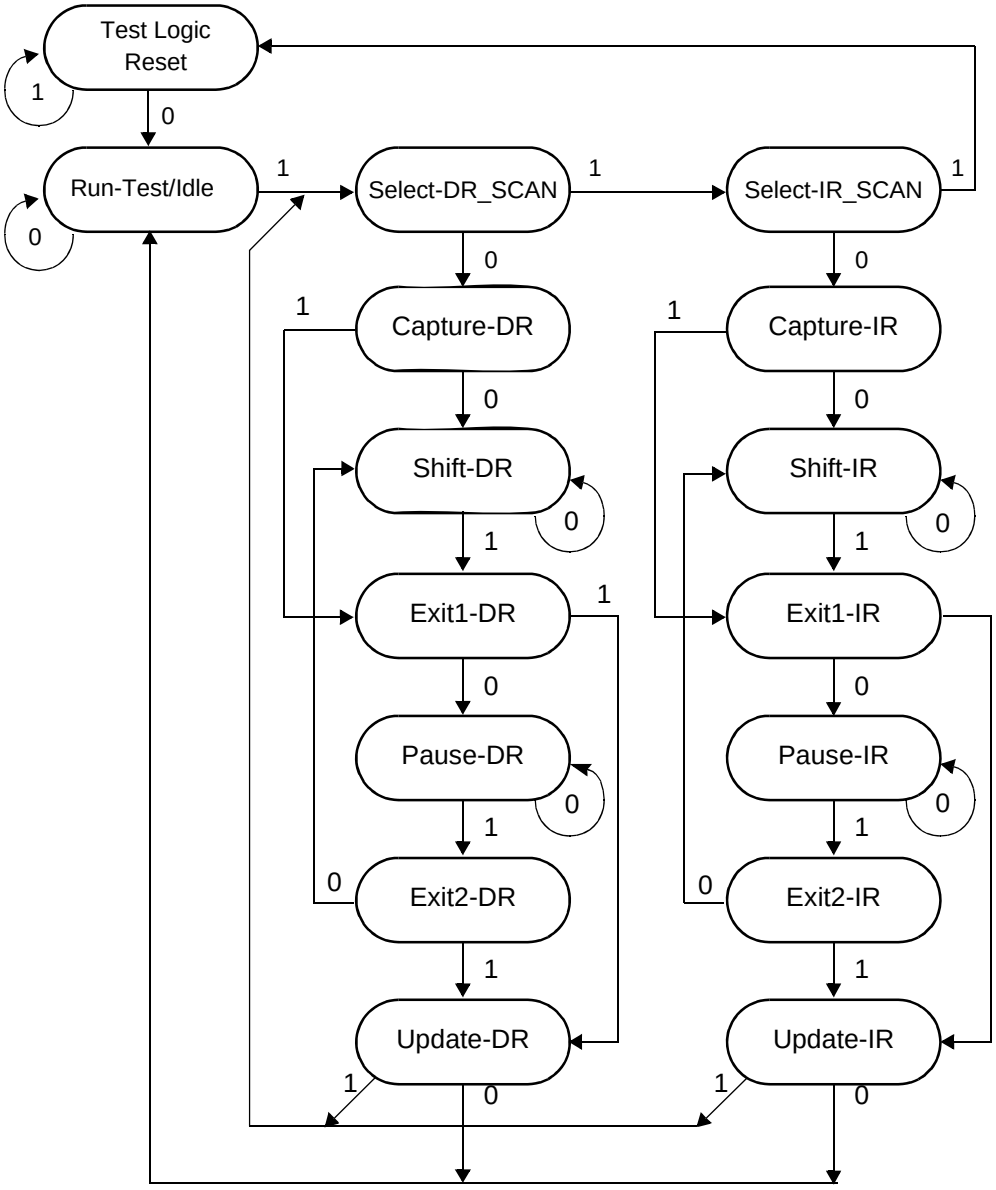


Figure 25-2. TAP Controller State Machine

25.1.3 Instruction Decoding

The MSC8144E includes the three mandatory public instructions EXTEST, SAMPLE/PRELOAD, and BYPASS and also supports the optional CLAMP and HIGHZ instructions defined by **IEEE Std.** 1149.1. The following public instructions perform key functions:

- READ_STATUS enables the JTAG port to query the status of the OCE circuitry.
- ENABLE_ONCE enables the JTAG port to communicate with the OCE circuitry.
- DEBUG_REQUEST enables the JTAG port to force the MSC8144E 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 SC3400 OCE module choose cells for each module that you want to enable. Since there are four internal OCE modules, you must shift 4 bits to the choose cells. For details, see **Section 25.1.4**.

The MSC8144E 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 SC3400 core, whose status is output from the multiplexer. Therefore, the status of all SC3400 cores can be viewed serially by updating the GPR between each SC3400 core status reading. Alternatively, all four SC3400 cores can be viewed simultaneously from the PIREG. For details on core states, refer to the *SC3000-Family Processor Core Reference Manual*.

Table 25-2. Instruction Register Capture and SC3400 Core Status Values

Name/bits	Description	Settings
FBiST_Done 7	**FBIST Done** Field built-in self test completed.	0 FBIST not complete. 1 FBIST completed
MBIST_Failed 6	**MBIST Failed** Indicates an MBIST failure.	0 No MBIST failure. 1 MBIST failed
MBIST_Done 5	**MBIST Done** Indicates that the MBIST is completed.	0 Either an activated MBIST is still running or no MBIST was initiated by the last $\overline{\text{HRESET}}$ 1 All active MBISTs are complete
— **4**	Reserved. Always 0.	

Table 25-2. Instruction Register Capture and SC3400 Core Status Values

Name/bits	Description	Settings
clock_end_ count 3	**Clock End Count** Indicates that the counter in the clock block completed its count.	0 Clock block done signal not asserted 1 Clock block done signal asserted
retention-stopped 2	**Retention Stopped** Indicates whether all of the activated MBISTs in retention mode have stopped. This bit is cleared by assertion of HRESET or MBIST initiation.	0 An activated MBIST in retention mode has not stopped, or no MBIST retention stop was performed since the last HRESET assertion 1 All activated MBISTs in retention mode stopped.
— 1–0	Contains value (01) required by the JTAG standard	Read-only

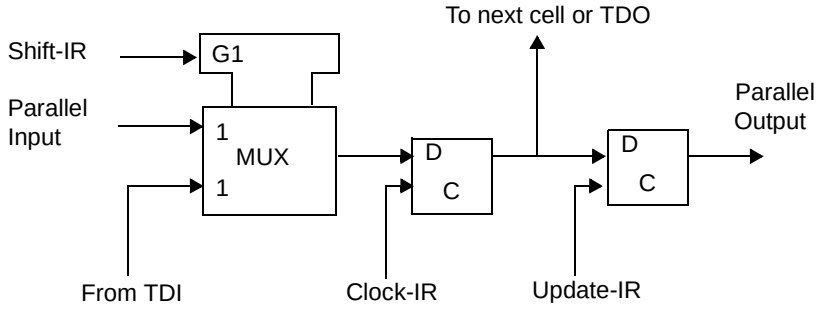


Figure 25-3. Instruction Register (IR) Configuration

Table 25-3 describes the 8-bit instructions coded in the Instruction Register.

Table 25-3. Instruction Decoding

Opcode	Instruction	Updates IR	Description
Standard Instructions			
0x00	EXTEST	Yes	Selects the Boundary Scan Register (BSR). EXTEST also asserts internal reset for the MSC8144E system logic to force a predictable internal state while external boundary scan operations are performed. By using the TAP, the register can: • Scan user-defined values into the output buffers • Capture values presented to inputs • Control the direction of bidirectional signals • Control the output drive of tri-statable outputs For details on the function and use of EXTEST, refer to the **IEEE** Std. 1149.1 documentation. Although the latest specification recommends not using all zeroes, it was mandated by earlier versions of the specification and is retained for backward compatibility.
0xF0	SAMPLE	Yes	SAMPLE provides a means to obtain a snapshot of system data and control signals.
0xF0	PRELOAD	Yes	Initializes the BSR output cells prior to the selection of EXTEST. This initialization ensures that known data appears on the outputs when an EXTEST instruction is entered.

Table 25-3. Instruction Decoding (Continued)

Opcode	Instruction	Updates IR	Description
0xF1	CLAMP	Yes	Optional in the **IEEE** Std. 1149.1. This public instruction selects the one-bit Bypass Register as the serial path between TDI and TDO, while allowing signals driven from the component to be determined from the Boundary Scan Register. During testing of ICs on PCBs, it may be necessary to place static guarding values on signals that control logic operations not involved in the test. The EXTEST instruction can be used for this purpose, but since it selects the BSR, the required guarding signals would be loaded as part of the complete serial data stream shifted in, both at the start of the test and each time a new test pattern is entered. Since the CLAMP instruction allows guarding values to be applied using the BSR of the appropriate ICs while selecting their Bypass Registers, it allows much faster testing than EXTEST. Data in the boundary scan cell remains unchanged until a new instruction is shifted in. **Note:** The CLAMP instruction also asserts internal reset for the MSC8144E system logic to force a predictable internal state while external boundary scan operations are performed.
0xF2	HIGHZ	Yes	Optional in the **IEEE** Std. 1149.1. It is a manufacturer's public instruction to prevent back-drive of the outputs during circuit-board testing. When HIGHZ is invoked, all output drivers, including the two-state drivers, are turned off (that is, high impedance). The HIGHZ instruction selects the Bypass Register. It also asserts internal reset for the MSC8144E system logic to force a predictable internal state while external boundary scan operations are performed.
0xF3	IDCODE	Yes	Selects the ID Register. This instruction is a public instruction to allow the manufacturer, part number and version of a component to be determined through the TAP. The ID Register configuration is as follows: • Bits 31–28: Version Information • Bits 27–12: Customer Part Number • Bits 11–1: Manufacturer Identity One application of the ID Register is to distinguish the manufacturer(s) of components on a board when multiple sourcing is used. included in the design. As required by the **IEEE** Std. 1149.1, the operation of the test logic has no effect on the operation of the internal system logic when the IDCODE instruction is selected. The value for the MSC8144E is 0x0188801D.
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 MSC8144E-based device is the device under test. When the current instruction selects the Bypass Register, the shift-register stage is set to a logic zero on the rising edge of TCK in the CAPTURE-DR controller state. Therefore, the first bit to be shifted out after the Bypass Register is selected is always a logic zero.
Clocking Control			
0x05	FREEZE	Yes	Private instruction that stops the clocks. The instructions causes all device clocks to stop. Resuming execution from this stopped state is not possible.

Table 25-3. Instruction Decoding (Continued)

Opcode	Instruction	Updates IR	Description
TLM			
0x03	TLM_SELECT	Yes	Private instruction that provides access to the TAP Linking Module.
0x04	TLM_HOLD	No	Private instruction that provides access to a TAP Linking Module. The most recent instruction that updated the Instruction Register is not overwritten by this instruction. Both the TLM Update Register and whatever else is selected by the instruction in the IR receive the TDI, but only the TLM Shift Register sends data to TDO.
OCE Instructions			
0xA0	ENABLE_ONCE	Yes	Not included in the **IEEE** Std. 1149.1. This public instruction allows you to perform system debug functions. When the ENABLE_ONCE instruction is decoded, TDI and TDO connect directly to the OCE registers. The OCE controller selects the specific OCE register connected between TDI and TDO, depending on the OCE instruction being executed. All communication with the OCE controller is through the SELECT-DR-SCAN path of the JTAG TAP Controller. Before the ENABLE_ONCE instruction is selected, the CHOOSE_ONCE instruction should be executed to define which OCE is to be activated. **Note:** This instruction is valid only if the core processor is running.
0xA1	DEBUG_REQUEST	Yes	Not included in the **IEEE** Std. 1149.1. This public instruction allows you to generate a debug request signal to the MSC8144E. 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 MSC8144E 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 SC3400 core enters the debug state. Monitor the core status to make sure that it has stopped.
0xA2	CHOOSE_ONCE	Yes	Not included in the **IEEE** Std. 1149.1. This instruction enables selected SC3400 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 MSC8144E uses JTAG TAP for standard defined testing compatibilities and for multi-core OCE module control and OCE module interconnection control. The MSC8144E has *four* internal OCE modules, one module per SC3400 core. The OCE modules interconnect in a chain and are configured and directed by the JTAG TAP controller. **Figure 25-4** shows the chained connection.

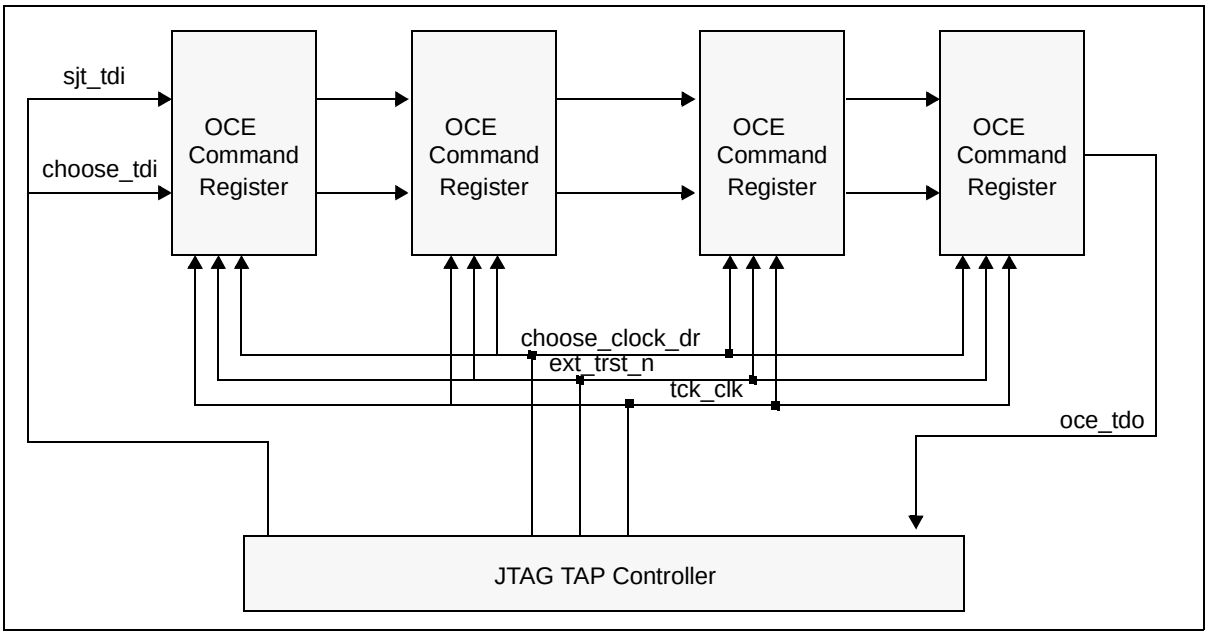


Figure 25-4. JTAG TAP Controller and OCE Module Multi-Core Interconnection

Each of the *four* MSC8144E OCE modules has an interface to a JTAG port. The interface is active even when a reset signal to the SC3400 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 *OCE Architecture 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 MSC8144E. The OCE modules selected by the CHOOSE_ONCE instruction are cascaded as shown in **Figure 25-5**. Only selected OCE modules respond to ENABLE_ONCE and DEBUG_REQUEST instructions from the JTAG. All OCE modules are deselected after reset.

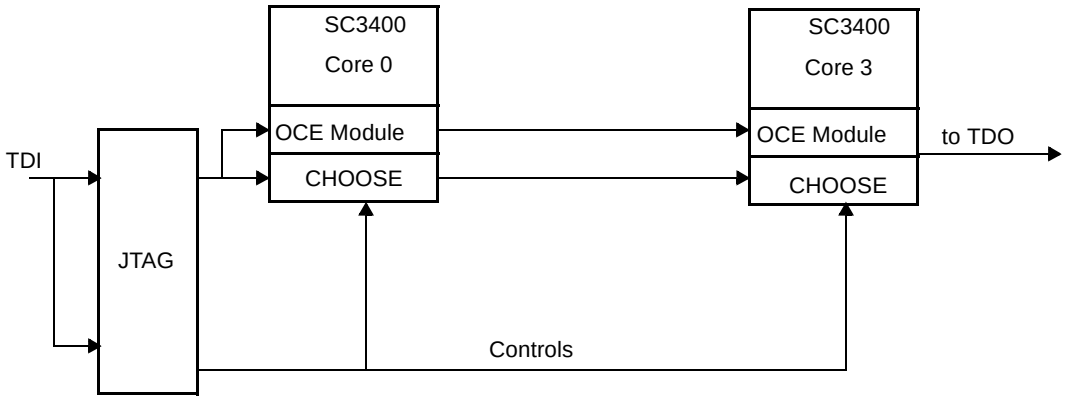


Figure 25-5. Cascading Multiple OCE Modules

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 SC3400 cores in the cascade, which is *four*. This state is indicated by the CHOOSE_CLOCK_DR signal. For example, for the four SC3400 cores on the MSC8144E, to activate the fourth core in the cascade, which is the closest to TDO and the farthest from TDI, the data is 1,0,0,0 (first a one, then three zeros). If the data is 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 SC3400 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 25-6** shows an example of CORE_CMD and RD_STATUS instructions.

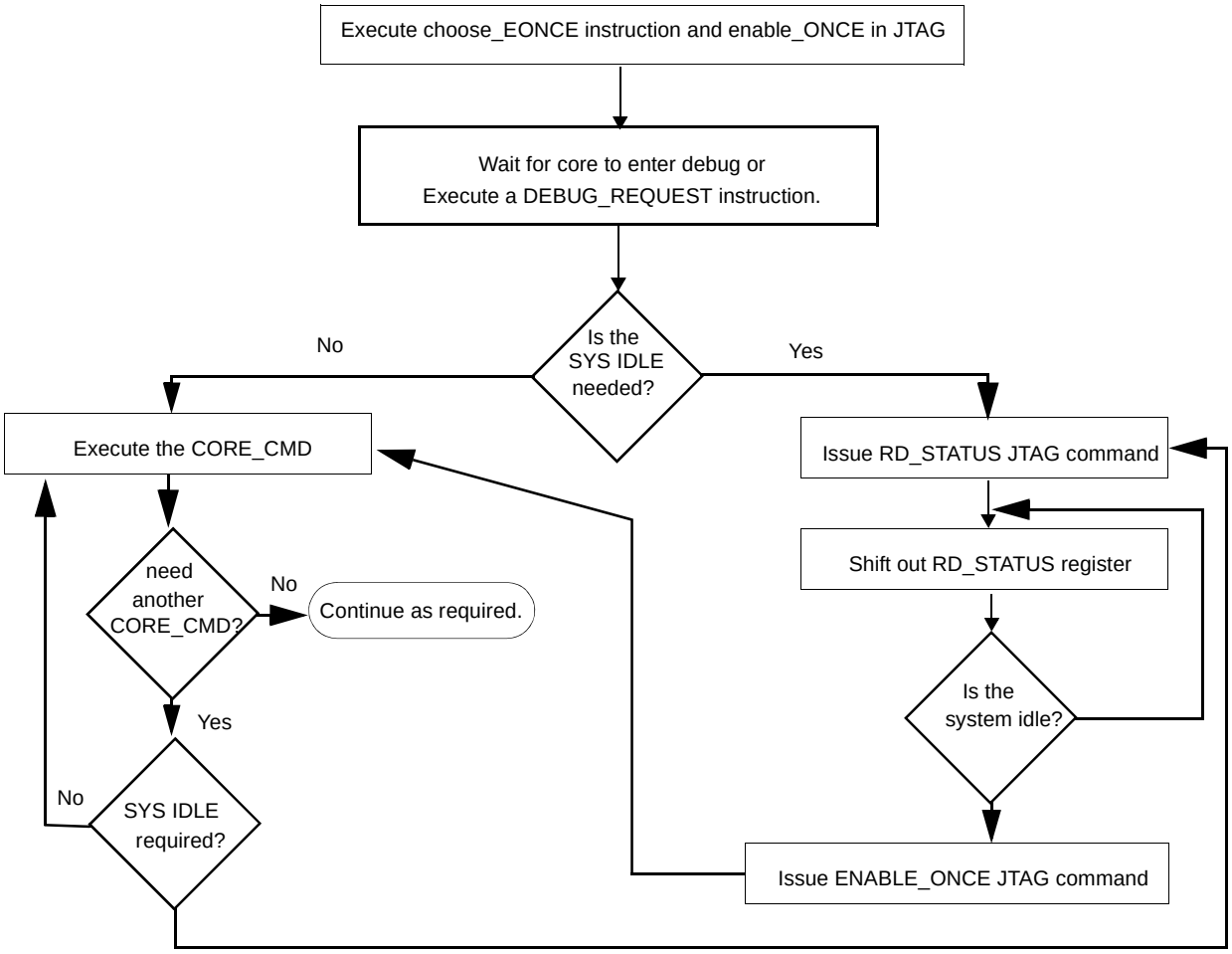


Figure 25-6. CORE_CMD and RD_STATUS Flow

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 25-7**.

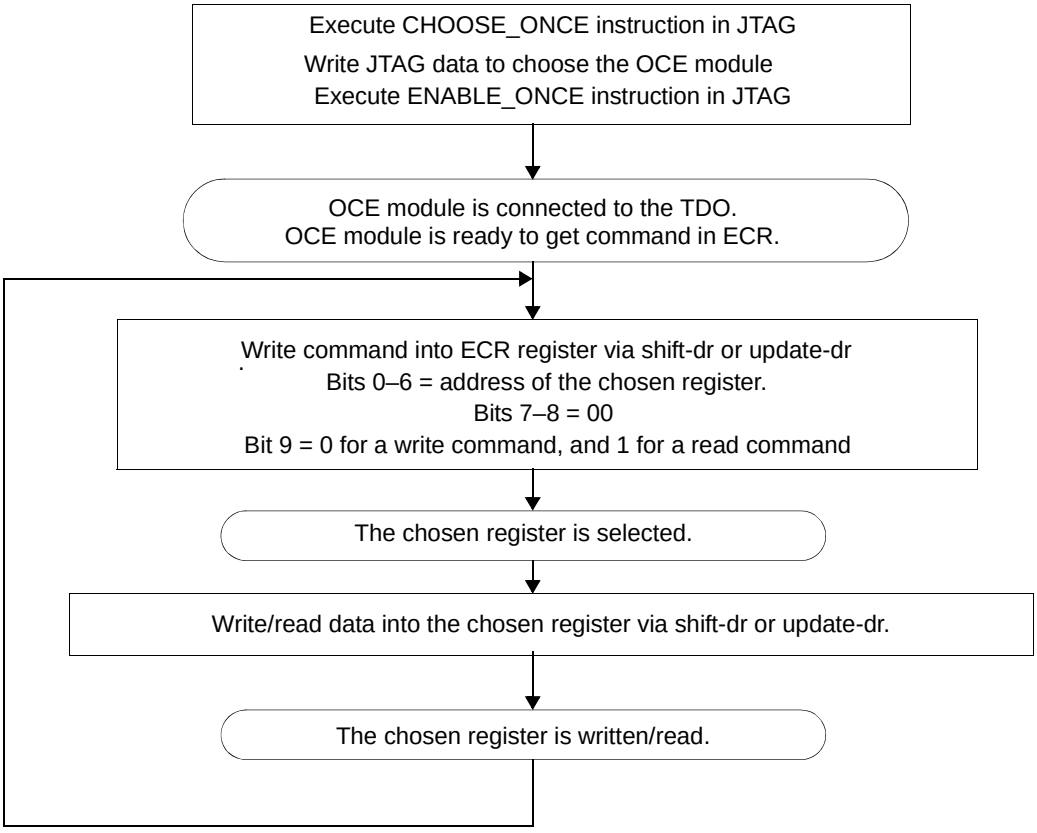


Figure 25-7. Reading and Writing OCE Registers Via the JTAG TAP

25.1.9 Signalling a Debug Request

The EE[0–1] signals connect to each of the MSC8144E 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 MSC8144E

The relevant paragraph from the **Emulation and Debug (OCE)** chapter of the *SC3400 Reference Manual* is reproduced here with the appropriate amendments.

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 SC3400 core into Debug mode (the default) right after the SC3400 core is reset. Holding EE0 at logic value 1 during and after the reset puts the SC3400 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 SC3400 core. If you want some SC3400 cores running and others in Debug mode, you must disable either the inputs of the running SC3400 cores or the outputs of the stopped SC3400 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 (00) 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. The MSC8144E only supports EE0 and EE1 signals. Also, there is a requirement to block triggering from EE0 if only some SC3400 cores must enter Debug mode. The EDCA control register can be used to enable the use of EE1 as the debug indicator. See the *OCE Reference Manual* for more information.

25.1.12 Real-Time Debug Request

All the SC3400 cores can enter Debug mode in several ways. The EE0 debug input request of all four SC3400 cores is wired to the output of an “OR” gate that sums the state of all EE1 outputs of the other SC3400 cores and the external EE0 signal (see **Figure 25-8**). Therefore, if any one SC3400 core sets its EE1 output (that is, enters Debug mode) or EE0 is asserted, the debug request input on all SC3400 cores is asserted. EE1 is activated when at least one of the SC3400 cores enters Debug mode.

Note: The EE0 input initiates Debug mode, and the EE1 output is the debug acknowledge indication.

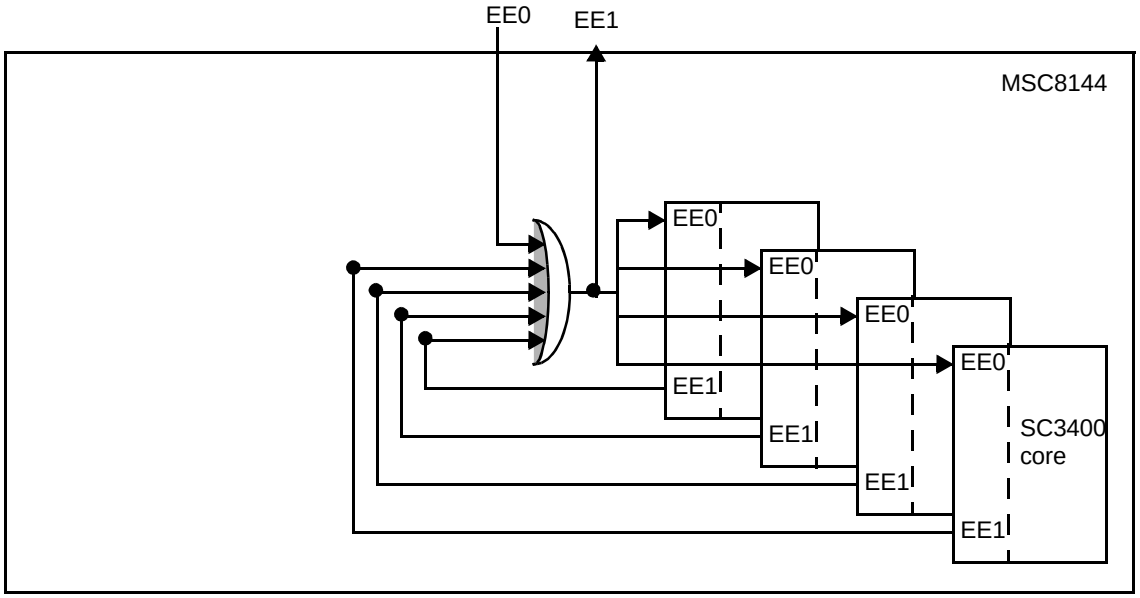


Figure 25-8. Selected SC3400 Core Issues a Debug Request to All Other SC3400 Cores

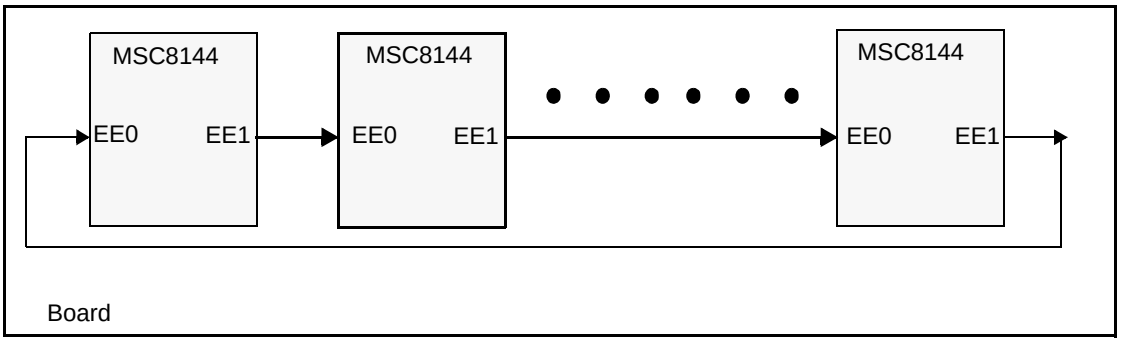


Figure 25-9. Board EE Signal Interconnectivity

25.1.13 Exiting Debug Mode

When an SC3400 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 SC3400 core OCE module is masked, preventing any more debug requests. When all the SC3400 cores exit Debug mode, the EE0 internal signals of all SC3400 cores are unmasked, enabling further debug requests. To restart the SC3400 cores, a **go** instruction is scanned into all four SC3400 cores. When the scan completes, the update launches all four SC3400 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 SC3400 cores are enabled via the CHOOSE_ONCE command, and then a **step** instruction is scanned into all four SC3400 cores. When the scan is done, the update launches all four SC3400 cores simultaneously. No retriggering occurs through EE0.

25.1.14 Debug Exception Request

After issuing a software debug instruction, initiate a handshake procedure. If the handshake procedure fails, the core is frozen. This should be handled in the same way as a timeout exception, that is, issue a user alert. Any debug exception issued while the core is frozen is ignored. Core status can be checked by selecting the core and executing a RD_STATUS command.

25.1.15 Tracing in the MSC8144E

The trace buffer in each OCE module in the MSC8144E is shared in a configurable memory space by configuring the TB address field the OCE Configuration (ECFG) Register. See the *OCE Reference Manual* for details. Use of a trace buffer requires special procedures, depending on the access used.

25.1.16 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 MSC8144E 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 MSC8144E 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.

25.1.17 JTAG and OCE Module Programming Model

25.1.17.1 Identification Register

The JTAG ID register is a 32-bit read-only factory-programmed register that distinguishes the component on a board according to the **IEEE** Std. 1149.1. The fields are defined as follows:

JTAGID JTAG Identification (ID) Register **JTAG port access only**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	1	0	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 MSC8144E first mask set is 0b0000.
- Design Center number is 0b000110.
- Sequence Number for the MSC8144E is 0b0010001000.
- Manufacture identity is 0b00000001110.
- The final 1 is required by the **IEEE** Std. 1149.1.

Note: The hexadecimal value stored in this register is 0x0188801D.

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.

25.1.17.2 Boundary Scan Register (BSR)

The MSC8144E BSR contains bits for most device signals and control signals. All MSC8144E 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 MSC8144E and are described by the BSDL file on the product website. **Figure 25-10** through **Figure 25-13** show various BSR cell types.

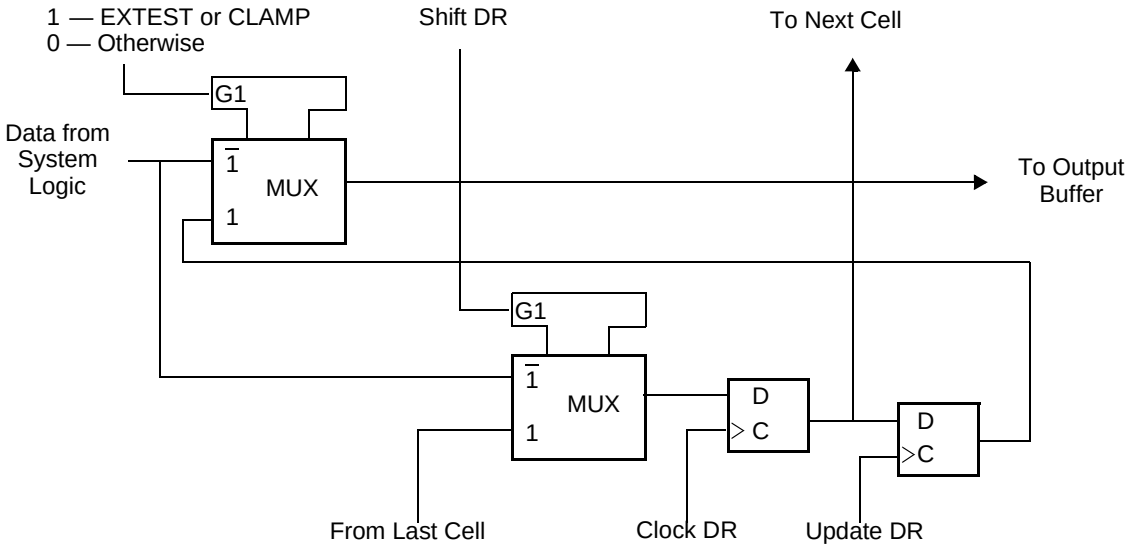


Figure 25-10. Output Signal Cell (O.PIN)

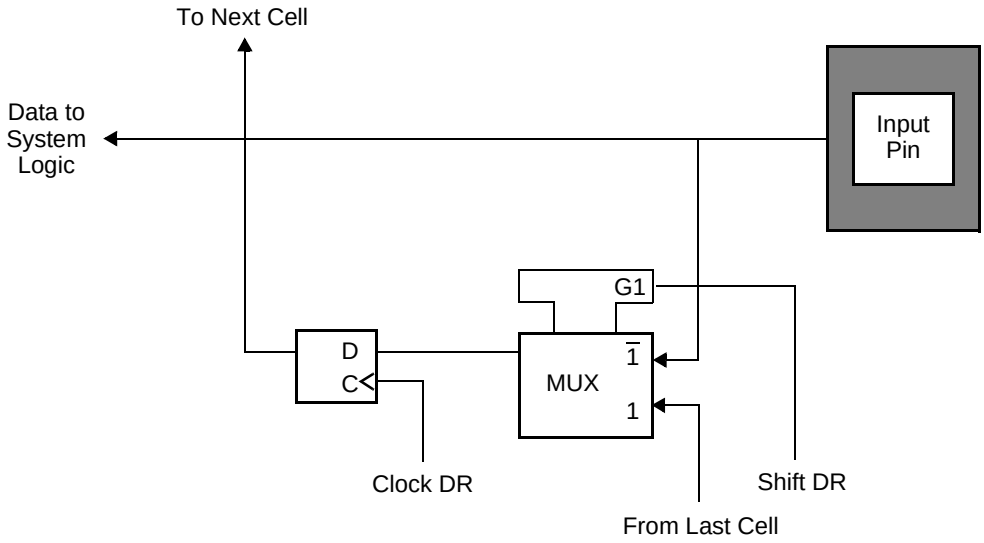


Figure 25-11. Observe-Only Input Signal Cell (I.OBS)

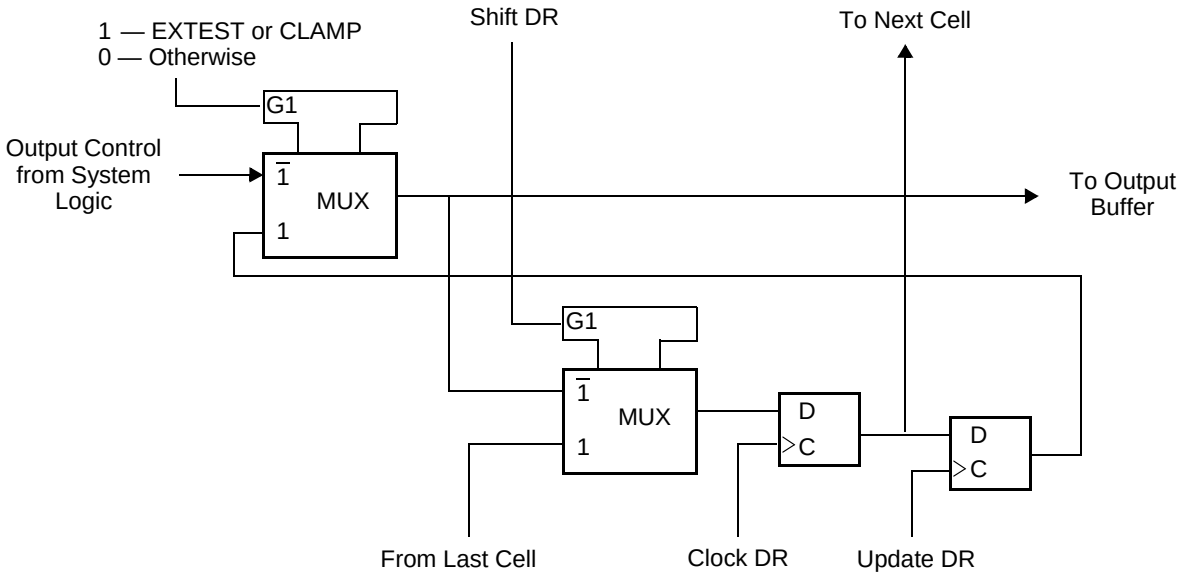


Figure 25-12. Output Control Cell (IO.CTL)

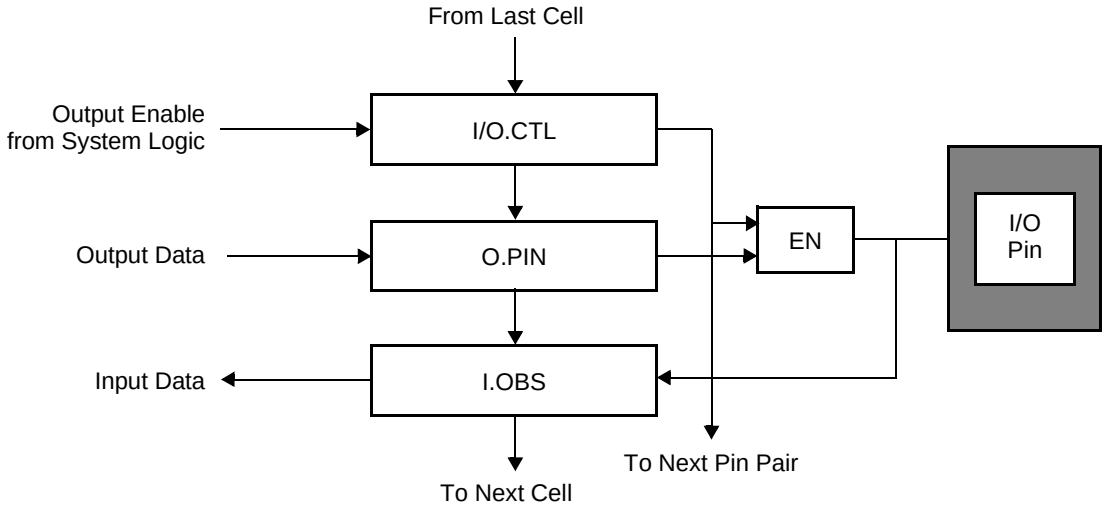


Figure 25-13. General Arrangement of Bidirectional Signal Cells

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 25-13**, and these bits are controlled by the cell shown in **Figure 25-12**. 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 MSC8144E cell types described in this file are depicted in **Figure 25-10** through **Figure 25-12**, which describe the cell structure for each type.

25.1.17.3 Shift Registers

The shift registers include the Bypass Register, General-Purpose Register (GPR), BSR, Identification Register, and Parallel Input register.

25.1.17.4 Bypass Register

The Bypass Register is a single-bit shift register (see **Figure 25-14**). 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.

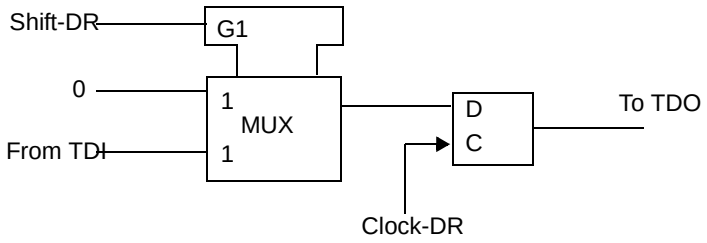


Figure 25-14. Bypass Register Configuration

25.1.17.5 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 25-15**.

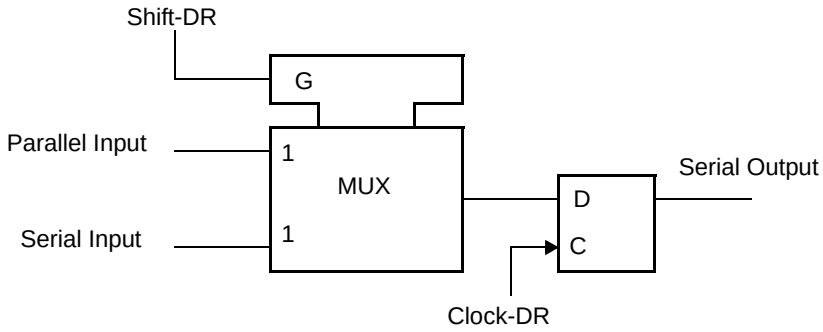


Figure 25-15. Identification Register Configuration (ID)

25.2 Debug and Profiling

The main debugging and profiling purposes are:

- Parallel counting of different events characterizing the operation of the MSC8144E device.
- Support for entering Debug mode upon detecting a predefined state of the MSC8144E 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 MSC8144E registers and memories in debug mode by a host processor.

25.2.1 Features

Table 25-6 describes MSC8144E device debug and profiling features

Table 25-6. MSC8144E Debug and Profiling Features

Block Name	Number of Blocks in MSC8144E	Supports Internal Debug	Supports Internal Profiling	Supports Profiling by PM Block	Supports Profiling and Debug by a CLASS Module
DSP core subsystem	4	+	+	+	+
L2 ICache (includes 2 classes)	1	+	+	+	+
DMA	1	+	—	+	+
QUICC Engine subsystem	1	+	+	—	+
Class	3	+	+	—	+
TDM	8	+	—	—	+
PCI	1	+	—	—	+
RapidIO complex	1	+	—	+	+
M2 memory	4	—	—	—	+
M3 memory	1	—	—	—	+
DDR memory	1	—	—	—	+

Other features:

- The performance monitor block supports counting of RapidIO, DMA, and L2 ICache 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 MSC8144E provides access for all peripherals (QUICC Engine subsystem, RapidIO, PCI, TDM, DMA, and UART) to all shared and CCSR memory space.

25.2.2 Entering Debug Mode

The individual modules have their own Debug state, which can be entered as follows:

- *DSP core subsystem*. Enters the Debug state when one of the following events occur:
  - Assertion of dedicated input signals (normally connected to the debugging agent)
  - Execution of the DEBUG or DEBUGEV instructions by the core.
  - A DPU event (depends on the configuration of the DPU and OCE).
  - Initiator or peripheral writes a certain value to the GCR2 control register.
- *L1 ICache and DCache*. 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 **Chapter 11**, *Internal Memory Subsystem* for details.

- *L2 ICache*. Activated only when the DSP core subsystems and L1 caches are in Debug state and certain values are written to respective control registers of the L2 ICache. In this mode, the internal state of the L2 ICache (tags, valid bits, PLRU table and cache array) can be read by the JTAG.

Note: See **Chapter 11**, *Internal Memory Subsystem* 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 ICache 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 SC3400 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 SC3400 Debug and Profiling

The MSC8144E device contains four extended DSP cores. Each DSP core subsystem supports the debug and profiling capabilities. When the DSP core subsystem is in the Debug state, the SC3400 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 **Section 10.6**, *Real-Time Debug Support*, on page 10-6 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 ICache and DCache 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 *MSC8144 MSC3400 Core Subsystem Reference Manual* for details.

25.2.6 L2 ICache Debug and Profiling

The L2 ICache enters its Debug state only after all four SC3400 cores and their L1 caches are in Debug mode when certain values are written to L2 respective control registers. In this mode, the internal state of the L2 ICache (tags, valid bits, PLRU table and cache array) can be read by the JTAG. The L2 ICache also includes two CLASS modules that support additional debug and profiling capabilities.

Note: See **Section 11.4**, *L2 Instruction Cache* for details.

25.2.6.1 CLASS Debug Profiling Units (CDPU) in the L2 ICache

The L2 ICache contains initiator and target CLASS modules. Each CLASS module supports debug and profiling measurements in its class debug profiling unit (CDPU) sub-block. In addition, the L2 ICache contains two 64-KB memory banks. Each bank supports debug and profiling measurements by the profiling monitor blocks. The main features for Target and Initiator class 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.
- CLASS profiling unit. The initiator and target CLASS profiling units provide the following profiling information for initiator and target L2 ICache buses:
  - Data acknowledges of read accesses.
  - Acknowledged accesses.
  - 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 in which the priority was not upgraded because the upgradeable signal was low.
  - Acknowledged read accesses.
- Over-flow mechanism.

Table 25-7. L2 Complex Initiator and Target Classes Debug Interrupts

DSP core subsystem Interrupt Number	Description of Interrupt
221	Watch Point mechanism interrupt in initiator CLASS.
223	Watch Point mechanism interrupt in target CLASS.
220	Over Flow mechanism interrupt in initiator CLASS.
222	Over Flow mechanism interrupt in target CLASS.

Note: See **Chapter 4**, *Chip-Level Arbitration and Switching System (CLASS)* for details.

25.2.6.2 L2 ICache Bank Profiling Events

- The L2 ICache provides this profiling information for each L2 ICache block (2 blocks):
  - Hit/miss/prefetch hit accesses.
  - Hit/miss/prefetch hit accesses in supervisor mode.
  - Hit/miss/prefetch hit accesses in user mode.
  - Hold cycles of cacheable accesses due to miss or pre-fetch hit.
  - Hold cycles of cacheable accesses due to miss or pre-fetch hit in supervisor mode.
  - Hold cycles of cacheable accesses due to miss or pre-fetch hit in user mode.
  - Thrash/miss accesses.
- All L2 ICache events connect to a performance monitory (PM) block. See **Section 25.3** for details.

25.2.7 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 MSC8144E can access the DMA controller PRAM through JTAG bus.

25.2.7.1 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 25-8**.

Table 25-8. 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.

25.2.7.2 Profiling Unit

The 16 channel DMA controller supports system level profiling. You can profile specific channels by configuring the desired channel number (CHA_NUM) and channel source or destination (DEST) fields in the DMA_LPCR. See **Section 14.5**, *Profiling* for details.

After configuration, all DMA events are connected to and monitored by the performance monitor (PM) block. See **Section 25.3**, *Performance Monitor* for details on how to use the information.

25.2.8 CLASS Modules

Each of the three CLASS modules includes the ability to generate interrupts and perform profiling.

25.2.8.1 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.

25.2.8.2 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 25-9. Class0, Class1 and Class2 debug interrupts

DSP core subsystem Interrupt Number	Description of Interrupt
205	Class0 watch point mechanism interrupt
208	Class1 watch point mechanism interrupt
211	Class2 watch point mechanism interrupt
204	Class0 over flow mechanism interrupt
207	Class1 over flow mechanism interrupt
210	Class2 over flow mechanism interrupt
209	Class1 error interrupt

Chapter 4, *Chip-Level Arbitration and Switching System (CLASS)* for details.

25.2.9 QUICC Engine Debug and Profiling

The QUICC Engine module has several means to debug and profile its operation as described in the following sections.

25.2.9.1 Trace Buffer

The QUICC Engine module provides chip-level testing capability through the trace buffer block (TRB). The TRB provides means of tracing the code ran by the CP (communication processor) in real time (through storing it non-intrusively) and reporting several internal CP/TRB events. The QUICC Engine module RISC has 4 kinds of breakpoints for on chip software debugging

- Instruction breakpoint
- Software breakpoint
- Data breakpoint
- External breakpoint

Note: See **Section 18.2**, *RISC Processors* for details.

25.2.9.2 Loopback Modes

SGMII and SMII support Loopback mode which connects the transmitter output internally to the receiver input (transmitter-to-receiver). The communication controllers support Diagnostic modes, including local and external loopback (transmitter-to-receiver), normal operation, and echo mode (receiver-to-transmitter). See **Section 20.4**, *Diagnostic Modes* for details.

25.2.10 TDM Debug and Profiling

The TDM modules provide a set of debug interrupts and loopback support for debugging.

25.2.10.1 Debug

The TDM complex in the MSC8144E device consists of eight TDM modules. Each TDM module supports transmit sync error and receive sync error interrupts. **Table 25-10** lists the TDM debug interrupts.

Table 25-10. TDM Debug Interrupts

DSP core subsystem Interrupt Number	Description of Interrupt
36	RX sync error interrupt
39	TX sync error interrupt
42	RX sync error interrupt
45	TX sync error interrupt
48	RX sync error interrupt
51	TX sync error interrupt
54	RX sync error interrupt
57	TX sync error interrupt
60	RX sync error interrupt
63	TX sync error interrupt
66	RX sync error interrupt
69	TX sync error interrupt
72	RX sync error interrupt
75	TX sync error interrupt
78	RX sync error interrupt
81	TX sync error interrupt

See **Chapter 19**, *TDM Interface* for details.

25.2.10.2 TDM Loopback Support

The TDM modules support loopback test mode in which the receiver receives the same data that is transmitted out. See **Section 19.5**, *Loopback Support* for details.

25.2.11 RapidIO Debug and Profiling

The RapidIO system debuggin system includes an error interrupt and the ability to connect to the profiling unit.

25.2.11.1 Debug Errors

The RapidIO system asserts the error interrupt listed in **Table 25-11** when a system error is detected.

Table 25-11. RapidIO Debug Interrupt

DSP core subsystem Interrupt Number	Description of Interrupt
90	RapidIO Errors interrupt

See **Chapter 16**, *Serial RapidIO® Controller* for details.

25.2.11.2 Profiling Unit

All RapidIO events can connect to Performance monitor (PM) block. See **Section 25.3**, *Performance Monitor* for details.

Note: The RapidIO complex frequency is 200 Mhz when PM block frequency is 400 Mhz (each RapidIO complex event is counted twice in the PM module).

25.2.12 PCI Debug

The PCI system supports an error interrupt to indicate that an address parity, data parity, or some other system error was detected. If any of these errors occur, the interrupt is asserted.

Table 25-12. PCI debug interrupts

DSP core subsystem Interrupt Number	Description of Interrupt
219	PCI Errors interrupt

See **Chapter 15**, *PCI* for details.

25.2.13 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.

25.2.14 Profiling Unit Programming Model

All DPU registers are memory-mapped and can be written or read by the core in Execution mode or through the OCE Core Command in Debug mode. Only one access per execution set to a DPU register is allowed in order to assure that the programming of the DPU registers occurs in a deterministic order. Reserved or unused bits in all registers should be written as zeros and the read value should be masked. Writing to unimplemented or read-only registers has no effect, and should be avoided for future software compatibility. Reading from unimplemented or write-only registers is illegal and produces undefined results.

Writing and reading of the DPU registers is done via the QBus. The DPU registers are located in Bank 0. This means that there are a number of cycles until the value is actually written to the DPU registers. In case DPU register synchronization is important, then the programming of the last DPU register should have a SYNCIO instruction in parallel. Code following this action can assume that the DPU registers written earlier were actually written. All registers are 32 bits with 16-bit accesses, which enable the use of bit-mask operations. When a 16-bit access is used on the 32-bit registers, the software address offset to the MSB part of the registers is equal to the software address offset of the LSB part + 2. The LSB part of the address is as shown in the registers memory map, and is not influenced by whether the system is Big Endian or Little Endian.) Any other access type other than word or long (such as byte, 2 long) should be avoided, and will result in an undefined result. When accessing the counter value registers, the 31 LSBs of the bus are written to the 31 LSBs of the register. Bit 31 of these registers is reserved.

- General Registers
  - DPU Control Register (DP_CR)
  - DPU Status Register (DP_SR)
  - DPU Monitor Register (DP_MR)
  - DPU PID Detection Reference Value Register (DP_RPID)
  - DPU DID Detection Reference Value Register (DP_RDID)
- General Counters
  - DPU Counter Triad A Control Register (DP_TAC)
  - DPU Counter Triad B Control Register (DP_TBC)
  - DPU Counter A0 Control Register (DP_CA0C)
  - DPU Counter A0 Value Register (DP_CA0V)
  - DPU Counter A1 Control Register (DP_CA1C)
  - DPU Counter A1 Value Register (DP_CA1V)
  - DPU Counter A2 Control Register (DP_CA2C)
  - DPU Counter A2 Value Register (DP_CA2V)
  - DPU Counter B0 Control Register (DP_CB0C)
  - DPU Counter B0 Value Register (DP_CB0V)
  - DPU Counter B1 Control Register (DP_CB1C)
  - DPU Counter B1 Value Register (DP_CB1V)

- DPU Counter B2 Control Register (DP_CB2C)
- DPU Counter B2 Value Register (DP_CB2V)

- Trace Buffer Registers
  - DPU Trace Control Register (DP_TC)
  - DPU VTB Start Address Register (DP_TSA)
  - DPU VTB End Address Register (DP_TEA)
  - DPU Trace Event Request Register (DP_TER)
  - DPU Trace Write Pointer Register (DP_TW)
  - DPU Trace Data Register (DP_TD)

Note: The DPU registers use the base address: 0xFFF0A000.

25.2.14.1 DPU Control Register (DP_CR)

DP_CR DPU Control Register Offset 0x00

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		TIDCM		ISEDCA5		ISEDCA4		ISEDCA3		ISEDCA2		ISEDCA1		ISEDCA0	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—	EIS	DETB		DECB2		DECB1		DECB0		DECA2		DECA1		DECA0	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_CR register is a 32-bit register responsible for controlling the debug logic of the DPU and the task ID comparator. **Table 25-13** defines the DP_CR bit fields.

Table 25-13. DP_CR Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
TIDCM 29–28	0	**Task ID Comparator Mask** Controls the operation of the task ID comparator, deciding which part takes part in the comparison. The reference program and data ID values are written in the DP_RPID and DP_RDID registers.	00 Neither the data task ID or the program task ID participate in the comparison. The comparison result is always 1. 01 The data task ID does not participate in the comparison (masked). 10 The program task ID does not participate in the comparison (masked). 11 Both the program task ID and the data task ID participate in the comparison.
ISEDCA5 27–26	0	**Interrupt Selector EDCA5** An event generated by the EDCA5 channel of the OCE causes interrupt Debug A or Debug B to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC

Table 25-13. DP_CR Bit Descriptions (Continued)

Name	Reset	Description	Settings
ISEDCA4 25–24	0	**Interrupt Selector EDCA4** An event generated by the EDCA4 channel of the OCE causes interrupt Debug A or Debug B to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
ISEDCA3 23–22	0	**Interrupt Selector EDCA3** An event generated by the EDCA3 channel of the OCE causes interrupt Debug A or Debug B to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
ISEDCA2 21–20	0	**Interrupt Selector EDCA2** An event generated by the EDCA2 channel of the OCE causes interrupt Debug A or Debug B to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
ISEDCA1 19–18	0	**Interrupt Selector EDCA1** An event generated by the EDCA1 channel of the OCE causes interrupt Debug A or Debug B to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
ISEDCA0 17–16	0	**Interrupt Selector EDCA0** An event generated by the EDCA0 channel of the OCE causes interrupt Debug A or Debug B to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
— 15	0	Reserved. Write to zero for future compatibility.	
EIS 14	0	**OCE Interrupt Selector** A maskable interrupt generated by the OCE is directed either to interrupt Debug A or Debug B.	0 A maskable interrupt generates Debug A 1 A maskable interrupt generates Debug B
DETB 13–12	0	**Trace Buffer Debug Request/Interrupt Enable** An event generated by the trace logic of the DPU causes a debug request to the OCE or an interrupt to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
DECB2 11–10	0	**Counter B2 Debug Request/Interrupt Enable** An event generated by counter B2 of the DPU causes a debug request to the OCE or an interrupt to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
DECB1 9–8	0	**Counter B1 Debug Request/Interrupt Enable** An event generated by counter B1 of the DPU causes a debug request to the OCE or an interrupt to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
DECB0 7–6	0	**Counter B0 Debug Request/Interrupt Enable** An event generated by counter B0 of the DPU causes a debug request to the OCE or an interrupt to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC

Table 25-13. DP_CR Bit Descriptions (Continued)

Name	Reset	Description	Settings
DECA2 5–4	0	**Counter A2 Debug Request/Interrupt Enable** An event generated by counter A2 of the DPU causes a debug request to the OCE or an interrupt to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
DECA1 3–2	0	**Counter A1 Debug Request/Interrupt Enable** An event generated by counter A1 of the DPU causes a debug request to the OCE or an interrupt to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
DECA0 1–9	0	**Counter A0 Debug Request/Interrupt Enable** An event generated by counter A0 of the DPU causes a debug request to the OCE or an interrupt to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC

25.2.14.2 DPU Status Register (DP_SR)

DP_SR DPU Status Register Offset 0x04

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—									TWBA	ENCB2	ENCB	ENCB0	ENCA2	ENCA1	ENCA0
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_SR is a 32-bit register that reflects the status of the DPU counters and if a trace write buffer (TWB) flush is in progress. All the bits are sticky status bits, and are read-only. Only the 6 lsb bits are used to enable the execution of bit-mask instructions.

Note: The ENCA and ENCB bits are used whenever the counter is enabled or disabled by events that occur after the DPU is initialized (for example, an EDCA0 event enables the counter and the DEBUGEV instruction disables it).

Table 25-14 defines the DP_SR bit fields.

Table 25-14. DP_SR Bit Descriptions

Name	Reset	Description	Settings
— 31–7	0	Reserved. Write to zero for future compatibility.	

Table 25-14. DP_SR Bit Descriptions (Continued)

Name	Reset	Description	Settings
TWBA 6	0	**Trace Write Buffer Active** The user can poll this bit to determine whether to disable tracing by setting the DP_TC[TMPDIS] bit or by clearing the DP_TC[EN] bit.	0 Trace Write Buffer flush is complete. 1 Flush sent to Trace Write Buffer.
ENCB2 5	0	**Counter B2 Enable** Indicates whether the counter is disabled or enabled.	0 Disabled. 1 Enabled.
ENCB1 4	0	**Counter B1 Enable** Indicates whether the counter is disabled or enabled.	0 Disabled. 1 Enabled.
ENCB0 3	0	**Counter B0 Enable** Indicates whether the counter is disabled or enabled.	0 Disabled. 1 Enabled.
ENCA2 2	0	**Counter A2 Enable** Indicates whether the counter is disabled or enabled.	0 Disabled. 1 Enabled.
ENCA1 1	0	**Counter A1 Enable** Indicates whether the counter is disabled or enabled.	0 Disabled. 1 Enabled.
ENCA0 0	0	**Counter A0 Enable** Indicates whether the counter is disabled or enabled.	0 Disabled. 1 Enabled.

25.2.14.3 DPU Monitor Register (DP_MR)

DP_MR DPU Status Register Offset 0x08

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—						TBF	DRA	—	DRTB	DRCB2	DRCB1	BRCB0	DRCA2	DRCA1	DRCA0
Type	R						W1C									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_MR is a 32-bit register that reflects information about an event generated by a counter or the trace buffer. All the bits are sticky bits, and can be only cleared by writing 1 to the appropriate bit(s). Only the 16 lsbs are used to enable the execution of bit-mask instructions. **Table 25-15** defines the DP_SR bit fields.

Table 25-15. DP_MR Bit Descriptions

Name	Reset	Description	Settings
— 31–10	0	Reserved. Write to zero for future compatibility.	

Table 25-15. DP_MR Bit Descriptions (Continued)

Name	Reset	Description	Settings
TBF 9	0	**Trace Buffer Full** Indicates whether the trace buffer is full.	0 Trace buffer not full. 1 Trace buffer full.
DRA 8	0	**Debug is External Asynchronous Debug Request** Indicates whether an external synchronous debug request.	0 No external synchronous debug request. 1 External synchronous debug request.
— 7	0	Reserved. Write to zero for future compatibility.	
DRTB 6	0	**Debug/Interrupt Reason is Trace Buffer** Indicates a trace buffer event debug request.	0 No trace buffer event debug request. 1 Trace buffer event debug request.
DRCB2 5	0	**Debug/Interrupt Reason is Counter B2 Event** Indicates a counter B2 Event debug request.	0 No counter B2 event debug request. 1 Counter B2 event debug request.
DRCB1 4	0	**Debug/Interrupt Reason is Counter B1 Event** Indicates a counter B1 Event debug request.	0 No counter B1 event debug request. 1 Counter B1 event debug request.
DRCB0 3	0	**Debug/Interrupt Reason is Counter B0 Event** Indicates a counter B0 Event debug request.	0 No counter B0 event debug request. 1 Counter B0 event debug request.
DRCA2 2	0	**Debug/Interrupt Reason is Counter A2 Event** Indicates a counter A2 Event debug request.	0 No counter A2 event debug request. 1 Counter A2 event debug request.
DRCA1 1	0	**Debug/Interrupt Reason is Counter A1 Event** Indicates a counter A1 Event debug request.	0 No counter A1 event debug request. 1 Counter A1 event debug request.
DRCA0 0	0	**Debug/Interrupt Reason is Counter A0 Event** Indicates a counter A0 Event debug request.	0 No counter A0 event debug request. 1 Counter A0 event debug request.

25.2.14.4 DPU PID Detection Reference Value Register (DP_RPID)

DP_RPID DPU PID Detection Reference Value Registers Offset 0x0C

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

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

The DP_RPID is a 32-bit register containing the reference program ID value for the task ID comparator. The DPU Control register controls the operation mode of the comparator. **Table 25-16** defines the DP_RPID bit fields.

Table 25-16. DP_RPID Bit Descriptions

Name	Reset	Description	Settings
— 31–8	0	Reserved. Write to zero for future compatibility.	
RPID 7–0	0	**Reference Program ID Value** Stores the value of the reference program ID.	

25.2.14.5 DPU DID Detection Reference Value Register (DP_RDID)

DP_RDID DPU DID Detection Reference Value Registers Offset 0x10

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

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

The DP_RDID is a 32- bit register containing the reference data ID value for the task ID comparator. The DPU Control register controls the operation mode of the comparator. **Table 25-17** defines the DP_RDID bit fields.

Table 25-17. DP_RDID Bit Descriptions

Name	Reset	Description	Settings
— 31–8	0	Reserved. Write to zero for future compatibility.	

Table 25-17. DP_RDID Bit Descriptions (Continued)

Name	Reset	Description	Settings
RDID 7–0	0	**Reference Data Task ID Value** Stores the value of the reference data task ID.	

25.2.14.6 DPU Counter Triad A Control Register (DP_TAC)

DP_TAC DPU Triad A Control Register Offset 0x20

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—		CEGP		—			CEG					—	CMODE		TCEN
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_TAC register is a 32-bit register that controls the operation of the DPU counter triad A, including what events and when they are counted. If the TCEN bit in the DP_TAC register is set, the appropriate counters are controlled by this register and ignore the programming of their own control register. When the TCEN bit is cleared, each counter is controlled individually by its own control register. **Table 25-21** defines the DP_TAC bit fields.

Table 25-18. DP_TAC Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
TDMP 29–28	0	**Triad Disable Mode Privilege Level** The event disabling the counters belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
TDM 27–24	0	**Triad Disable Mode** The event that disables the counters in the triad.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000–1111 reserved

Table 25-18. DP_TAC Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 23–22	0	Reserved. Write to zero for future compatibility.	
TENMP 21–20	0	**Triad Enable Mode Privilege Level** The event enabling the counters belongs to the task described by these bits. If the MARK instruction enables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
TENM 19–16	0	**Triad Enable Mode** The event that enables the counters.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	
CEGP 13–12	0	**Counted Event Group Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CEG 8–4	0	**Counted Event Group** The source counted by the counter.	See **Table 25-19** for details.
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Triad Counters Mode** Specifies the mode of the counter	00 One shot. Each counters in the triad generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. Each counter in the triad has its value saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10– 11 reserved.

Table 25-18. DP_TAC Bit Descriptions (Continued)

Name	Reset	Description	Settings
TCEN 0	0	**Triad Control Enable** Determines whether the three counters in the triad are controlled by this control register or their individual control counters.	0 Each counter is controlled independently. 1 The three counters are controlled by this register and the individual register settings have no effect.

Table 25-19. Counted Event Group

CEG Value	Group Name	Counter 0 Counts	Counter 1 Counts	Counter 2 Counts
00000	ICache hit-miss	ICache misses (without prefetch hits)	ICache hits	ICache prefetch hits
00001	DCache hit-miss	DCache misses (without prefetch hits)	DCache hits	DCache prefetch hits
00010	Core wait state	Clock cycles (non-debug)	Wait processing-state cycles	Not used
00011	Breakdown of application cycles - Group 1	Application cycles (non-debug, non-wait, non-stop)	No bubble	Bubble due to COF or interrupt
00100	Breakdown of application cycles - Group 2	Bubble due to starvation (no instructions in the prefetch/dispatch buffer)	Bubble due to core resource conflicts	Bubble due to data memory holds
00101	Not implemented in the MSC8144			
00110	Not implemented in the MSC8144			
00111	Hold associated with debug	Hold due to VTB writes	Hold due to internal freeze	not used
01000	Hold due to WRQ or WTB	Hold due to WRQ flush or atomic operation	Hold due to WRQ hazard	Hold due to WRQ or WTB full
01010	Hold due to Dcache system	Hold due to cacheable access (read, write-back miss or write-trough prefetch hit).	Hold due to non-cacheable access (read)	not used
01011–01110	Not implemented in the MSC8144			
10000	Instruction accesses to L2 subsystem	L2 instruction access miss	L2 instruction access hit	Total L2 instruction accesses
10001	Data accesses to L2 subsystem	L2 data access miss	L2 data access hit	Total L2 data accesses
10010	M2 RAM contentions (if L2 subsystem exists)	Contentions between IQ DQ accesses	Contentions between Q (IQ or DQ) and DMA accesses	not used
10011	M2 ROM contentions (if L2 subsystem exists)	Contentions between IQ DQ accesses	Contentions between Q (IQ or DQ) and DMA accesses	not used
10100	BTB Characterization Group 1	Total number of execution sets	Sequentially executed execution sets.	BTB-able instructions correctly predicted

Table 25-19. Counted Event Group (Continued)

CEG Value	Group Name	Counter 0 Counts	Counter 1 Counts	Counter 2 Counts
10101	BTB Characterization Group 2	BTB-able instructions not in the BTB, wrongly predicted.	BTB-able instructions, in the BTB, wrongly predicted.	Not BTB-able instructions
Note: Other combinations reserved				

25.2.14.7 DPU Counter Triad B Control Register (DP_TBC)

DP_TBC DPU Triad B Control Register Offset 0x24

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—		CEGP		—			CEG					—	CMODE		TCEN
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_TBC register is a 32-bit register that controls the operation of the DPU counter triad B, including what events and when they are counted. If the TCEN bit in the DP_TBC register is set, the appropriate counters are controlled by this register and ignore the programming of their own control register. When the TCEN bit is cleared, each counter is controlled individually by its own control register. **Table 25-21** defines the DP_TBC bit fields.

Table 25-20. DP_TBC Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
TDMP 29–28	0	**Triad Disable Mode Privilege Level** The event disabling the counters belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].

Table 25-20. DP_TBC Bit Descriptions (Continued)

Name	Reset	Description	Settings
TDM 27–24	0	**Triad Disable Mode** The event that disables the counters in the triad.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	
TENMP 21–20	0	**Triad Enable Mode Privilege Level** The event enabling the counters belongs to the task described by these bits. If the MARK instruction enables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
TENM 19–16	0	**Triad Enable Mode** The event that enables the counters.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	
CEGP 13–12	0	**Counted Event Group Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CEG 8–4	0	**Counted Event Group** The source counted by the counter.	See **Table 25-19** for details.

Table 25-20. DP_TBC Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Triad Counters Mode** Specifies the mode of the counter	00 One shot. Each counters in the triad generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. Each counter in the triad has its value saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10– 11 reserved.
TCEN 0	0	**Triad Control Enable** Determines whether the three counters in the triad are controlled by this control register or their individual control counters.	0 Each counter is controlled independently. 1 The three counters are controlled by this register and the individual register settings have no effect.

25.2.14.8 DPU Counter A0 Control Register (DP_CA0C)

DP_CA0C DPU Counter A0 Control Register Offset 0x2C

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

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

The DP_CA0C is a 32-bit register that controls the operation of the DPU Extension Support Counter A0, including what events and when they are counted. **Table 25-21** defines the DP_CA0C bit fields.

Table 25-21. DP_CA0C Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
CDMP 29–28	0	**Counter Disable Mode Privilege Level** The event disabling the counter belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].

Table 25-21. DP_CA0C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CDM 27–24	0	**Counter Disable Mode** The event that disables the counter.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	
CENMP 21–20	0	**Counter Enable Mode Privilege Level** The event enabling the counter belongs to the task described by these bits. If the MARK instruction enables the counter, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CENM 19–16	0	**Counter Enable Mode** The event that enables the counter.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	
CEP 13–12	0	**Counted Event Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	

Table 25-21. DP_CA0C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CE 8–4	0	**Counted Event** The source counted by the counter.	00000 Clock cycles (non-debug) 00001 Application cycles (non-wait, non-stop, non-debug 00010 Number of events generated by the EDCA0 of the OCE (CEP bits can be 00 or 01) 00011 Number of interrupts 00100 Number of ICache thrashes due to miss. 00101 Number of L2 ICache thrashes due to instruction access miss. 00110– 11111 reserved
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Counter Mode** Specifies the mode of the counter	00 One shot. The counter generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. The counter value is saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10– 11 reserved.
— 0	0	Reserved. Write to zero for future compatibility.	

25.2.14.9 DPU Counter A0 Value Register (DP_CA0V)

DP_CA0V DPU Counter A0 Value Registers Offset 0x30

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

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

DP_CA0V is a 32-bit register containing the specified counter value. **Table 25-22** defines the counter bit fields.

Table 25-22. DP_CA0V Bit Descriptions

Name	Reset	Description	Settings
— 31	0	Reserved. Write to zero for future compatibility.	
CV 30–0	0	**Counter Value** Stores the value of the counter.	

25.2.14.10 DPU Counter A1 Control Register (DP_CA1C)

DP_CA1C DPU Counter A1 Control Register Offset 0x34

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

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

The DP_CA1C is a 32-bit register that controls the operation of the DPU Extension Support Counter A1, including what events and when they are counted. **Table 25-23** defines the DP_CA1C bit fields.

Table 25-23. DP_CA1C Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	

Table 25-23. DP_CA1C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CDMP 29–28	0	**Counter Disable Mode Privilege Level** The event disabling the counter belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CDM 27–24	0	**Counter Disable Mode** The event that disables the counter.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	
CENMP 21–20	0	**Counter Enable Mode Privilege Level** The event enabling the counter belongs to the task described by these bits. If the MARK instruction enables the counter, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CENM 19–16	0	**Counter Enable Mode** The event that enables the counter.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	

Table 25-23. DP_CA1C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CEP 13–12	0	**Counted Event Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CE 8–4	0	**Counted Event** The source counted by the counter.	00000 Clock cycles (non-debug) 00001 Application cycles (non-wait, non-stop, non-debug 00010 Number of events generated by the EDCA1 of the OCE (CEP bits can be 00 or 01) 00011 Number of rollovers by counter A2 (CEP bits must be 00). 00100 Number of DCache thrashes due to miss. 00101 Number of L2 ICache thrashes due to access miss. 00110– 11111 reserved
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Counter Mode** Specifies the mode of the counter	00 One shot. The counter generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. The counter value is saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10– 11 reserved.
— 0	0	Reserved. Write to zero for future compatibility.	

25.2.14.11 DPU Counter A1 Value Registers (DP_CA1V)

DP_CA1V DPU Counter A1 Value Registers Offset 0x38

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

DP_CA1V is a 32-bit register containing the specified counter value. **Table 25-22** defines the counter bit fields.

Table 25-24. DP_CA1V Bit Descriptions

Name	Reset	Description	Settings
— 31	0	Reserved. Write to zero for future compatibility.	
CV 30–0	0	**Counter Value** Stores the value of the counter.	

25.2.14.12 DPU Counter A2 Control Register (DP_CA2C)

DP_CA2C DPU Counter A2 Control Register Offset 0x3C

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

The DP_CA2C is a 32-bit register that controls the operation of the DPU Extension Support Counter A2, including what events and when they are counted. **Table 25-25** defines the DP_CA2C bit fields.

Table 25-25. DP_CA2C Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	

Table 25-25. DP_CA2C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CDMP 29–28	0	**Counter Disable Mode Privilege Level** The event disabling the counter belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CDM 27–24	0	**Counter Disable Mode** The event that disables the counter.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	
CENMP 21–20	0	**Counter Enable Mode Privilege Level** The event enabling the counter belongs to the task described by these bits. If the MARK instruction enables the counter, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CENM 19–16	0	**Counter Enable Mode** The event that enables the counter.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	

Table 25-25. DP_CA2C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CEP 13–12	0	**Counted Event Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CE 8–4	0	**Counted Event** The source counted by the counter.	00000 Clock cycles (non-debug) 00001 Application cycles (non-wait, non-stop, non-debug 00010 Number of events generated by the EDCA2 of the OCE (CEP bits can be 00 or 01) 00011 Number of task switches; includes the number of times the service of a task started including the first time (CEP bits must be 00). 00100– 11111 reserved
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Counter Mode** Specifies the mode of the counter	00 One shot. The counter generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. The counter value is saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10 Extension mode. The counter rolls over when it reaches 0 and continues to count. Counter A1 can be programmed to count the number of overflows. In this case, counter A1 must operate in one-shot mode. 11 reserved.
— 0	0	Reserved. Write to zero for future compatibility.	

25.2.14.13 DPU Counter A2 Value Registers (DP_CA2V)

DP_CA2V DPU Counter A2 Value Registers Offset 0x40

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

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

DP_CA2V is a 32-bit register containing the specified counter value. **Table 25-22** defines the counter bit fields.

Table 25-26. DP_CA2V Bit Descriptions

Name	Reset	Description	Settings
— 31	0	Reserved. Write to zero for future compatibility.	
CV 30–0	0	**Counter Value** Stores the value of the counter.	

25.2.14.14 DPU Counter B0 Control Register (DP_CB0C)

DP_CB0C DPU Counter B0 Control Register Offset 0x54

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

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

The DP_CB0C is a 32-bit register that controls the operation of the DPU Extension Support Counter B0, including what events and when they are counted. **Table 25-29** defines the DP_CB0C bit fields.

Table 25-27. DP_CB0C Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	

Table 25-27. DP_CB0C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CDMP 29–28	0	**Counter Disable Mode Privilege Level** The event disabling the counter belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CDM 27–24	0	**Counter Disable Mode** The event that disables the counter.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	
CENMP 21–20	0	**Counter Enable Mode Privilege Level** The event enabling the counter belongs to the task described by these bits. If the MARK instruction enables the counter, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CENM 19–16	0	**Counter Enable Mode** The event that enables the counter.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	

Table 25-27. DP_CB0C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CEP 13–12	0	**Counted Event Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CE 8–4	0	**Counted Event** The source counted by the counter.	00000 Clock cycles (non-debug) 00001 Application cycles (non-wait, non-stop, non-debug 00010 Number of events generated by the EDCA3 of the OCE (CEP bits can be 00 or 01) 00011 Number of interrupts 00100 Number of ICache thrashes due to miss. 00101 Number of L2 ICache thrashes due to instruction access miss. 00110– 11111 reserved
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Counter Mode** Specifies the mode of the counter	00 One shot. The counter generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. The counter value is saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10– 11 reserved.
— 0	0	Reserved. Write to zero for future compatibility.	

25.2.14.15 DPU Counter B0 Value Registers (DP_CB0V)

DP_CB0V DPU Counter B0 Value Registers Offset 0x58

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

DP_CB0V is a 32-bit register containing the specified counter value. **Table 25-22** defines the counter bit fields.

Table 25-28. DP_CB0V Bit Descriptions

Name	Reset	Description	Settings
— 31	0	Reserved. Write to zero for future compatibility.	
CV 30–0	0	**Counter Value** Stores the value of the counter.	

25.2.14.16 DPU Counter B1 Control Register (DP_CB1C)

DP_CB1C DPU Counter B1 Control Register Offset 0x5C

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

The DP_CB1C is a 32-bit register that controls the operation of the DPU Extension Support Counter B1, including what events and when they are counted. **Table 25-29** defines the DP_CB1C bit fields.

Table 25-29. DP_CB1C Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	

Table 25-29. DP_CB1C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CDMP 29–28	0	**Counter Disable Mode Privilege Level** The event disabling the counter belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CDM 27–24	0	**Counter Disable Mode** The event that disables the counter.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	
CENMP 21–20	0	**Counter Enable Mode Privilege Level** The event enabling the counter belongs to the task described by these bits. If the MARK instruction enables the counter, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CENM 19–16	0	**Counter Enable Mode** The event that enables the counter.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	

Table 25-29. DP_CB1C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CEP 13–12	0	**Counted Event Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CE 8–4	0	**Counted Event** The source counted by the counter.	00000 Clock cycles (non-debug) 00001 Application cycles (non-wait, non-stop, non-debug 00010 Number of events generated by the EDCA4 of the OCE (CEP bits can be 00 or 01) 00011 Number of rollovers by counter B2 (CEP bits must be 00). 00100 Number of DCache thrashes due to miss. 00101 Number of L2 ICache thrashes due to access miss. 00110– 11111 reserved
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Counter Mode** Specifies the mode of the counter	00 One shot. The counter generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. The counter value is saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10– 11 reserved.
— 0	0	Reserved. Write to zero for future compatibility.	

25.2.14.17 DPU Counter B1 Value Registers (DP_CB1V)

DP_CB1V DPU Counter B1 Value Registers Offset 0x60

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

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

DP_CB1V is a 32-bit register containing the specified counter value. **Table 25-30** defines the counter bit fields.

Table 25-30. DP_CB1V Bit Descriptions

Name	Reset	Description	Settings
— 31	0	Reserved. Write to zero for future compatibility.	
CV 30–0	0	**Counter Value** Stores the value of the counter.	

25.2.14.18 DPU Counter B2 Control Register (DP_CB2C)

DP_CB2C DPU Counter B2 Control Register Offset 0x64

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

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

The DP_CB2C is a 32-bit register that controls the operation of the DPU Extension Support Counter B2, including what events and when they are counted. **Table 25-31** defines the DP_CB2C bit fields.

Table 25-31. DP_CB2C Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	

Table 25-31. DP_CB2C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CDMP 29–28	0	**Counter Disable Mode Privilege Level** The event disabling the counter belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CDM 27–24	0	**Counter Disable Mode** The event that disables the counter.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	
CENMP 21–20	0	**Counter Enable Mode Privilege Level** The event enabling the counter belongs to the task described by these bits. If the MARK instruction enables the counter, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CENM 19–16	0	**Counter Enable Mode** The event that enables the counter.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	

Table 25-31. DP_CB2C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CEP 13–12	0	**Counted Event Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CE 8–4	0	**Counted Event** The source counted by the counter.	00000 Clock cycles (non-debug) 00001 Application cycles (non-wait, non-stop, non-debug 00010 Number of events generated by the EDCA5 of the OCE (CEP bits can be 00 or 01) 00011 Number of task switches; includes the number of times the service of a task started including the first time (CEP bits must be 00). 00100– 11111 reserved
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Counter Mode** Specifies the mode of the counter	00 One shot. The counter generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. The counter value is saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10 Extension mode. The counter rolls over when it reaches 0 and continues to count. Counter B1 can be programmed to count the number of overflows. In this case, counter B1 must operate in one-shot mode. 11 reserved.
— 0	0	Reserved. Write to zero for future compatibility.	

25.2.14.19 DPU Counter B2 Value Registers (DP_CB2V)

DP_CB2V DPU Counter B2 Value Registers Offset 0x68

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

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

DP_CB2V is a 32-bit registers containing the specified counter value. **Table 25-32** defines the counter bit fields.

Table 25-32. DP_CA[0–2]V and DP_CB[0–2]V Bit Descriptions

Name	Reset	Description	Settings
— 31	0	Reserved. Write to zero for future compatibility.	
CV 30–0	0	**Counter Value** Stores the value of the counter.	

25.2.14.20 DPU Trace Control Register (DP_TC)

DP_TC DPU Trace Control Register Offset 0x7C

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—			TMPDIS	—		VTBWM		—	SAMPLE	—	TMODE				EN
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_TC is a 32-bit register responsible that controls the VTB memory. **Table 25-33** defines the DP_TC bit fields.

Table 25-33. DP_TC Bit Descriptions

Name	Reset	Description	Settings
— 31–27	0	Reserved. Write to zero for future compatibility.	

Table 25-33. DP_TC Bit Descriptions (Continued)

Name	Reset	Description	Settings
SHARE 26	0	**Share** Used when the DSP core subsystem operates in mixed endian mode. This bit allows the VTB to operate in a memory range shared with a system using little-endian mode.	0 Cannot share with system using little-endian mode. 1 Can share with system using little-endian mode.
— 25	0	Reserved. Write to zero for future compatibility.	
CSS 24–23	0	**Chip Select** The chip select attribute of the VTB write access.	00 M2 memory 01 Extension of M2 memory 10 M3 memory or L2 ICache 11 Reserved
— 22	0	Reserved. Write to zero for future compatibility.	
PRIV 21	0	**Privilege Mode** The privilege attribute of the VTB write access.	0 User 1 Supervisor
GLOBAL 20–18	0	**Global Attribute** The global attribute of the VTB write access	000 Cacheable write-through 001 Cacheable write-back 010 Non-cacheable write-through 011–111 reserved
BURST_ SIZE 17–16	0	**Burst Size** The burst size of the VTB write access.	00 1 VBR (16 bytes) 01 2 reserved 10 4 VBRs (64 bytes) 11 8 reserved
— 15–13	0	Reserved. Write to zero for future compatibility.	

Table 25-33. DP_TC Bit Descriptions (Continued)

Name	Reset	Description	Settings
TMPDIS 12	0	**Temporary Disable** To verify that all traced information arrived at the VTB, a flush can be generated that is similar to the flush generated when tracing is disabled. The only difference is that when tracing is disabled by setting the TMPDIS bit, when tracing is re -enabled by resetting this bit, the value of the Start Address is not sampled at the TB Write Pointer. The TB Write Pointer saves its previous value, thereby enabling tracing to be continued from the point where it was interrupted. If the tracing is enabled, TMPDIS bit must not be changed in debug mode by Core Command. The TMPDIS bit must not be set together with the EN bit. When the tracing is disabled by resetting the EN bit, the TMPDIS bit should not be set. When the TMPDIS bit is set, its resetting should not be done in the 16 VLES after the setting. Before the tracing is disabled by setting TMPDIS bit, the user has to wait until the TWB finishes the previous flush that can be initiated by a previous trace disabling or by entering wait or stop states. (Bit TWBA in the DP_SR register indicates if the TWB is in the middle of a flush operation.) In order not to lose information, serving an interrupt between the checking of the TWBA bit and the disabling should be avoided. (It can be done by disabling the interrupts before the checking of the TWBA bit and enabling them after the disabling.) When the TMPDIS bit is set, the trace logic disregards further inputs.	0 Trace logic enabled 1 Trace logic disabled
— 11–10	0	Reserved. Write to zero for future compatibility.	
VTBWM 9–8	0	**Virtual Trace Buffer Write Mode** Specifies the manner in which the VTB is written.	00 Overwrite mode. The DPU writes all the time. After writing to the End Address, the write pointer wraps to the first address and starts to overwrite the first entries. When disabled, it contains the last the entries leading to the disable point. (The TBF bit in the DP_MR register shows that the VTB was full at least once.) 01 One address mode. The DPU always writes to the same address (described by the Start Address). 10 Trace event request mode. When the write access is generated to the address equal to the value in the Trace Event Request, then depending on the programming of the DETB bit in the DP_CR register, either an interrupt to the EPIC or a debug request to the core (to the OCE) is generated. The debug request to the core may become an internal debug exception according to the programming of the OCE. 11 reserved
— 7	0	Reserved. Write to zero for future compatibility.	

Table 25-33. DP_TC Bit Descriptions (Continued)

Name	Reset	Description	Settings
SAMPLE 6	0	**Sample** A bit that can only be set. When set, the counter values are sampled to the trace buffer. This bit is only relevant when the TMODE bits are 0100. This bit is cleared by the DPU after writing the information to the trace buffer.	0 No action 1 Counter values are added to the trace buffer.
— 5	0	Reserved. Write to zero for future compatibility.	
TMODE 4–1	0	**Trace Mode** Identifies the source that is written to the trace buffer.	0000 Information generated by the OCE. Data compression is off. 0001 Information generated by the OCE after compression and adding task ID flags. 0010 Depending on the OCE programming, the data transferred at a task switch is either the first and last PC of the tasks together with a task flag and all six counter values; or, for a jump to a subroutine, a jump to an interrupt routine, or a return from either, the task ID and the values of all six counters. 0011 reserved 0100 When the SAMPLE bit in the Trace Buffer Control register is set by software at specified points during the application execution, the task ID and all six counters. 1010 The value written to the Trace Buffer Data register every time a new value is written. 0110 The task ID and the value of all six counters when EDCA5 generates an event that belongs to the task described by the task ID comparator. When the task is a user task, the task ID comparator must be programmed accordingly by the TIDCM bits in the DP_CR register. When the task is a supervisor level task or any task, the TIDCM bits in the DP_CR register must be 00. 0111– 1000 reserved

Table 25-33. DP_TC Bit Descriptions (Continued)

Name	Reset	Description	Settings
EN 0	0	**Enable** This bit is used to enable/disable the trace buffer. Use the following guidelines to enable/disable the trace buffer: • Always wait until any current flush operations are completed before disabling a trace operation. Poll the DP_SR[TWBA] bit to determine the flush status. • To prevent any interrupt servicing that may occur between reading the DP_SR[TWBA] bit and disabling the trace buffer, always disable the interrupts before reading the DP_SR[TWBA] bit and enable them only after disabling the trace buffer. • Never change the configuration of a trace operation during execution. Always disable the trace buffer first and then change the configuration.	0 Trace buffer disabled. 1 Trace buffer enabled

25.2.14.21 DPU VTB Start Address Register (DP_TSA)

DP_TSA DPU VTB Start Address Register Offset 0x80

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

The DP_TSA is a 32-bit register that contains the start address of the VTB (physical address). The VTB can be located only in Bank 3 (between addresses 0x00000000–0xFF000000). Alignment depends on the burst size:

- For a burst size of 1 VBR, TER must be aligned to the value: 32 × Burst Size (programmed in the DP_TC register).
- For a burst size of 4 VBRs, the value of TER can be 32 × (2 × n – 1), where n is a positive integer.

Note: Bits 4–0 must be written as zeros and are read as zeros.

Table 25-35 defines the DP_TSA bit fields.

Table 25-34. DP_TSA Bit Descriptions

Name	Reset	Description	Settings
SA 31–5	0	**Start Address** Specifies the end address of the VTB address range.	
— **4–0**	0	Reserved. Write to zero for future compatibility.	

25.2.14.22 DPU VTB End Address Register (DP_TEA)

DP_TEA DPU VTB End Address Register Offset 0x84

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

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

The DP_TEA is a 32-bit register that contains the end address of the VTB (physical address). The VTB can be located only in Bank 3 (between addresses 0x00000000–0xFF000000). Alignment depends on the burst size:

- For a burst size of 1 VBR, TER must be aligned to the value: 32 × Burst Size (programmed in the DP_TC register).
- For a burst size of 4 VBRs, the value of TER can be $32 \times (2 \times n - 1)$, where n is a positive integer.

Note: Bits 4–0 must be written as zeros and are read as zeros.

Table 25-35 defines the DP_TEA bit fields.

Table 25-35. DP_TEA Bit Descriptions

Name	Reset	Description	Settings
EA 31–5	0	**End Address** Specifies the end address of the VTB address range.	
— **4–0**	0	Reserved. Write to zero for future compatibility.	

25.2.14.23 DPU Trace Event Request Register (DP_TER)

DP_TER DPU Trace Event Request Register Offset 0x88

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

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

The DP_TER is a 32-bit register that is used when the VTB is written in Trace Event Request mode (programmed in the DP_TC[TMODE] field). The DP_TER register contains the address within the range of the VTB where an interrupt or debug request should be generated (depending on the programming of the DETB bit in the DP_CR register). Alignment depends on the burst size:

- For a burst size of 1 VBR, TER must be aligned to the value: 32 × Burst Size (programmed in the DP_TC register).
- For a burst size of 4 VBRs, the value of TER can be 32 × (2 × n – 1), where n is a positive integer.

Note: Bits 4–0 must be written as zeros and are read as zeros.

Table 25-38 defines the DP_TER bit fields.

Table 25-36. DP_TER Bit Descriptions

Name	Reset	Description	Settings
TER 31–5	0	**Trace Event Request** Specifies the address within the VTB address range where the interrupt or debug request is generated.	
— **4–0**	0	Reserved. Write to zero for future compatibility.	

25.2.14.24 DPU Trace Write Pointer Register (DP_TW)

DP_TW DPU Trace Write Pointer Register Offset 0x8C

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

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

The DP_TW is a 32-bit register that contains the pointer to the location in the VTB where the last entry was written by the DPU. When tracing is enabled (by setting the DP_TC[EN] bit), the value of the VTB Start Address register is sampled to it. When tracing is re-enabled (by resetting the DP_TC[TMPDIS] bit), the value of the TB Write Pointer does not change. If it is written by the user, its value must be aligned to the value “32 × Burst Size” programmed in the DP_TC register.

Note: Before changing the value of the DP_TW register, you must generate a flush by setting the DP_TC[TMPDIS] bit.

Table 25-38 defines the DP_TW bit fields.

Table 25-37. DP_TW Bit Descriptions

Name	Reset	Description	Settings
WP 31–5	0	**Write Pointer** Stores the pointer to the last entry written by the DPU to the VTB.	
— **4–0**	0	Reserved. Write to zero for future compatibility.	

25.2.14.25 DPU Trace Data Register (DP_TD)

DP_TD DPU Trace Data Register Offset 0x90

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

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

The DP_TD is a 32-bit register containing the value to be written to the TB. This value is only relevant when the MODE bits in the DP_TC register are 0101.

Table 25-38 defines the DP_TD bit fields.

Table 25-38. DP_TD Bit Descriptions

Name	Reset	Description	Settings
TBD 31–0	0	**Trace Buffer Data** Stores the data to write to the Trace Buffer.	

25.3 Performance Monitor

The MSC8144E includes a performance monitor facility that can be used to monitor and record selected behaviors of the integrated device. **Section 25.3.1.6** 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 MSC8144E 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 25-16 is a high-level block diagram of the performance monitor. The module consists of a global control register (PMGC), one 64-bit counter (PMC0), eight 32-bit counters, and two control registers per counter. The global control register PMGC affects all counters and takes priority over local control registers. The local control registers are divided into two groups, as follows:

- Local control A registers control counter freezing, overflow condition enable, and event selection. Local control register PMLCA0, which controls counter PMC0, does not contain event selection because PMC0 counts only cycles.
- Local control B registers control the start and stop triggering, contain the counters' threshold values. Local control register PMLCB0, which controls PMC0, does not contain threshold information because PMC0 only counts cycles.

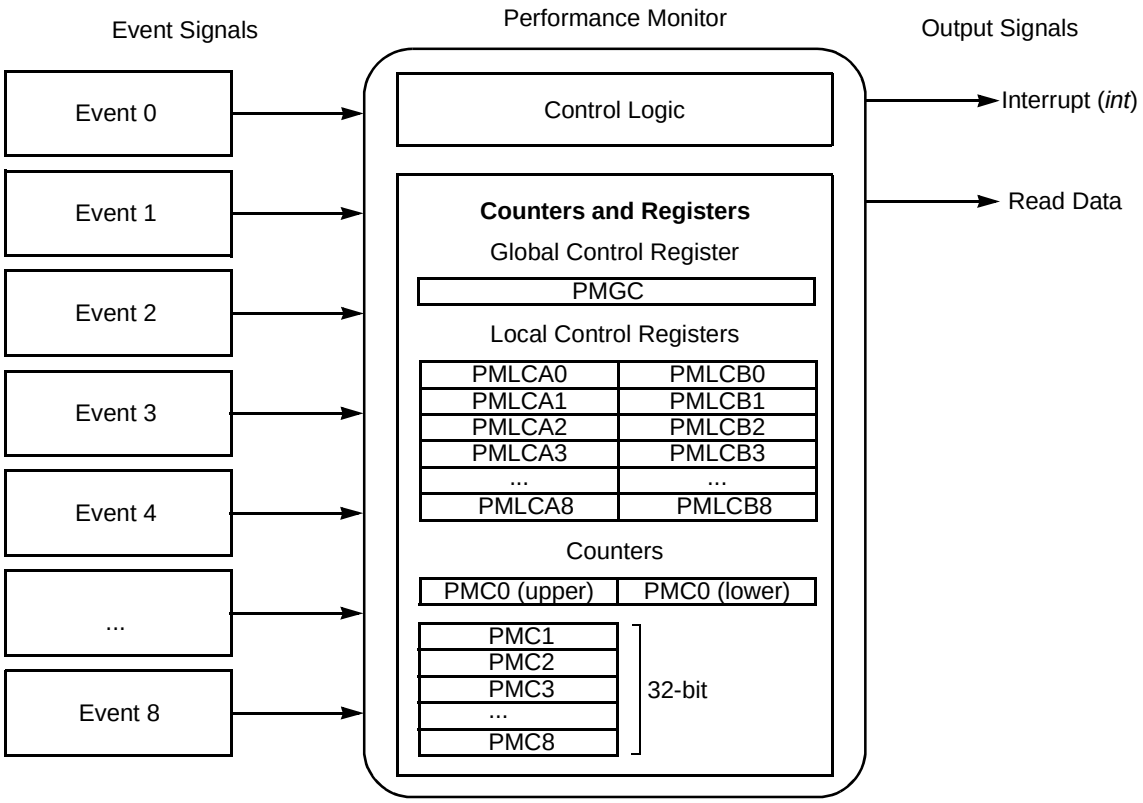


Figure 25-16. Performance Monitor Block Diagram

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 MSC8144E 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.

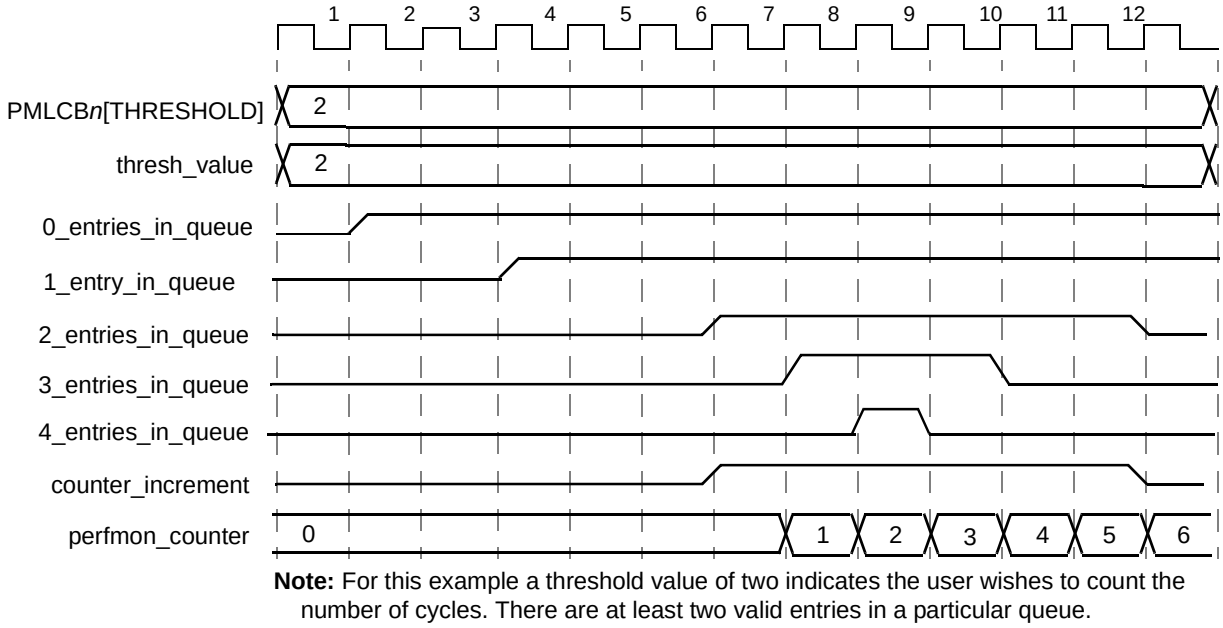
25.3.1.2 Event Counting

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 25-17**.



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 25-17. 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 25-39**, 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 Triggering

Triggering allows one counter to start or stop counting on the change of another counter or on the overflow of another counter. More specifically, if PMC1 is set to start or stop counting as a result of a change or overflow in counter PMC2, then counter PMC2 must be identified in the local control register of counter PMC1. This is done by appropriately setting the trigger-on select bit or trigger-off select bit (PMLCB1[TRIGOFFSEL] or PMLCB1[TRIGONSEL]). Additionally, the condition that triggers the counter must be selected by configuring the corresponding control bits (PMLCB1[TRIGONCNTL] or PMLCB1[TRIGOFFCNTL]). Assuming the counter is enabled by other control register settings, the counter increments (or freezes) when its specified event occurs after the trigger-on (or off) condition occurs. When trigger on and trigger off are both selected, the trigger-off condition is ignored until the trigger-on condition has occurred. Furthermore, when a trigger-off condition occurs, the counter state is preserved; it is not restarted by subsequent trigger-on conditions. Triggering is disabled when the counter trigger-select bits specify itself as the trigger source. Similarly, triggering is disabled when the trigger control bits are cleared.

25.3.1.6 Performance Monitor Events

Table 25-39 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 (PIC interrupt wait cycles).

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 25-39** are reserved.

Note: Each RapidIO event is counted twice because RapidIO frequency is 200 MHz and PM frequency is 400 MHz.

Table 25-39. 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 (DSP core subsystem) clock cycles
L2 ICache Events		
Hit in L2 ICache bank 1 in user mode	Ref:10	-
Hit in L2 ICache bank 2 in user mode	Ref:11	-
Miss in L2 ICache bank 1 in user mode	C1:57	-
Miss in L2 ICache bank 2 in user mode	C2:0	-
Prefetch hit in L2 ICache bank 1 in user mode	C3:60	-
Prefetch hit in L2 ICache bank 2 in user mode	C4:0	-
Hold cycles in L2 ICache bank 1 in user mode	C5:56	-
Hold cycles in L2 ICache bank 2 in user mode	C6:0	-
Thrash miss in L2 ICache bank 1	C7:57	miss which cause to line thrashing in L2 ICache
Thrash miss in L2 ICache bank 2	C8:0	miss which cause to line thrashing in L2 ICache
Hit in L2 ICache bank 1 in supervisor mode	Ref:12	-
Hit in L2 ICache bank 2 in supervisor mode	Ref:13	—
Hold cycles in L2 ICache bank 1 in supervisor mode	C4:2	—
Hold cycles in L2 ICache bank 2 in supervisor mode	C5:1	—
Miss in L2 ICache bank 1 in supervisor mode	C3:2	—
Miss in L2 ICache bank 2 in supervisor mode	C4:3	—
Prefetch hit in L2 ICache bank 1 in supervisor mode	C7:0	—
Prefetch hit in L2 ICache bank 2 in supervisor mode	C8:1	—
DMA Events		
Asserted for every cycle DMA specific channel is active	C5:2	
Asserted for fourth cycles when DMA specific channel win arbitration	Ref:14	-
Asserted for one cycle when DMA finished buffer	C6:1	—
	C6:2	
DMA specific channel request active in the system (Valid until get address acknowledge)	C7:1	-
Asserted for fourth cycles when DMA specific channel get consecutive grant	C8:2	-
RapidIO RMU EVENTS		
Packet received of any priority	C4:4	
Packet received of priority 0	C2:3	
Packet received of priority 1	C5:3	

Table 25-39. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Packet received of priority 2	C1:36	
Packet received of priority 3	C3:43	
Clock cycle occurred in which the inbound buffer is full to any priority (measured from when SOP received to buffer release transferred on OCN). Event asserted for as many clock cycles as this is true.	C2:45	
Clock cycle occurred in which the inbound buffer is full to priority 0 (measured from when SOP received to buffer release transferred on OCN). Event asserted for as many clock cycles as this is true.	C4:44	
Clock cycle occurred in which the inbound buffer is full to priority 1 (measured from when SOP received to buffer release transferred on OCN). Event asserted for as many clock cycles as this is true.	C5:5	
Clock cycle occurred in which the inbound buffer is full to priority 2 (measured from when SOP received to buffer release transferred on OCN). Event asserted for as many clock cycles as this is true.	C6:4	
Clock cycle occurred in which the inbound buffer is full to priority 3 (measured from when SOP received to buffer release transferred on OCN). Event asserted for as many clock cycles as this is true.	C7:3	
Clock cycle occurred in which an OCN tag is unavailable, any priority. Event asserted for as many clock cycles as this is true.	C8:3	
Clock cycle occurred in which an OCN tag is unavailable, priority 0. Event asserted for as many clock cycles as this is true.	C3:3	
Clock cycle occurred in which an OCN tag is unavailable, priority 1. Event asserted for as many clock cycles as this is true.	C4:5	
Clock cycle occurred in which an OCN tag is unavailable, priority 2. Event asserted for as many clock cycles as this is true.	C5:4	
OCN reorder occurred.	C6:3	
OCN reorder occurred, reordered packet was priority 0.	C7:2	
OCN reorder occurred, reordered packet was priority 1.	C1:1	
OCN reorder occurred, reordered packet was priority 2.	C2:4	
OCN reorder occurred reordered packet was priority 3.	C8:4	
Packet sent to OCN	C5:7	
Packet sent to OCN of priority 0	C6:6	
Packet sent to OCN of priority 1	C7:5	
Packet sent to OCN of priority 2	C1:7	
Packet sent to OCN of priority 3	C2:11	

Table 25-39. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Inbound buffer utilization. When this event is selected, each bit corresponds to the number of inbound buffers being used. Each bit will be asserted for as long as the corresponding number of buffers is used.	Ref:33	
Inbound buffer, priority 0 utilization. When this event is selected, each bit corresponds to the number of priority 0 inbound buffer being used. Each bit will be asserted for as long as the corresponding number of buffers is used.	Ref:31	
RapidIO Delta Events		
A received packet has been sent to OCN for passthrough	C3:11	
OCN reorder occurred.	C4:12	
OCN reorder occurred, priority 0.	C5:8	
OCN reorder occurred, priority 1.	C6:7	
OCN reorder occurred, priority 2.	C7:6	
OCN reorder occurred priority 3.	C8:5	
Clock cycle occurred in which an OCN tag is unavailable, any priority. Event asserted for as many clock cycles as this is true.	C1:59	
Clock cycle occurred in which an OCN tag is unavailable, priority 0. Event asserted for as many clock cycles as this is true.	C2:63	
Clock cycle occurred in which an OCN tag is unavailable, priority 1. Event asserted for as many clock cycles as this is true.	C3:62	
Clock cycle occurred in which an OCN tag is unavailable, priority 2. Event asserted for as many clock cycles as this is true.	C4:60	
Clock cycle occurred in which an OCN tag is unavailable, priority 3. Event asserted for as many clock cycles as this is true.	C5:57	
Packet sent to RapidIO	C6:61	
Packet was misaligned	C7:58	
Packet was retried	C8:55	
Packet was reordered	C1:8	
Packet sent to RapidIO of priority 0	C2:12	
Packet sent to RapidIO of priority 1	C3:12	
Packet sent to RapidIO of priority 2	C4:13	
Packet sent to RapidIO of priority 3	C5:9	
Clock cycle occurred in which the outbound buffer is full to any priority. Event asserted for as many clock cycles as this is true.	C6:8	
Clock cycle occurred in which the outbound buffer is full to priority 0. Event asserted for as many clock cycles as this is true.	C7:7	

Table 25-39. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Clock cycle occurred in which the outbound buffer is full to priority 1. Event asserted for as many clock cycles as this is true.	C8:6	
Clock cycle occurred in which the outbound buffer is full to priority 2. Event asserted for as many clock cycles as this is true.	C5:58	
Clock cycle occurred in which the outbound buffer is full to priority 3. Event asserted for as many clock cycles as this is true.	C8:56	
Clock cycle occurred in which a RapidIO tag is unavailable, any priority. Event asserted for as many clock cycles as this is true.	C6:62	
Clock cycle occurred in which a RapidIO tag is unavailable, priority 0. Event asserted for as many clock cycles as this is true.	C7:59	
Clock cycle occurred in which a RapidIO tag is unavailable, priority 1. Event asserted for as many clock cycles as this is true.	C8:57	
Clock cycle occurred in which a RapidIO tag is unavailable, priority 2. Event asserted for as many clock cycles as this is true.	C6:9	
Clock cycle occurred in which a misaligned RapidIO tag is unavailable, any priority. Event asserted for as many clock cycles as this is true.	C7:56	
Clock cycle occurred in which a misaligned RapidIO tag is unavailable, priority 0. Event asserted for as many clock cycles as this is true.	C4:11	
Clock cycle occurred in which a misaligned RapidIO tag is unavailable, priority 1. Event asserted for as many clock cycles as this is true.	C2:62	
Clock cycle occurred in which a misaligned RapidIO tag is unavailable, priority 2. Event asserted for as many clock cycles as this is true.	C3:10	
Packet accepted on RapidIO	C8:10	
Packet accepted from RapidIO of priority 0	C1:9	
Packet accepted from RapidIO of priority 1	C2:13	
Packet accepted from RapidIO of priority 2	C3:13	
Packet accepted from RapidIO of priority 3	C4:14	
Non idles received. This can be used to determine the RapidIO link utilization. This is actually 1/2 of the actual count.	C4:17	
Packet retry occurred due to inbound buffer limitations for any priority	C5:11	
Packet retry occurred due to inbound buffer limitations, priority 0	C6:11	
Packet retry occurred due to inbound buffer limitations, priority 1	C7:9	
Packet retry occurred due to inbound buffer limitations, priority 2	C8:8	
Packet retry occurred due to inbound buffer limitations, priority 3	C7:10	

Table 25-39. 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.	C6:12	
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.	C5:12	
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.	C3:15	
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.	C2:15	
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.	C1:11	
Non idles transmitted. This can be used to determine the RapidIO link utilization. This is actually 1/2 of the actual count.	C8:9	
NBOUND buffer utilization. When this event is selected, each bit corresponds to an inbound buffer being used. Each bit will be asserted for as long as the corresponding buffer is used.	Ref:34	
Chaining Events		
PMC0 carry-out	Ref:1	PMC0[0] 1-to-0 transitions.
PMC1 carry-out	Ref:2	PMC1[0] 1-to-0 transitions. Reserved for PMC1.
PMC2 carry-out	Ref:3	PMC2[0] 1-to-0 transitions. Reserved for PMC2.
PMC3 carry-out	Ref:4	PMC3[0] 1-to-0 transitions. Reserved for PMC3.
PMC4 carry-out	Ref:5	PMC4[0] 1-to-0 transitions. Reserved for PMC4.
PMC5 carry-out	Ref:6	PMC5[0] 1-to-0 transitions. Reserved for PMC5.
PMC6 carry-out	Ref:7	PMC6[0] 1-to-0 transitions. Reserved for PMC6.
PMC7 carry-out	Ref:8	PMC7[0] 1-to-0 transitions. Reserved for PMC7.
PMC8 carry-out	Ref:9	PMC8[0] 1-to-0 transitions. Reserved for PMC8.

25.3.1.7 Performance Monitor Examples

Table 25-41 contains sample register settings for the four supported modes.

- Simple event performance monitoring example
- Triggering event performance monitoring example
- Threshold event performance monitoring example

The settings in **Table 25-40** are identical for all four examples.

Table 25-40. 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 the triggering example any event can be selected in PMLCA*n*[EVENT]. All other features are disabled by clearing these register fields except for CE to allow interrupt signalling. If PMLCB*n*[TRIGONSEL] is 3 and PMLCB*n*[TRIGOFFSEL] is 5, the counter begins and ends counting based on the conditions in counters three and five. Furthermore, if PMLCB*n*[TRIGONCNTL] is 1, the counter begins counting when PMC3 changes value. According to the setting in PMLCB*n*[TRIGOFFCNTL], the counter ends counting when PMC5 overflows. Also, although the register settings for PMC5 is not shown, PMLCA*n*[CE] for this counter must be cleared so that interrupt signalling is not enabled and the counter does not freeze when it overflows.

For threshold counting, a threshold event must be specified in PMLCA*n*[EVENT] and threshold value in PMLCB*n*[THRESHOLD].

Table 25-41. Register Settings for Counting Examples

Register	Register Field	Simple Event	Triggering	Threshold
PMGC	FAC	0	0	0
	PMIE	1	1	1
	FCECE	1	1	1
PMLCA*n*	FC	0	0	0
	CE	1	1	1
	EVENT	121	68	33

Table 25-41. Register Settings for Counting Examples (Continued)

Register	Register Field	Simple Event	Triggering	Threshold
PMLCB*n*	TRIGONSEL	0	3	0
	TRIGOFFSEL	0	5	0
	TRIGONCNTL	0	1	0
	TRIGOFFCNTL	0	2	0
	THRESHOLD	0	0	3

The performance monitor must be reset before event counting sequences. The performance monitor can be reset by first freezing one or more counters and then clearing the freeze condition to allow the counters to count according to the settings in the performance monitor registers. Counters can be frozen individually by setting PMLCAn[FC] bits, or simultaneously by setting PMGC[FAC]. Simply clearing these freeze bits will then allow the performance monitor to begin counting based on the register settings.

Note that using PMLCAn[FC] to reset the performance monitor resets only the specified counter. Performance monitor registers can be configured through reads or writes while the counters are frozen as long as freeze bits are not cleared by the register accesses.

25.3.2 Performance Monitor Programming Model

The performance monitor system includes the following registers:

- Performance Monitor Global Control Register (PMGC), see **page 25-80**.
- Performance Monitor Local Control Register A0 (PMLCA0), see **page 25-81**.
- Performance Monitor Local Control Register A[1–8] (PMLCA[1–8]), see **page 25-82**.
- Performance Monitor Local Control Register B0 (PMLCB0), see **page 25-83**.
- Performance Monitor Local Control Register B[1–8] (PMLCB[1–8]), see **page 25-84**.
- Performance Monitor Counter 0 (PMC0), see **page 25-85**.
- Performance Monitor Counter 1–8 (PMC[1–8]), see **page 25-86**.

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: 0xFFFB0100.

25.3.2.1 Performance Monitor Global Control Register (PMGC)

PMGCR 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 25-42** defines the PMGC bit fields.

Table 25-42. 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 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 25-43** defines the PMLCA0 bit fields.

Table 25-43. PMLCA0 Bit Descriptions

Name	Reset	Description	Settings
FC 31	0	**Freeze Counter 0** Enables or freezes PMC0.	0 PMC0 increments if permitted by other control bits. 1 PMC0 disabled and does not increment.
— 30–27	0	Reserved. Write to zero for future compatibility.	
CE 26	0	**Condition Enable** Controls the counter 0 overflow condition. This bit should be cleared when PMC0 is used as a trigger or is selected for chaining. An overflow condition occurs when PMC0[msb] is set	0 Overflow cannot occur. PMC0 cannot cause interrupts or freeze counters. 1 Overflow conditions are enabled and can generate interrupts and freeze counters.
— 25–0	0	Reserved. Write to zero for future compatibility.	

25.3.2.3 Performance Monitor Local Control An

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 25-44** defines the PMLCA[1–8] bit fields.

Table 25-44. PMLCA[1–8] Bit Descriptions

Name	Reset	Description	Settings
FC 31	0	**Freeze Counter 0** Enables or freezes PMC0.	0 PMC0 increments if permitted by other control bits. 1 PMC0 disabled and does not increment.
— 30–27	0	Reserved. Write to zero for future compatibility.	
CE 26	0	**Condition Enable** Controls the counter 0 overflow condition. This bit should be cleared when PMCn is used as a trigger or is selected for chaining. **Note:** An overflow condition occurs when PMCn[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.
— 25–23	0	Reserved. Write to zero for future compatibility.	
EVENT 22–16	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.
— 15–0	0	Reserved. Write to zero for future compatibility.	

25.3.2.4 Performance Monitor Local Control B0

PMLCB0 Performance Monitor Local Control B0 Offset 0x14

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

PMLCB0, along with PMLCA0, configures the PMC0 operation. **Table 25-45** defines the PMLCB0 bit fields.

Table 25-45. PMLCB0 Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
TRIGONSEL 29–26	0	**Trigger On Select** Specifies the number of the counter that triggers event counting to start. When the specified counter's TRIGONCNTL event overflows, the current counter begins counting. No triggering occurs when TRIGONSEL = current counter value.	
— 25–24	0	Reserved. Write to zero for future compatibility.	
TRIGOFFSEL 23–20	0	**Trigger Off Select** Specifies the number of the counter that triggers event counting to stop. When the specified counter's TRIGONCNTL event overflows, the current counter stops counting. No triggering occurs when TRIGOFFSEL = current counter.	
TRIGONCNTL 19–18	0	**Trigger On Control** Indicates the condition under triggering to start counting occurs.	00 Trigger off (no start trigger). 01 Trigger on change. 10 Trigger on overflow. 11 Reserved.
TRIGOFFCNTL 17–16	0	**Trigger Off Control** Indicates the condition under triggering to stop counting occurs.	00 Trigger off (no stop trigger). 01 Trigger on change. 10 Trigger on overflow. 11 Reserved.
— 15–0	0	Reserved. Write to zero for future compatibility.	

25.3.2.5 Performance Monitor Local Control Bn

PMLCB[1–8] Performance Monitor Local Control B[1–8] Offset 0x14 + n*0x10

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

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

PMLCB[1–8], along with PMLCA[1–8], configure the respective PMC[1–8] operation. **Table 25-46** defines the PMLCB[1–8] bit fields.

Table 25-46. PMLCB[1–8] Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
TRIGONSEL 29–26	0	**Trigger On Select** Specifies the number of the counter that triggers event counting to start. When the specified counter's TRIGONCNTL event overflows, the current counter begins counting. No triggering occurs when TRIGONSEL = current counter value.	
— 25–24	0	Reserved. Write to zero for future compatibility.	
TRIGOFFSEL 23–20	0	**Trigger Off Select** Specifies the number of the counter that triggers event counting to stop. When the specified counter's TRIGONCNTL event overflows, the current counter stops counting. No triggering occurs when TRIGOFFSEL = current counter.	
TRIGONCNTL 19–18	0	**Trigger On Control** Indicates the condition under triggering to start counting occurs.	00 Trigger off (no start trigger). 01 Trigger on change. 10 Trigger on overflow. 11 Reserved.
TRIGOFFCNTL 17–16	0	**Trigger Off Control** Indicates the condition under triggering to stop counting occurs.	00 Trigger off (no stop trigger). 01 Trigger on change. 10 Trigger on overflow. 11 Reserved.
— 15–6	0	Reserved. Write to zero for future compatibility.	
THRESHOLD 5–0	0	**Threshold** Sets the counting threshold level. Only event with a number of occurrence greater than this number are counted.	

25.3.2.6 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 25-47** defines the PMC0 bit fields.

Table 25-47. PMC0 Bit Descriptions

Name	Reset	Description	Settings
PMC0 63–0	0	**Performance Monitor Counter 0** Contains the current PMC0 count.	

25.3.2.7 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 25-48** defines the PMC[1–8] bit fields.

Table 25-48. PMC[1–8] Bit Descriptions

Name	Reset	Description	Settings
PMCn 31–0	0	**Performance Monitor Counter n** Contains the current PMCn count.	

Security Engine (SEC) 26

The Security Coprocessor version 2.1 (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), and the **IEEE** 802.11i™ security standard. 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 six execution units (EUs) with a shared random number generator (RNG) that use a common interface to the controller. The EUs perform the specific mathematical manipulations required by protocols used in cryptographic processing.

26.1 Architecture Overview

Because the SEC can act as a master on the internal system bus, it does not impose the core processing data movement bottleneck normally associated with slave-only processors. In addition, the SEC resides in 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.

26.1.1 Functional Diagram

A functional diagram of the SEC internal architecture is shown in **Figure 26-1**. The controller block is designed to transfer 64-bit sets between the bus and any register inside the SEC.

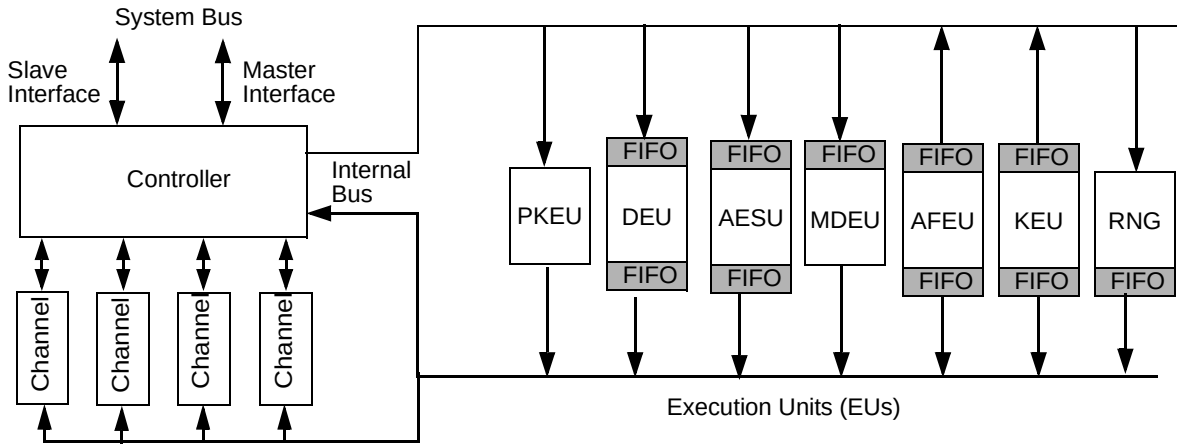


Figure 26-1. SEC Functional Diagram

An SEC operation begins when the core processor writes a descriptor pointer to the Fetch FIFO in one of the four SEC channels. This write operation uses the slave interface (in which the core processor is master and SEC is the slave). From this point, 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. Next the channel requests that the controller fetch the keys, context, and data from locations specified in the rest of the descriptor. 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. The channel requests that the controller write data from the output FIFOs and registers back to system memory.

The channel indicates completion by sending a descriptor via an interrupt to the core processor or by a writeback of the descriptor header into core processor memory. For details about this signaling, see **Section 26.1.3**, *Channels*. When a descriptor indicates completion, 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. The SEC then 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 has been fetched. Similarly, the channel directs the controller to write bursts of output whenever enough accumulates in the execution unit output FIFO.

26.1.2 Controller

The controller manages internal device resources, including the individual execution units (EUs), FIFOs, the master and slave interfaces to the device system bus, and the internal buses that connect all the various modules. The controller receives service requests from the core processor via the slave interface and from the channels and schedules the required activities.

26.1.2.1 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.

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 allows it to achieve performance unattainable on secondary buses.

26.1.2.1.1 Channel-Controlled Access

Processing begins when a descriptor pointer is written to the Fetch FIFO of one of the channels. Based on the services requested by the descriptor header, the channel asks the controller to assign the necessary EUs to that channel. If all appropriate EUs are already reserved by other channels, the channel stalls and waits to fetch data until an appropriate EU is available. If multiple channels simultaneously request the same EU, the EU is assigned on a weighted priority or round-robin basis.

Once the required EU has been reserved, the channel requests that the controller fetch and load the appropriate data. The controller acts as a master on the system bus, reading and writing on byte boundaries. The channel operates the EU, and makes further requests to the controller to write output data to system memory. When the descriptor processing is complete, the channel asks the controller to release the EU for use by other channels.

26.1.2.1.2 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.

26.1.2.2 Descriptors and Link Tables

The SEC controller 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. A sample descriptor is described in **Table 26-1**.

Table 26-1. 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
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

Each descriptor contains eight 64-bit 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.

Note: For details, see **Section 26.2.1.1**, *Descriptors*, on page 26-9.

26.1.3 Channels

The SEC includes four channels that manage data and EU function. Each channel contains the following:

- *Fetch FIFO*. Holds a queue of pointers to descriptors waiting to be serviced.
- *Configuration register*. Allows the user a number of options for SEC event signalling.
- *Control registers*. Information about the transaction in process.
- *Status register*. Indicates the last unfulfilled bus request.
- *Descriptor buffer memory*. Stores the active descriptor.

26.1.3.1 Channel Servicing

Whenever a channel is idle and its Fetch FIFO is non-empty, the channel reads the next descriptor pointer from the Fetch FIFO. Using this pointer, the channel fetches the descriptor and places it in its descriptor buffer. To service 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 EUs from the controller.
2. Wait for the controller to grant access to the required EUs.
3. Set the appropriate mode bits in the EU(s) for the required service.
4. Fetch *data parcels* using pointers from the descriptor buffer, and place them in either an EU input FIFO or EU registers (as appropriate). The term data 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.
5. If the data size is greater than EU FIFO size, continue fetching input data and writing output data to memory.
6. Wait for EU(s) to complete processing.
7. Upon completion, unload results from output FIFOs and Context Registers and write them to external memory using pointers in the descriptor buffer.
8. If multiple services are requested, go back to step 4. If not, continue to step 9.
9. Release the EUs.
10. If *done notification* is enabled, perform this notification.

26.1.3.2 Channel Completion

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. 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.

26.1.3.3 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). 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.

Note: For details on configuring signaling, see **Section 26.5.5.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4])*, on page 26-89. For detail on the writeback fields, see **Section 26.2.1.1.2**, *Descriptor Header*, on page 26-11.

26.1.3.4 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 more than one EU. 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.

26.1.3.5 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, the secondary EU is always the MDEU. For details, see **Section 26.3**, *Channels*, on page 26-20.

Note: All SEC memory transactions are snooped by the device coherency module. This is part of the SEC interface and requires no user intervention.

26.1.4 Common EU Interface

The controller and the channels use a standardized common interface to the EUs. As shown in **Figure 26-2**, 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.

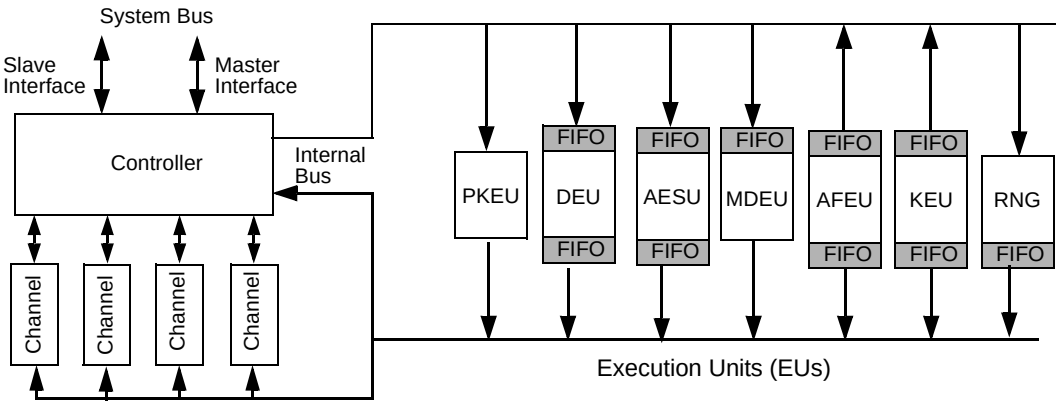


Figure 26-2. SEC Functional Diagram

26.1.5 Execution Units (EUs)

The EUs are compatible with IPSec, IKE, SSL/TLS, iSCSI, SRTP, and **IEEE** Std. 802.11i processing, and can work together to perform high-level cryptographic tasks. The SEC EUs include the following:

- PKEU for computing asymmetric key operations, including modular exponentiation (and other modular arithmetic functions) or ECC point arithmetic.
- DEU for performing block-cipher symmetric-key cryptography using DES and 3DES.
- AESU for performing the Advanced Encryption Standard algorithm.
- MDEU for performing security hashing using MD-5, SHA-1, SHA-224, or SHA-256.
- AFEU for performing RC-4 compatible stream cipher symmetric key cryptography.
- KEU for performing 3GPP confidentiality and integrity (F8 and F9) algorithms.
- RNG for random number generation.

Each EU is described in detail in **Section 26.4**, *Execution Units*, on page 26-23.

26.2 SEC Controller

The controller within the SEC is responsible for overseeing the operations of the execution units (EUs), the interface to the core processor, and the management of the channels. The controller interfaces to the core processor via the master and slave buses and to the channels and EUs via internal buses. All transfers between the core processor and the EUs are moderated by the controller. Some of the main functions of the controller are as follows:

- Maintain a correct data parcel flow
- Control the internal bus accesses to the EUs
- Provide arbitration and control for bus accesses
- Provide arbitration for channels requesting EUs and assign EUs to channels
- Monitor interrupts from channels and pass them to a core processor
- Realign read and write data using the proper byte alignment

26.2.1 Operation Definition and Data Control

The controller uses a combination of structures and registers to control the flow of data parcels. Descriptors define the required security operations and specify how the data parcels are manipulated. Each descriptor consists of one header and seven pointers. These can be combined with link tables to allow concatenation of specific data blocks in memory.

Figure 26-3 illustrates various ways that descriptors and link tables specify data parcels:

- The first pointer in the descriptor specifies Parcel A using the simplest method—the parcel is specified directly through Pointer0 and Length0.
- The next pointer 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 that identify data segments to concatenate. The last 4 bytes 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.
- One pointer in a descriptor can be used to specify multiple parcels. Pointer2 and Length2 specify Parcel C, then Parcel D follows immediately afterward with length specified by Extent2. Pointer3 is used for three data parcels (E, F, and G), this time using link tables.

For any sequence of data 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 appropriate error state in the Channel Pointer Status Register—G-STATE for gather error or S-STATE for scatter error (see **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4])*, on page 26-92).

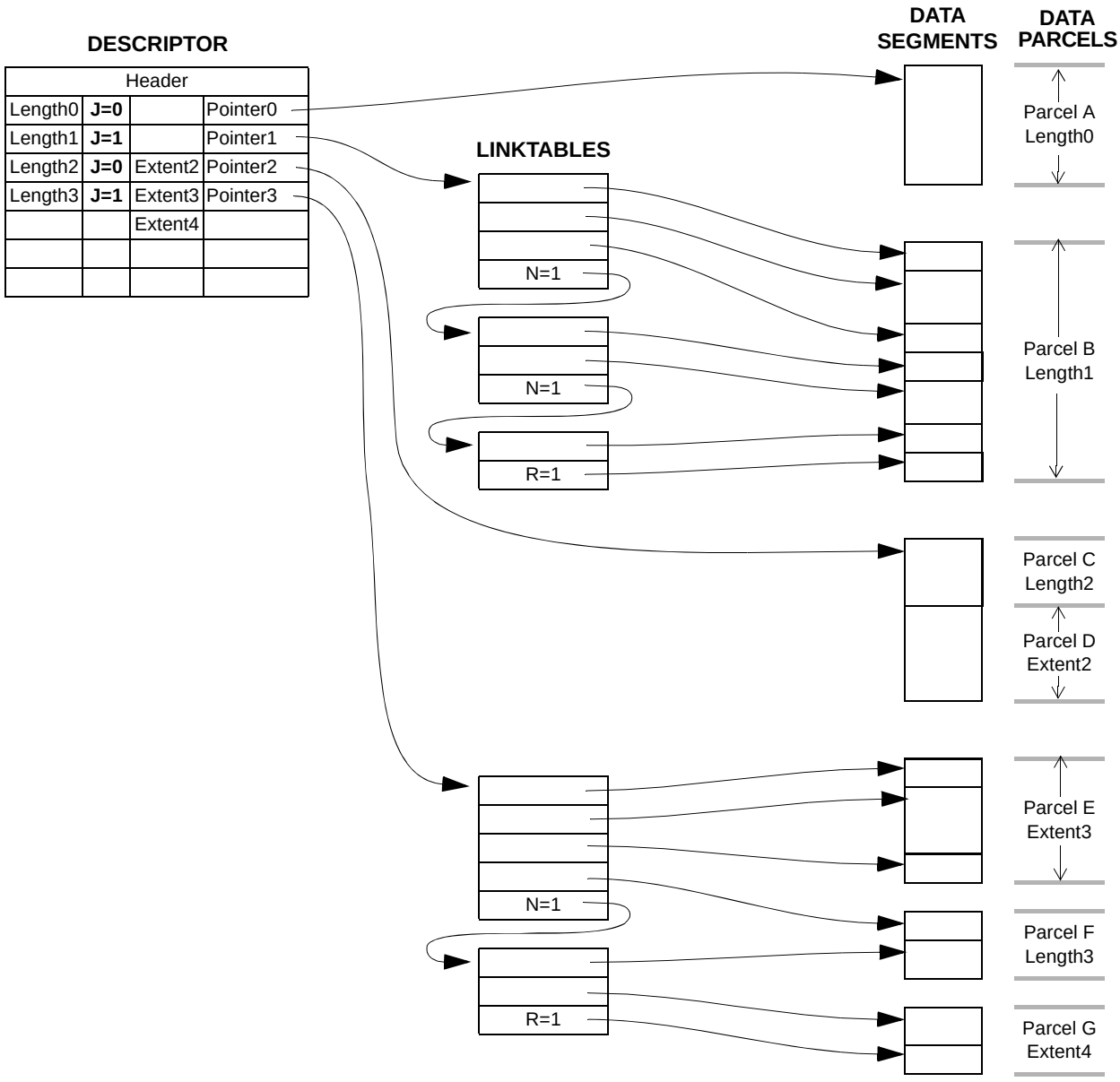


Figure 26-3. Descriptors, Link Tables, and Data Parcels

26.2.1.1 Descriptors

The core processor maintains a record of current secure sessions and the corresponding keys and contexts of those sessions. Once the core processor has determined that a security operation is required, it creates a descriptor containing all the information the SEC needs to perform the security operation. The core processor creates the descriptor in main memory, then writes a pointer to the descriptor into the Fetch FIFO of one of the SEC channels. The channel uses this pointer to read the descriptor into its descriptor buffer. Once it obtains the descriptor, the SEC uses its bus mastering capability to obtain inputs and write results, thus off-loading data movement and encryption operations from the core processor.

For test purposes, the core processor can also write keys, context, and text-data directly to execution units, using SEC core processor-controlled access. This method avoids using descriptors.

26.2.1.1.1 Descriptor Structure

SEC descriptors 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), as seen in **Figure 26-4**.

	63 48	47	46 40	39 36	35 32	31 0
Header	Header					Reserved
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

Figure 26-4. Descriptor Format

The header specifies the security operation to perform, the needed execution unit(s), 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 data 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, and the channel skips over the corresponding operations.

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 data 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.

26.2.1.1.2 Descriptor Header

Descriptors are created by the core processor to guide the SEC through required cryptographic operations. The header defines the operations to perform, the mode for each operation, and internal addressing used by the controller and channel for internal data movement.

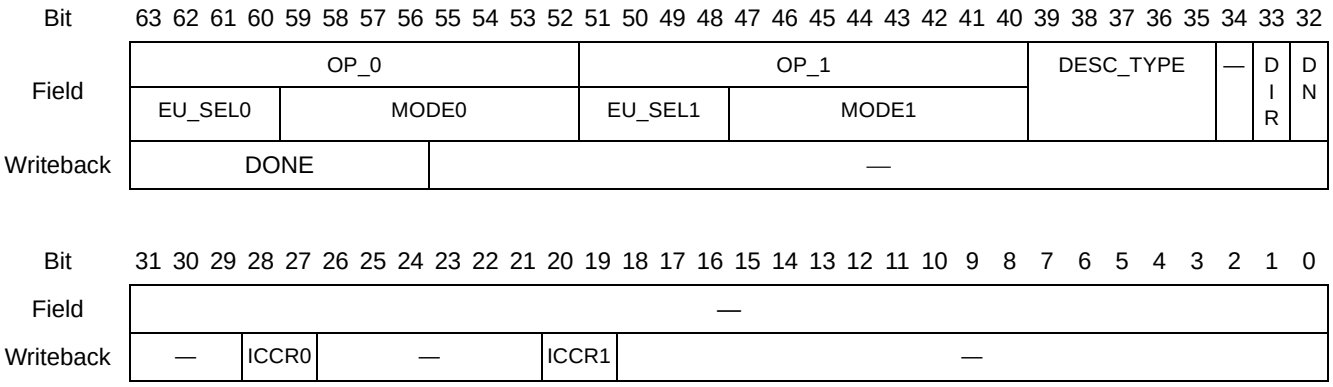


Figure 26-5. Header

Section 26.5.1.1, *Descriptor Structure* describes the header programming model.

Note: For more about descriptor types and the data used for each type, see **Table 26-4** on page 26-69.

26.2.1.1.3 Descriptor Pointers

The seven pointers define where the SEC reads its input and writes output data parcels in memory. The pointers are numbered from 0 to 6 as shown in **Figure 26-4**. 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 accesses the number of bytes defined 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 may 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.

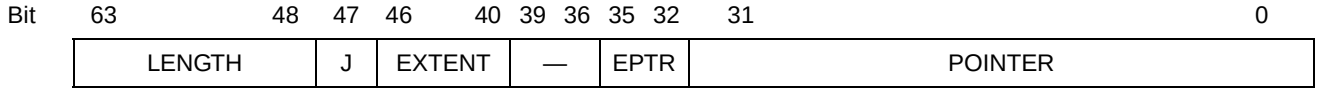


Figure 26-6. Pointer

Section 26.5.2, *Pointers* describes the pointer programming model.

26.2.1.2 Link Tables

Link tables implement scatter/gather capability. For gather operations, a link table specifies a list of memory segments to concatenate in the process of assembling data parcels. For scatter operations, a link table specifies a list of memory segments into which the output data is written. Scatter or gather of a data parcel may be specified by a single link table or by a chain of link tables linked together with pointers (see **Figure 26-3**).

The link table or chain of link tables accessed through a descriptor POINTER must specify enough memory segments to hold all the data accessed via that pointer. In most cases, only a single data parcel is accessed through a given POINTER, and the chain of link tables specifies only that parcel. In other cases, the descriptor POINTER is used multiple times to access a sequence of data parcels, and the chain of link tables must supply data for the entire sequence. If a link table is used to access a sequence of data parcels, the end of each parcel must also be at the end of a memory segment. In other words, a single memory segment must not straddle two data parcels. An example of proper construction of link tables is illustrated in **Figure 26-7**.

A link table may 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 may 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).

Note: See **Section 26.5.3**, *Link Tables*, on page 26-76 for the link table programming model.

26.2.1.3 Using Descriptors and Link Tables

Figure 26-7 demonstrates the use of link tables. Assume that the current descriptor type calls for the channel to read a data parcel using the Pointer3 and Extent3 fields, and assume that J3 = 1. Due to the J3 value, Pointer3 is used as the address of a link table. The channel begins by reading the first four 64-bit strings starting at Pointer3 into an internal gather table buffer. Using the first entry, the channel starts accessing the data parcel by reading SEGLEN bytes beginning at SEGPTR. If the required data 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. This process continues until there are no more bytes to read. If the channel gather table buffer is exhausted, the channel reads the next four 64-bit strings of the link table into its gather table buffer. If the controller encounters a gather table buffer entry where N = 1, the channel uses SEGPTR to find the next link table in the chain. The last byte of the parcel (Extent3) must coincide with the last byte of a memory segment, or unpredictable results may occur.

Now assume that the channel accesses its next data 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 data parcel is obtained by using Pointer3 yet again, this time with length given by Extent4.

Assume that for the current descriptor type, the Extent4 data parcel is the last one to be 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.

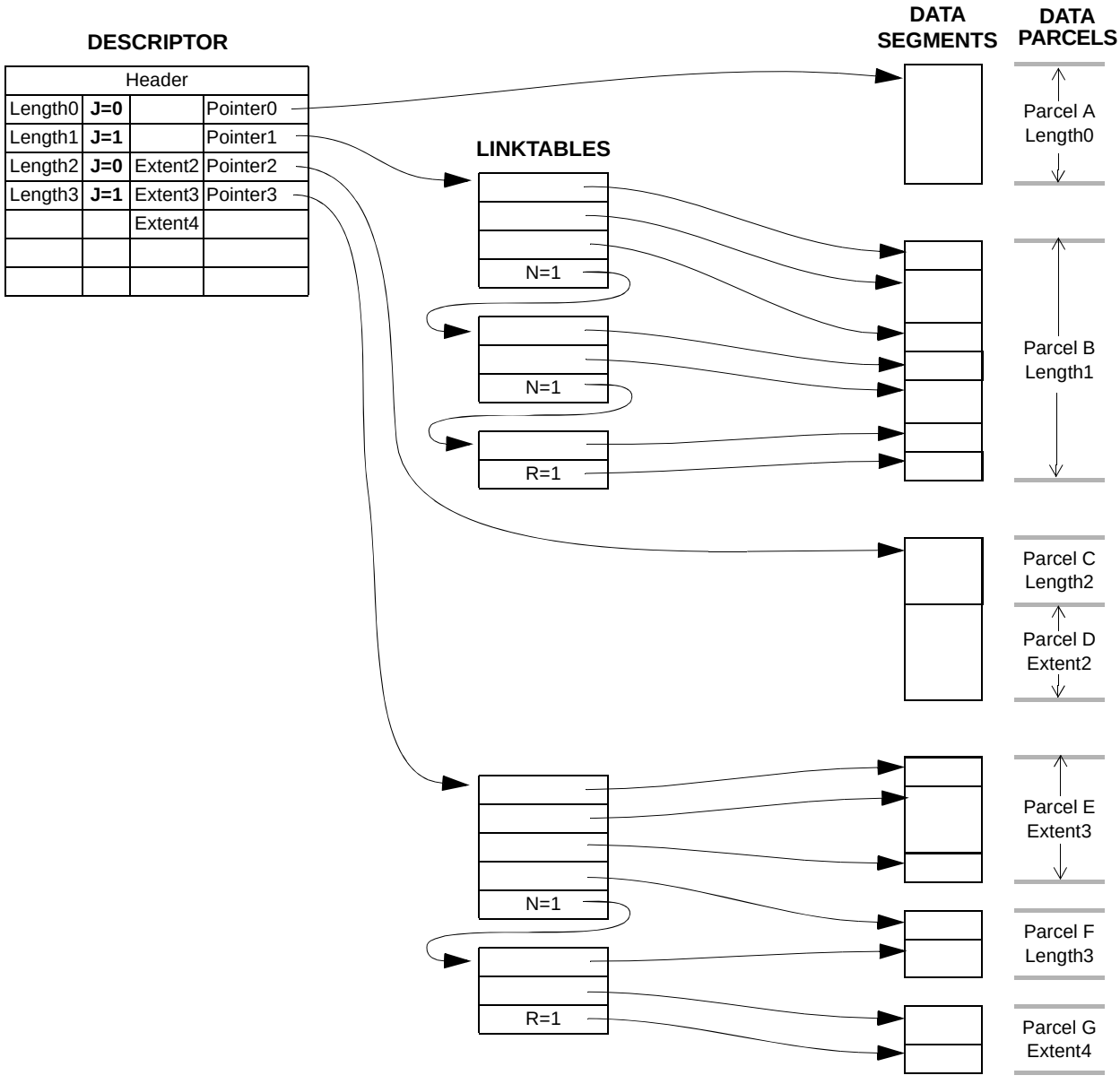


Figure 26-7. Descriptors, Link Tables, and Data Parcels

26.2.2 Assignment of EUs to Channels

Assignment of an EU to a channel is done dynamically, as follows:

- The channel requests an EU, the controller checks to see if the requested EU is available, and if it is, the controller grants the channel assignment of the EU.
- If an EU is available for a channel when requested, the controller asserts the grant signal pertaining to the request from the channel. The grant signal remains asserted until the channel is done and releases the EU.
- In some cases, a channel may request two EUs. The channel does this by requesting the primary EU and then requesting the secondary EU. Once the controller has granted both EUs, this channel can request that the secondary EU snoop the bus. Snooping status is indicated in the MI and MO bits of Channel Pointer Status Register (CPSR). See **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4])* for details.
- In all cases, the controller assigns the primary EU to a requesting channel as the EUs become available. The controller does not wait until both EUs are available before issuing any grants to a channel that requests two EUs.

Since there are multiple channels in the SEC, access to execution units through the channels must have arbitration. The controller implements a snapshot arbiter for each EU. When access requests occur, the arbiter records the set of pending requests (that is, it takes a snapshot) and then satisfies those requests as the resource becomes available. When all requests in the snapshot are satisfied, the arbiter takes another snapshot.

Within a given snapshot, the arbiters can use either a weighted priority-based scheme or a round-robin scheme, depending on the values of the Channel 3 and Channel 4 EU priority count fields in the Master Control Register. Each of these fields is 2-bits and can have a value from 0 to 3. If both fields have non-zero values, the arbiter implements the weighted priority scheme. If both fields are 0, the arbiter uses the round-robin scheme. Having only one field with a zero value results in unpredictable operation.

26.2.2.1 Weighted Priority Arbitration

The weighted priority scheme uses the following:

- Channel 1—Highest priority
- Channel 2—Second highest priority, the channel 3 or channel 4 priority has expired
- Channel 3—Third priority, unless the channel 4 priority has expired.
- Channel 4—Lowest priority, until the channel 4 priority expires.

Initially, the priority is channel 1, channel 2, channel 3, and channel 4, in that order. To prevent channels 3 and 4 from always being locked out, the priority scheme uses the configured values in the channel 3 and 4 priority fields in the Master Control Registers to determines how many times channel 3 or channel 4 can be refused access to an EU in favor of a higher priority channel.

The arbiter uses the channel EU priority count field for each of these channels to keep track of the arbitration losses. When the channel has lost the arbitration (that is, is not granted access) for the specified EU the number of times specified in its count field, then that channel becomes the second highest priority when the requested EU becomes available. Channel 1 always has the highest priority, but it cannot make back-to-back requests, so the second highest priority channel is serviced upon completion of the current channel 1 operation.

Note: Access to the system bus uses the same type of arbitration as the EU access. The arbitration methods do not have to be configured as the same. That is, you can use weighted priority for one and round-robin priority for the other.

26.2.2.2 Round-Robin Arbitration

In round-robin arbitration, requesting channels are granted access in rotating numerical order: 1, 2, 3, 4, 1, 2, 3, 4, and so on.

26.2.3 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 for operating SEC as a slave
- Master for transactions with SEC as the master

The internal bus is 64 bits wide, 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 processor-controlled access, the core processor uses the external 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 the internal location indicated by the address. For a read, the controller goes to the internal location, fetches the requested data from the specified address, 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. Multiple channels may request use of the controller at the same time, so the controller performs arbitration to choose a channel. The controller then services the request and performs the required transfer. Most transfers involve not only the internal bus, but also the system bus with the controller as bus master. Here are examples of the various types of transfers:

- Obtaining a descriptor:
  - channel makes request, arbitrates for attention from controller
  - controller arbitrates for use of the system bus and performs read from external memory

  - controller sends descriptor to channel over the internal bus
- Transferring a data length parameter from a channel to the EU
  - channel makes request, arbitrates for attention from controller
  - 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, arbitrates for attention from controller
  - 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, arbitrates for attention from controller
  - 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

26.2.3.1 Arbitration for Use of the Controller and Buses

The controller attempts to maximize utilization of the system bus by grouping outstanding bus requests from the channels by request type (read or write). The controller performs all write requests to the system bus, followed by all read requests, and then repeats.

Within a request type, the controller grants bus access via the same snapshot scheme used for granting EUs. If, for example, the controller is doing writes, it takes a snapshot of the current write requests, satisfies them as the bus becomes available, then takes another snapshot of write requests, and repeats. If there are no more write requests in a snapshot, the arbiter switches to handling reads. It repeatedly takes snapshots of the reads waiting and satisfies them until there are no more read requests in a snapshot. It then switches back to handling writes.

As with arbitration for EUs, controls for setting channel priorities for bus access are in the Master Control Register (Section 26.5.4.1, **Master Control Register (MCR)**), and the same two methods are available for selecting which request to satisfy within a snapshot. If both the channel 3 and channel 4 bus priority count fields are set to non-zero values, the arbiter implement the weighted priority scheme (see **Section 26.2.2.1**, *Weighted Priority Arbitration*). If both are zero, the arbitration will be round-robin (see **Section 26.2.2.2**, *Round-Robin Arbitration*). Setting only one of the fields to a non-zero value results in unpredictable operation. When the buses are granted to a channel, they are granted until the channel transfer is completely satisfied.

The SEC does not dynamically adjust its own transaction priorities. System software, however, can adjust SEC transaction priority in real time, with the change in priority taking effect immediately.

26.2.3.2 System Bus Master Read

When the controller is the bus master, a system bus read uses the following sequence:

1. Channel asserts its bus read request to the controller.
2. Channel furnishes external read address, internal write address, and transfer length.
3. Controller sends request acknowledge to channel.
4. Controller asserts request to the system bus through the master interface.
5. Controller waits for system bus read to begin.
6. 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 read does not begin on a 32-bit boundary, or
   - the previous write to an execution unit input FIFO does not end on a 32-bit boundary.
7. 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 is read.

When the SEC performs a transaction as master, the intended slave can terminate the transfer due to an error. The SEC transaction requests are posted to the device target queue, after which the device takes responsibility for either completing the transaction or signalling an error. An error in an SEC initiated transaction is also reported by the SEC via the Channel Interrupt Status Register (CISR_n). The core processor can determine which channel generated the interrupt by checking the ISR for the channel ERROR bit.

26.2.3.3 System Bus Master Write

When the controller is the system bus master, a system bus write uses the following sequence:

1. Channel asserts its bus write request to the controller.
2. Channel furnishes internal read address, external write address, and transfer length.
3. Controller sends request acknowledge to the channel.
4. 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 interface, and waits for the system bus to become available.
5. When the system bus becomes available, controller writes data from its FIFO to the master interface.

Even though the controller is handling a request from one channel, it can still acknowledge and queue requests from the other channels.

26.2.4 Controller Interrupts

All interrupt outputs from other blocks in the SEC feed to the controller as interrupt conditions. In addition, the controller can detect some interrupt independently. The controller maintains an Interrupt Status Register (ISR) with bits corresponding to all of possible interrupt conditions. If an interrupt condition occurs and it is enabled by corresponding bit of the Interrupt Enable Register (IER), then the associated ISR bit is set, indicating the presence of a pending interrupt. Whenever bits are set in the ISR, the controller asserts its interrupt output line to the core processor. In addition to the main interrupt (that includes all the interrupt conditions), there are five additional interrupts, one for each channel to indicate a DONE state and a general interrupt.

To handle an interrupt, the core processor must read the ISR to determine the interrupt source. It may need to perform further reads of Interrupt Status Registers in other blocks to obtain more detailed information about the cause. 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 ISR 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 another interrupt condition from the same source, then the Interrupt Status Register bit clears for a cycle and then sets again, keeping the interrupt output line to the core processor active. If the ISR bit is successfully cleared and no other interrupt conditions are present, the controller deasserts its interrupt output. If any interrupts are still pending in the ISR, the interrupt output remains asserted.

The EU interrupt conditions can 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 ISR. In addition, interrupt conditions from EUs can be individually blocked by bits of the controller IER before they reach the controller ISR. For normal operation, interrupt conditions from EUs are disabled in the controller IER, 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 IER.

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 queued by the controller. When the core processor clears a channel interrupt, if there are no other interrupts queued from that channel, then the channel Done interrupt is deasserted. If other interrupts remain in the queue, the controller deasserts the interrupt for one cycle and then reasserts it again. Interrupts are queued separately for each channel.

The SEC generates a single interrupt to the device embedded programmable interrupt controller (EPIC). See the *MSC8144 SC3400 DSP Core Subsystem Reference Manual* for additional information on the EPIC. The user allows SEC interrupts to be reported to the core processor by clearing the mask bit in the associated vector/priority register of the EPIC.

26.2.5 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 26.5.4.1**, *Master Control Register (MCR)*, on page 26-77.
- 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 26.5.4.2**, *Controller Identification Register (CIDR)*, on page 26-79.
- Controller IP Block Revision Register (CIPBRR). This register contains a 64-bit value that identifies the version of the SEC 2.1 protocol supported by this device. For details on this register, see **Section 26.5.4.3**, *Controller IP Block Revision Register (CIPBRR)*, on page 26-79.
- 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 26.5.4.4**, *EU Assignment Status (EUASR)*, on page 26-80.
- 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 26.5.4.5**, *Controller Interrupt Enable Register (CIER)*, on page 26-81.
- 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 26.5.4.6**, *Controller Interrupt Status Register (CISR)*, on page 26-84.
- 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 26.5.4.7**, *Controller Interrupt Clear Register (CICR)*, on page 26-87.

26.3 Channels

A channel in the SEC manages 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 26.2.1.1.1, **Descriptor Structure**) 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. 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 may involve using link tables to gather input data that has been 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 may 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 may 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.

26.3.1 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 26.5.5.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4])*, on page 26-89.
- Channel Pointer Status Register (CPSR). 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 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4])*, on page 26-92.
- Current Descriptor Pointer Register (CDPR). Contains the address for the currently processing descriptor. For details, see **Section 26.5.5.3**, *Current Descriptor Pointer Register (CDPR)*, on page 26-98.
- 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 26.5.5.4**, *Channel Fetch FIFO (CFF)*, on page 26-99.
- 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. This structure is read-only because the descriptor is always fetched from system memory. For details, see **Section 26.5.5.5**, *Channel Descriptor Buffer (DB)*, on page 26-100.

26.3.2 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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84 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 26.5.4.5**, *Controller Interrupt Enable Register (CIER),* on page 26-81 for details).

26.3.2.1 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 26.5.5.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4])*, on page 26-89). Assuming the CDIE (Channel Done Interrupt Enable) is set, the channel will generate 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.

Note: 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 26.2.4**, *Controller Interrupts*, on page 26-18).

26.3.2.2 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 Status Register (CPSR). Refer to **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4]),* on page 26-92 for a complete listing of error types.

26.3.2.3 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 has been 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.

26.4 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 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)
- One private internal Random Number Generator (RNG)

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 26.5**, *Programming Model*, on page 26-66). 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 RNG all include the following 64-bit registers:

- Mode Register
- Key Size 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 26.5**, *Programming Model*, on page 26-66 for a detailed description of each register and associated register fields.

26.4.1 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 26.5**, *Programming Model*, on page 26-66 provides a detailed description of each register and associated register fields.

26.4.1.1 PKEU Mode Register

This register pacifies 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.

26.4.1.2 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 256. Specifying a key size outside this range causes a key size error (KSE) in the PKEU Interrupt Status Register.

26.4.1.3 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 may 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.

An illegal data size error will be generated as follows:

- All non ECC routines with a data size > 256 generate an illegal data size error.
- All ECC routines with a data size > 64 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.

26.4.1.4 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 97 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.

26.4.1.5 PKEU Reset Control Register

This register contains three reset options specific to the PKEU.

26.4.1.6 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.

26.4.1.7 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 26.2.4**, *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 Status Register (see **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4])*, on page 26-92) 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.

26.4.1.8 PKEU Interrupt Mask Register

This register controls the result of detected errors. For a given error (as defined in Section 26.4.1.7, **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.

26.4.1.9 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.

26.4.1.10 PKEU Parameter Memories.

The PKEU uses four 2048-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 2048 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 512 bit memories, and is used to specify particular curve parameters and input values.
- PKEU Parameter Memory B. This 2048 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 512 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 2048 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 will cause address error to be reflected in the PKEU Interrupt Status Register.

- PKEU Parameter Memory N. This 2048 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.

26.4.1.11 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 26.5.6.1**, *PKEU Mode Register (PKEUMR)*, on page 26-101 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
- A, B, E, N. 2048-bit values from the parameter memories (see Section 26.4.1.10). Unless otherwise noted, N is modulus, E is exponent or key, and A and B are other operands.
- $\underline{X}$. 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, 2048-bit memories A and B are each segmented into 512-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 384-byte data structure constructed by the SPK_BUILD routine, which is the concatenation of six 512-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 following sections give a brief description of the function performed by each routine using the inputs, outputs, and requirements.

26.4.1.11.1 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 = 97
  - maximum Data Size = 2048

26.4.1.11.2 MOD_EXP: Prime field (F_p) Exponential mod N and Deconvert From Montgomery Format (0x02)

- Input:
  - N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
  - Data Size = the number of significant bits in N
  - $\underline{A}$ = field element in Montgomery format
  - E = Exponent
  - Key Size = the number of significant bytes in E
- Output: B = $\underline{A}^E$ mod N, a field element, non-Montgomery format
- Requirements:
  - minimum modulus size = 97 bits
  - maximum modulus size = 2048 bits
  - E > 1
  - maximum key (exponent) size 256 bytes
  - $\underline{A}$ < N

26.4.1.11.3 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 = 97 bits
  - maximum modulus size = 2048 bits

26.4.1.11.4 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 = 97 bits
  - maximum modulus size = 1023 bits
  - minimum key size = 13 bytes
  - maximum key size = 256 bytes
  - key size > modulus size/8

26.4.1.11.5 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 = $R^2 \bmod N$, 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 = 97 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 = 256 bytes

26.4.1.11.6 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 will be 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 = 97 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 = 256 bytes

26.4.1.11.7 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 will be 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 = 97 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 = 256 bytes

26.4.1.11.8 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 = 97 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 = 256 bytes

26.4.1.11.9 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 = 97 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.

26.4.1.11.10 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 will be 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 = 97 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.

26.4.1.11.11 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 will be 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 = 97 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.

26.4.1.11.12 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 will be 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 = 97 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.

26.4.1.11.13 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 = 97 bits
  - maximum modulus size = 512 bits

26.4.1.11.14 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 = 97 bits
  - maximum modulus size = 512 bits
  - A is a is an element of the polynomial field and therefore has fewer significant bits than the modulus N.

26.4.1.11.15 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} mod N, a field element
- Requirements:
  — minimum modulus size = 97 bits
  — maximum modulus size = 512 bits
  — A is a is an element of the prime field and therefore is less than the modulus N.

26.4.1.11.16 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 = 97 bits
  — maximum modulus size = 2048 bits
  — A and B are elements of the prime field and are therefore less than the modulus N

26.4.1.11.17 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 = 97 bits
  — maximum modulus size = 2048 bits
  — A and B are elements of the prime field and are therefore less than the modulus N

26.4.1.11.18 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 will be 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 = 97 bits
- maximum modulus size = 2048 bits
- Inputs A and B are elements of the prime field and are therefore less than the modulus N

26.4.1.11.19 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 will be 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 = 97 bits
  - maximum modulus size = 2048 bits
  - Inputs A and B are elements of the prime field and are therefore less than the modulus N

26.4.1.11.20 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 = 97 bits
  - maximum modulus size = 512 bits
  - A and B are elements of the polynomial field and therefore have fewer significant bits than the modulus N.

26.4.1.11.21 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 will be 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 = 97 bits
  - maximum modulus size = 512 bits
  - A and B are elements of the polynomial field and therefore have fewer significant bits than the modulus N.

26.4.1.11.22 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 will be 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 = 97 bits
  - maximum modulus size = 512 bits
  - A and B are elements of the polynomial field and therefore have fewer significant bits than the modulus N.

26.4.1.11.23 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 will be 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 = 97 bits
  - maximum modulus size = 2048 bits
  - E > 1
  - maximum key (exponent) size 256 bytes
  - A is an element of the field and is therefore less than the modulus N.

26.4.1.11.24 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 1536 bits (384 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 512 bits
- Requirements: maximum size of each parameter = 511 bits

26.4.2 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 26.5**, *Programming Model*, on page 26-66 provides a detailed description of each register and associated register fields.

26.4.2.1 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.

26.4.2.2 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.

26.4.2.3 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. 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.

26.4.2.4 DEU Reset Control Register

This allows three levels reset of just DEU, as defined by the three self-clearing bits:

26.4.2.5 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.

26.4.2.6 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 26.4.2.7**, *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 26.2.4**, *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 Status Register (see **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4])*, on page 26-92) 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 26.5.8.4**, *AESU Reset Control Register (AESURCR)*, on page 26-128).

26.4.2.7 DEU Interrupt Mask Register

The Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 26.4.2.6**, *DEU Interrupt Status Register*, on page 26-39), 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.

26.4.2.8 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.

26.4.2.9 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.

26.4.2.10 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.

26.4.2.11 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.

26.4.3 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 26.5.5.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4])*, on page 26-89), and mask the ICE bit in the Interrupt Mask Register (**Section 26.4.3.7**, *AESU Interrupt Mask Register*, on page 26-43). In this case the normal done signaling (by interrupt or writeback) is undisturbed.

To signal the ICV checking result by interrupt, unmask the ICE bit in the Interrupt Mask Register and turn off the IWSE and AWSE bits in the Channel Configuration Register. If there is no ICV mismatch, then the normal done signalling (by interrupt or writeback) occurs. When there is an ICV mismatch, there is an error interrupt generated to the core processor, but no done interrupt or writeback.

The following subsections include general descriptions of the AESU registers and structures. **Section 26.5**, *Programming Model*, on page 26-66 provides a detailed description of each register and associated register fields.

26.4.3.1 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).

The use of a restore decrypt key (RDK) is completely optional, because the input time of the preserved decrypt key may exceed the ~12 cycles required to restore the decrypt key for processing the first block.

The following procedure is recommended if you are using an RDK:

1. Set the AESU mode as Decrypt. The descriptor causes the SEC to write the contents of the Context Registers and the key registers (containing the expanded decrypt key) to memory.
2. Use the descriptor type 0100_0- AESU Key Expand Output for the decryption of the first portion of the message.
3. To process the remainder of the message, use a common descriptor type (0001_0), and set the restore decrypt key mode bit.
4. Load the Context Registers and the expanded decrypt key with previously saved key and context data from the first message. The key size is written as before (16, 24, or 32 bytes).

26.4.3.2 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.

26.4.3.3 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.

26.4.3.4 AESU Reset Control Register

This register allows three levels reset of just AESU, as defined by the three self-clearing bits:

26.4.3.5 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.

26.4.3.6 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 26.4.3.7**, *AESU Interrupt Mask Register*, on page 26-43).

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 26.2.4**, *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 Status Register (see **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4])*, on page 26-92) 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.

26.4.3.7 AESU Interrupt Mask Register

The AESU Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 26.5.8.6**, *AESU Interrupt Status Register (AESUISR)*, on page 26-130), 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.

26.4.3.8 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.

26.4.3.9 AESU Context Registers

There are seven 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.

The Context Registers must be read when changing context and restored to their original values to resume processing an interrupted message (CBC, CTR and CCM modes). For CTR and CCM modes, all seven 64-bit Context Registers must be read to retrieve context, and all seven must be written back to restore context. Effectively, the user must read the four empty place holder Context Registers in addition to the three Context Registers holding the Counter and Counter Modulus Exponent when in CTR mode. 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 with the lower bytes in the lowest 64-bit Context Register.

26.4.3.9.1 Context for CBC Mode

The Context Registers include two 64-bit context data registers for use in CBC mode that allow the core processor to read/write the contents of the initialization vector (IV):

- IV1 holds the least significant bytes (LSBs) of the initialization vector (bytes 1–8).
- IV2 holds the most significant bytes (MSBs) 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 CBC mode bit is not set, a context error is generated. The IV registers may only be read after processing has completed, as indicated by the assertion of a done interrupt (DI) in the AESU Status Register as shown in **Section 26.5.8.5**, *AESU Status Register (AESUSR)*, on page 26-129. If the IV registers are read prior to assertion of Interrupt Done, an early read error is generated. The IV registers must be read when changing context and restored to resume processing an interrupted message (CBC mode only).

26.4.3.9.2 Context for Counter Mode

In counter 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. The value of M is specified by writing to Context Register 3.

26.4.3.9.3 Context for SRT Mode

SRT 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. As with counter 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. The value of M is specified by writing to Context Register 3.

The only difference between SRT mode and CTR mode is that in SRT mode, the AES Context is loaded and read via Context Registers 1–3, with no requirement to access Context Registers 4–7. In CTR mode, Context Registers 1–4 must be loaded with zeros, with the counter and modulus being loaded into and read from Context Registers 5–7.

26.4.3.9.4 Context for CCM 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 26-8**.

		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 26-8. AESU CCM Context Registers

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 has been 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.

26.4.3.9.5 CCM Encryption Processing

With the session specific key and context, the AESU will perform 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 has been 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 has been 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 802.11i 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 has been converted to ciphertext. The SEC controller will manage FIFO reads and writes, fetching plaintext and writing ciphertext per the pointers provided in the descriptor. When all ciphertext and the MIC has been output, the CCM encrypt operation is complete.

26.4.3.9.6 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 has been 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 has been 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 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.

26.4.3.10 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 1. Any key data written to bytes beyond the value written to the Key Size Register will be 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 will be generated.

The Key Data Registers may be read when changing context in decrypt mode. To resume processing, the value read must be written back to the key registers and the restore decrypt key bit must be set in the Mode Register. This eliminates the overhead of expanding the key prior to starting decryption when switching context.

26.4.3.11 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 may 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 may be read any time the OFR signal is asserted (as indicated in the AESU Status Register). This will indicate 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. Note that AESU only supports 2 byte CCM headers and does not support extended CCM headers.

26.4.4 Message Digest Execution Unit (MDEU)

In 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.

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 26.5.5.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4])*, on page 26-89), and mask the ICE bit in the Interrupt Mask Register (**Section 26.4.4.7**, *MDEU Interrupt Mask Register*, on page 26-52). 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) will occur. When there is an ICV mismatch, there will be an error interrupt to the core processor, but no done interrupt or writeback.

The following subsections include general descriptions of the MDEU registers and structures. **Section 26.5**, *Programming Model*, on page 26-66 provides a detailed description of each register and associated register fields.

26.4.4.1 MDEU Mode Register

The MDEU Mode Register is used to program the function of the MDEU. Bits 7–0 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 (see **Section 26.4.4.1**, *MDEU Mode Register*, on page 26-49). 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.

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 26-2**

Table 26-2. 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 26-3**.

Table 26-3. 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 26.2.1.1**, *Descriptors*, on page 26-9.

26.4.4.2 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 most 64 bytes of key. MDEU generates a key size error if the value written to this register exceeds 64 bytes.

26.4.4.3 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 occur during inbound processing, when it is necessary to hold back data from the MDEU until the pad length has been decrypted.

Since the MDEU does not support bit offsets, the 3 least significant bits (lsbs) must be written as 0 and are always read as zero. Furthermore, when the CONT bit of the MDEU Mode Register is set (1), the data size must be a multiple of the 512-bit block size (that is, the 9 lsbs must be

written as 0). Violating either of these conditions causes a data size error (MDEU Interrupt Status Register DSE bit = 1).

This register is cleared when the MDEU is reset or reinitialized. At the end of processing, its contents has been decremented down to zero (unless there is an error interrupt).

Note: Writing to the Data Size Register will allow the MDEU to enter auto-start mode. Therefore, the required Context Registers must be written prior to writing the data size.

26.4.4.4 MDEU Reset Control Register

This register allows three levels of reset just for the MDEU, as defined by the three self-clearing bits.

26.4.4.5 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.

26.4.4.6 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 26.5.9.7**, *MDEU Interrupt Mask Register (MDEUIMR)*, on page 26-144).

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 26.5.4.6**, *Controller Interrupt Status Register (CISR)*, on page 26-84). 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 Status Register (see **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4])*, on page 26-92) 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.

26.4.4.7 MDEU Interrupt Mask Register

The MDEU Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 26.5.9.6**, *MDEU Interrupt Status Register (MDEUISR)*, on page 26-142), 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.

26.4.4.8 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 26.5.9.1**, *MDEU Mode Register (MDEUMR)*, on page 26-137). This register is cleared when the MDEU is reset or reinitialized.

26.4.4.9 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 (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 initiate a done interrupt.

26.4.4.10 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: SHA-1and SHA-256 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, only if the appropriate Context Register values are used in hash generation per the mode selected. Even thought 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 26.4.4.4**, *MDEU Reset Control Register*, on page 26-51) or SEC global software reset (**Section 26.5.4.1**, *Master Control Register (MCR)*, on page 26-77) does not clear these registers.

26.4.4.11 MDEU Key Registers

The MDEU maintains eight 64-bit registers for writing an HMAC key. The IPAD and OPAD operations are performed automatically on the key data when required.

Note: SHA-1 and SHA-256 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.

26.4.4.12 MDEU FIFOs

MDEU uses an input FIFO to hold data to be hashed. 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 by 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 last bytes remaining 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.

26.4.5 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 26.5**, *Programming Model*, on page 26-66 provides a detailed description of each register and associated register fields.

26.4.5.1 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.

26.4.5.1.1 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 will not require a key. Rather, the core processor will write the context to the AFEU and message processing will occur using the provided context. This mode is used to resume processing of a message using the already permuted S-Box. The context may 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 26.4.5.1.2, **Dump Context**).

26.4.5.1.2 Dump Context

This mode may 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 will make 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 will re-use 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.

26.4.5.2 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.

26.4.5.3 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 has been loaded. The user then writes the message data size to the context/Data Size Register. After this write, the user may 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.

26.4.5.4 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.

26.4.5.5 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.

26.4.5.6 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 26.5.10.7**, *AFEU Interrupt Mask Register (AFEUIMR)*, on page 26-157).

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 26.5.4.6**, *Controller Interrupt Status Register (CISR)*, on page 26-84). 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 Status Register (see **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4])*, on page 26-92) 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.

26.4.5.7 AFEU Interrupt Mask Register

The Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 26.5.10.6**, *AFEU Interrupt Status Register (AFEUISR)*, on page 26-155), 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.

26.4.5.8 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 will 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.

26.4.5.9 AFEU Context

This section provides additional information about the AFEU context memory and its related pointer register.

26.4.5.9.1 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 26.4.5.1**, *AFEU Mode Register*, on page 26-54). After context data, the context length must be written to the context/data length register (see **Section 26.5.10.3**, *AFEU Context/Data Size Register (AFEUCDSR)*, on page 26-152). 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 will be generated.

Valid context data can only be read after Reading context data before the module is done will generate 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 26.4.5.1, **AFEU Mode Register**). See Section 26.4.5.10.1, **AFEU FIFOs** for more about addressing the FIFOs.

26.4.5.9.2 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 will be 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.

26.4.5.10 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 will generate an address error interrupt.

26.4.5.10.1 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 address 3_8E00. 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 have been 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.

26.4.6 Kasumi Execution Unit (KEU)

The Kasumi execution unit (KEU) has been 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 26.5.5.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4])*, on page 26-89), and mask the ICE bit in the Interrupt Mask Register (**Section 26.5.11.7**, *KEU Interrupt Mask Register (KEUIMR)*, on page 26-169). 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 26.5**, *Programming Model*, on page 26-66 provides a detailed description of each register and associated register fields.

26.4.6.1 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.

26.4.6.2 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.

26.4.6.3 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 26-1. 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 26.5.11.1**, *KEU Mode Register (KEUMR)*, on page 26-161).

Example 26-2. 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.

26.4.6.4 KEU Reset Control Register

The KEU Reset Control Register allows 3 levels reset of the KEU, as defined by the 3 self-clearing bits:

26.4.6.5 KEU Status Register

The KEU Status Register is a read-only register that reflects the state of six status outputs. Writing to this location will result in an address error being reflected in the KEU Interrupt Status Register.

26.4.6.6 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 26.5.11.7**, *KEU Interrupt Mask Register (KEUIMR)*, on page 26-169).

If the AESU Interrupt Status Register is non-zero, the KEU halts and the KEU error interrupt signal is asserted to the controller (see **Section 26.5.4.6**, *Controller Interrupt Status Register (CISR)*, on page 26-84). 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 Status Register (see **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4])*, on page 26-92) 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.

26.4.6.7 KEU Interrupt Mask Register

The KEU Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 26.5.11.6**, *KEU Interrupt Status Register (KEUISR)*, on page 26-167), 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.

26.4.6.8 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 may 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.

26.4.6.9 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.

26.4.6.10 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.

26.4.6.11 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.

26.4.6.12 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.

26.4.6.13 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 will be 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.

26.4.6.14 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.

26.4.6.15 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 will result in an address error being reflected in the KEU Interrupt Status Register.

26.4.6.16 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 have been 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.

26.4.7 Random Number Generator (RNG)

The RNG is an execution unit capable of generating 64-bit random numbers. It is designed to comply with the FIPS-140 standard for randomness and non-determinism. A linear feedback shift register (LSFR) and cellular automata shift register (CASR) are operated in parallel to generate pseudo-random data.

The RNG consists of six major functional blocks:

- Bus interface unit (BIU)
- Linear feedback shift register (LFSR)

- Cellular automata shift register (CASR)
- Clock controller
- Six ring oscillators

The states of the LFSR and CASR are advanced at unknown frequencies determined by the two ring oscillator clocks and the clock control. When a read is performed, the oscillator clocks are halted and a collection of bits from the LFSR and CASR are XORed together to obtain the 64-bit random output.

In typical operation, the RNG is used through channel-controlled access, which means that most reads and writes of RNG 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 RNG registers and structures. **Section 26.5**, *Programming Model*, on page 26-66 provides a detailed description of each register and associated register fields.

26.4.7.1 RNG Mode Register

The RNG 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.

26.4.7.2 RNG Data Size Register

The RNG Data Size Register is used to tell the RNG to begin generating random data. The actual contents of the Data Size Register do not affect the operation of the RNG. After a reset and prior to the first write of data size, the RNG builds entropy without pushing data onto the FIFO. Once the Data Size Register is written, the RNG will begin pushing data onto the FIFO. Data is pushed onto the FIFO every 256 cycles until the FIFO is full. The RNG then attempts to keep the FIFO full.

26.4.7.3 RNG Reset Control Register

This register contains three reset options specific to the RNG.

26.4.7.4 RNG Status Register

This RNG Status Register contains six fields that reflect the state of the RNG internal signals. The RNG Status Register is read-only. Writing to this location results in an address error being reflected in the RNG Interrupt Status Register.

26.4.7.5 RNG 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 RNG Interrupt Mask Register is zero (see **Section 26.5.12.5**, *RNG Interrupt Status Register (RNGISR)*, on page 26-183).

If the RNG Interrupt Status Register is non-zero, the RNG halts and the RNG error interrupt signal is asserted to the controller (see **Section 26.5.4.6**, *Controller Interrupt Status Register (CISR)*, on page 26-84). In addition, if the RNG 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 Status Register (see **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4])*, on page 26-92) 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 RNG Reset Control Register.

26.4.7.6 RNG Interrupt Mask Register

The RNG Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 26.5.12.5**, *RNG Interrupt Status Register (RNGISR)*, on page 26-183), 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.

26.4.7.7 RNG End_of_Message Register

The RNG End_of_Message Register is a writable location but serves no function in the RNG. It is documented for the sake of consistency with the other EUs.

26.4.7.8 RNG FIFO

The RNG 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 RNG FIFO address space dequeues data from the RNG 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 RNG output FIFO are reflected in the RNG Interrupt Status Register. Also, a write to the RNG output FIFO space is reflected as an addressing error in the RNG Interrupt Status Register.

26.5 Programming Model

The SEC uses the following structures and registers to configure and operate the security engine and the execution units:

- Descriptors and Link Tables
  - Descriptor structure, **page 26-69**
  - Descriptor header, **page 26-72**
  - Descriptor Pointers, **page 26-74**
  - Link tables, **page 26-76**
- SEC Controller
  - Master Control Register (MCR), **page 26-77**
  - Controller Identification Register (CIDR), **page 26-79**
  - Controller IP Block Revision Register (CIPBRR), **page 26-79**
  - EU Assignment Status Register (EUASR), **page 26-80**
  - Controller Interrupt Enable Register (CIER), **page 26-81**
  - Controller Interrupt Status Register (CISR), **page 26-84**
  - Controller Interrupt Clear Register (CICR), **page 26-87**
- Channels
  - Channel Configuration Registers 1–4 (CCR[1–4], **page 26-89**
  - Channel Pointer Status Registers 1–4 (CPSR[1–4]), **page 26-92**
  - Channel Current Descriptor Pointer Registers 1–4 (CDPR[1–4]), **page 26-98**
  - Channel Fetch FIFOs 1–4 (CFF[1–4], **page 26-99**
  - Channel Descriptor Buffers, **page 26-100**
- PKEU
  - PKEU Mode Register (PKEUMR), **page 26-101**
  - PKEU Key Size Register (PKEUKSR), **page 26-103**
  - PKEU AB Size Register (PKEUABSR), **page 26-104**
  - PKEU Data Size Register (PKEUDSR), **page 26-105**
  - PKEU Reset Control Register (PKEURCR), **page 26-106**
  - PKEU Status Register (PKEUSR), **page 26-107**
  - PKEU Interrupt Status Register (PKEUISR), **page 26-108**
  - PKEU Interrupt Mask Register (PKEUIMR), **page 26-110**
  - PKEU End_of_Message Register (PKEUEOMR), **page 26-111**
  - PKEU Parameter Memories, **page 26-111**
- DEU
  - DEU Mode Register (DEUMR), **page 26-113**
  - DEU Key Size Register (DEUKSR), **page 26-114**
  - DEU Data Size Register (DEUDSR), **page 26-115**
  - DEU Reset Control Register (DEURCR), **page 26-116**
  - DEU Status Register (DEUSR), **page 26-117**

- DEU Interrupt Status Register (DEUISR), **page 26-118**
- DEU Interrupt Mask Register (DEUIMR), **page 26-120**
- DEU End_of_Message Register (DEUEOMR), **page 26-122**
- DEU IV Register (DEUIVR), **page 26-122**
- DEU Key Registers (DEUKR[1–3], **page 26-123**
- DEU Input FIFO / Output FIFO, **page 26-123**

- AESU
  - AESU Mode Register (AESUMR), **page 26-124**
  - AESU Key Size Register (AESUKSR), **page 26-126**
  - AESU Data Size Register (AESUDSR), **page 26-127**
  - AESU Reset Control Register (AESURCR), **page 26-128**
  - AESU Status Register (AESUSR), **page 26-129**
  - AESU Interrupt Status Register (AESUISR), **page 26-130**
  - AESU Interrupt Mask Register (AESUIMR), **page 26-132**
  - AESU End_of_Message Register (AESUEOMR), **page 26-134**
  - AESU Context Registers (AESUCR[1–7]), **page 26-134**
  - AESU Key Registers (AESUKR[1–3]), **page 26-135**
  - AESU Input FIFO/Output FIFO, **page 26-136**
- MDEU
  - MDEU Mode Register (MDEUMR), **page 26-137**
  - MDEU Key Size Register (MDEUKSR), **page 26-139**
  - MDEU Data Size Register (MDEUDSR), **page 26-139**
  - MDEU Reset Control Register (MDEURCR), **page 26-140**
  - MDEU Status Register (MDEUSR), **page 26-141**
  - MDEU Interrupt Status Register (MDEUISR), **page 26-142**
  - MDEU Interrupt Mask Register (MDEUIMR), **page 26-144**
  - MDEU ICV Size Register (MDEUICVSR), **page 26-146**
  - MDEU End_of_Message Register (MDEUEOMR), **page 26-147**
  - MDEU Context Registers (MDEUCR), **page 26-147**
  - MDEU Key Registers (MDEUKR[1–8]), **page 26-149**
  - MDEU Input FIFO, **page 26-149**
- AFEU
  - AFEU Mode Register (AFEUMR), **page 26-150**
  - AFEU Key Size Register (AFEUKSR), **page 26-151**
  - AFEU Content/Data Size Register (AFEUCDSR), **page 26-152**
  - AFEU Reset Control Register (AFEURCR), **page 26-153**
  - AFEU Status Register (AFEUSR), **page 26-154**
  - AFEU Interrupt Status Register (AFEUISR), **page 26-155**
  - AFEU Interrupt Mask Register (AFEUIMR), **page 26-157**
  - AFEU End_of_Message Register (AFEUEOMR), **page 26-159**

— AFEU Context Memory, **page 26-159**
— AFEU Context Memory Pointer Register (AFEUCMPR), **page 26-160**
— AFEU Key Registers (AFEUKR[1–2]), **page 26-160**
— AFEU Input FIFO / Output FIFO, **page 26-160**

- KEU
  - KEU Mode Register (KEUMR), **page 26-161**
  - KEU Key Size Register (KEUKSR), **page 26-163**
  - KEU Data Size Register (KEUDSR), **page 26-164**
  - KEU Reset Control Register (KEURCR), **page 26-165**
  - KEU Status Register (KEUSR), **page 26-166**
  - KEU Interrupt Status Register (KEUISR), **page 26-167**
  - KEU Interrupt Mask Register (KEUIMR), **page 26-169**
  - KEU Data Out Register (KEUDOR), **page 26-171**
  - KEU End_of_Message Register (KEUEOMR), **page 26-172**
  - KEU IV1 Register (KEUIV1R), **page 26-173**
  - KEU ICV_In Register (KEUICVIR), **page 26-174**
  - KEU IV2 Register (KEUIV2R), **page 26-175**
  - KEU Context Registers 1-6 (KEUCR[1–6]), **page 26-176**
  - KEU Key Data Registers 1–2 (KEUKDR[1–2]), **page 26-177**
  - KEU Key Data Registers 3–4 (KEUKDR[3–4], **page 26-178**
  - KEU Input FIFO/Output FIFO, **page 26-178**
- Random Number Generator (RNG)
  - RNG Mode Register (RNGMR), **page 26-179**
  - RNG Data Size Register (RNGDSR), **page 26-180**
  - RNG Reset Control Register (RNGRCR), **page 26-181**
  - RNG Status Register (RNGSR), **page 26-182**
  - RNG Interrupt Status Register (RNGISR), **page 26-183**
  - RNG Interrupt Mask Register (RNGIMR), **page 26-184**
  - RNG End_of_Message Register (RNGEOMR), **page 26-186**
  - RNG Output FIFO, **page 26-186**

Note: Offsets used with the memory structures and registers in the SEC are from the CCSR base address 0xFFF10000.

26.5.1 Descriptors and Link Tables

The following sections described the overall descriptor structure, the descriptor header, descriptor pointers, and the link table format.

26.5.1.1 Descriptor Structure

Bit	63 48	47	46 40	39 36	35 32	31 0
Header	Header					Reserved
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 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). **Table 26-4** lists the descriptor types. **Table 26-5** summarizes the various descriptor formats.

Table 26-4. Descriptor Types

Value[1] (binary)	Descriptor Type	Notes
0000_0[2]	aesu_ctr_nonsnoop	AESU CTR non snooping
0001_0[2]	common_nonsnoop	Common, non snooping, non-PKEU, non-AFEU
0010_0[3]	hmac_snoop_no_afeu	Snooping, HMAC, non-AFEU
0011_0	—	Reserved
0100_0	—	Reserved
0101_0	common_nonsnoop_afeu	Common, non snooping, AFEU
0110_ 0	—	Reserved
0111_0	—	Reserved
1000_0	pkeu_mm	PKEU-Montgomery Multiplication
1001_0	—	Reserved
1010_0	—	Reserved
1011_0	—	Reserved
1100_0[3]	hmac_snoop_aesu_ctr	AESU CTR hmac snooping
1101_0	—	Reserved
1110_0	—	Reserved
1111_0	—	Reserved
0000_1	ipsec_esp	IPsec ESP mode encryption and hashing
0001_1	802.11i AES ccmp	CCMP encryption and hashing, suitable for 802.11i
0010_1	srtp	SRTP encryption and hashing
0011_1	pkeu_build	pkeu_build Elliptic Curve Cryptography

Table 26-4. Descriptor Types (Continued)

Value[1] (binary)	Descriptor Type	Notes
0100_1	pkeu_ptmul	pkeu_ptmul Elliptic Curve Cryptography
0101_1	pkeu_ptadd_dbl	pkeu_ptadd_dbl Elliptic Curve Cryptography
0110_1	—	Reserved
0111_1	—	Reserved
1000_1	tls_ssl_block	TLS/SSL generic block cipher (new for SEC2.1)
1001_1	tls_ssl_stream	TLS/SSL generic stream cipher (new for SEC2.1)
1010_1	raid_xor	XOR 3 sources together (new for SEC2.1)
others	—	Reserved

Notes:
1. Descriptor types from the SEC1.0, which have 0 in the last bit, are listed first, followed by the newer SEC 2.x types, which have 1 in the last bit.
2. Type 0000_0 is for AES-CTR operations. Type 0001_0 also supports AES-CTR, however to use AES-CTR with 0001_0, the user must prepend zeros to the AES-Ctx before loading the AES Context Registers.
3. Type 1100_0 is for AES-CTR operations with HMAC. Type 0010_0 also supports AES-CTR with HMAC, however to use AES-CTR with 0010_0, the user must prepend zeros to the AES-Ctx before loading the AES Context Registers.

Table 26-5. Descriptor Format Summary

Descriptor Type	field type	Pointer0	Pointer 1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
0000_0 aesu_ctr_ nosnoop	Length	nil	Cipher IV	Cipher Key	In FIFO	Out FIFO	Cipher IV Out	nil
	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
0001_0 common_ nosnoop for DES, KEU F8, RNG, AES-CCM	Length	nil	Cipher IV (incl. ICV In)	Cipher Key	In FIFO	Out FIFO	Cipher IV Out (incl. ICV Out)	nil
	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
0001_0 common_ nosnoop for MDEU	Length	nil	Hash IV	Hash Key	In FIFO	ICV In	Hash IV Out	ICV Out
	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
0001_0 common_ nosnoop for KEU F9	Length	nil	Cipher IV, (incl. ICV In)	Cipher Key	In FIFO	nil	Cipher IV Out	ICV Out
	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
0010_0 hmac_snoop _no_afeu	Length	HMAC Key	HMAC Data	Cipher Key	Cipher IV	In FIFO	Out FIFO	HMAC Out
	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
0101_0 common_ nosnoop_ afeu	Length	nil	ARC-4 Context (via In FIFO)	ARC-4 Key	In FIFO	Out FIFO	ARC-4 Context (via Out FIFO)	nil
	Extent	undefined	undefined	undefined	nil	nil	nil	undefined

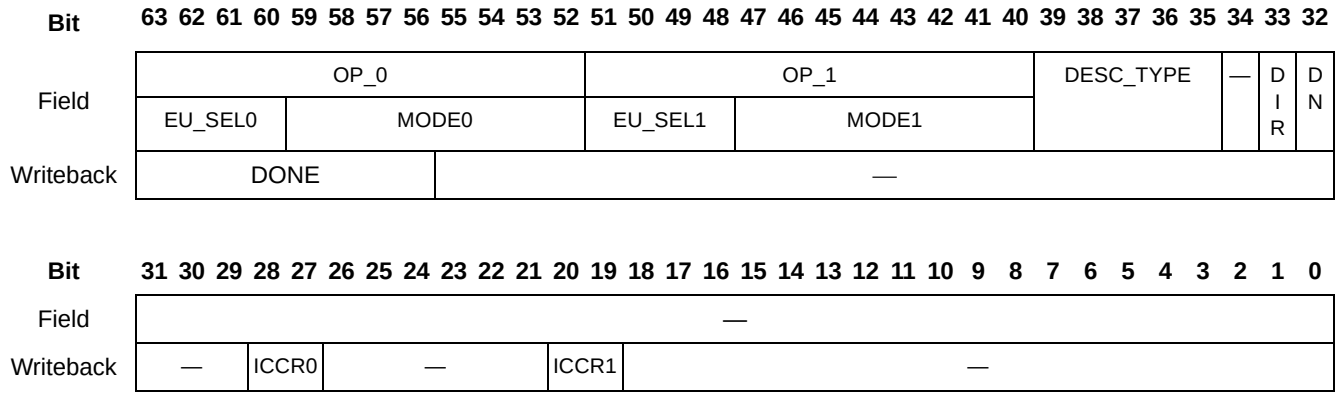
Table 26-5. Descriptor Format Summary (Continued)

Descriptor Type	field type	Pointer0	Pointer 1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
1000_0	Length	N	B	A	E	B Out	nil	nil
pkeu_mm	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
1100_0 hmac_snoop_ aesu_ctr	Length	HMAC Key	HMAC Data	AES Key	AES Ctx	In FIFO	Out FIFO	HMAC Out
	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
0000_1	Length	HMAC Key	HMAC Data	Cipher IV	Cipher Key	In FIFO	Out FIFO	Cipher IV Out
ipsec_esp	Extent	undefined	undefined	undefined	nil	HMAC In	HMAC Out	undefined
0001_1	Length	nil	AES-Ctx	AES Key	In FIFO	In FIFO	Out FIFO	AES-Ctx-Out
802.11i AES ccmp	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
0010_1	Length	HMAC Key	AES-Ctx	AES Key	In FIFO	Out FIFO	HMAC Out	AES-Ctx-Out
srtp	Extent	undefined	undefined	undefined	In FIFO	In FIFO	nil	undefined
0011_1	Length	A0	A1	A2	A3	B0	B1	'Build'
pkeu_build	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
0100_1	Length	N	E	'Build'	B1 Out	B2 Out	B3 Out	nil
pkeu_ptmul	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
0101_1	Length	N	'Build'	B2	B3	B1 Out	B2 Out	B3 Out
pkeu_ptadd_dbl	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
1000_1 outbound	Length	MAC Key	Cipher IV	Cipher Key	In FIFO Auth & CIpher	In FIFO Cipher Only	Out FIFO	Cipher IV Out
tls_ssl_ block	Extent	undefined	undefined	undefined	In FIFO Auth only	MAC Out	nil	undefined
1000_1 inbound	Length	MAC Key	Cipher IV	Cipher Key	nil	In FIFO Auth & CIpher	Out FIFO	Cipher IV Out
tls_ssl_ block	Extent	undefined	undefined	undefined	In FIFO Auth only	MAC In	MAC Out	undefined
1001_1 outbound	Length	MAC Key	Cipher IV	Cipher Key	In FIFO Auth & CIpher	In FIFO Cipher Only	Out FIFO	Cipher IV Out
tls_ssl_ stream	Extent	undefined	undefined	undefined	In FIFO Auth only	MAC Out	nil	undefined

Table 26-5. Descriptor Format Summary (Continued)

Descriptor Type	field type	Pointer0	Pointer 1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
1001_1 inbound	Length	MAC Key	Cipher IV	Cipher Key	nil	In FIFO Auth & CIpher	Out FIFO	Cipher IV Out
tls_ssl_ stream	Extent	undefined	undefined	undefined	In FIFO Auth only	MAC In	MAC Out	undefined
1010_1	Length	nil	nil	nil	In 1	In 2	In 3	Out
raid_xor	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
others		Reserved						

26.5.1.2 Descriptor Header



The header specifies the security operation to perform, the execution unit(s) needed, and the modes for each execution unit. **Table 26-6** 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 26-8**.

Table 26-6. 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 MDEU, then EU_SEL0 must be DEU, AESU, AFEU, or KEU. All other values of EU_SEL0 result in an “Unrecognized header” error condition.	See **Table 26-7** 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” or MDEU. Any other choice results in an “Unrecognized Header” error condition. **2.** If EU_SEL1 is MDEU, then EU_SEL0 must be DEU, AESU, AFEU, or KEU. All other values of EU_SEL0 result in an “Unrecognized header” error condition.	See **Table 26-7** 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.	
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 26-4** for valid settings. **Table 26-5** 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 26-7**). The done notification can take the form of an interrupt, a writeback in the DONE field of this header (see **Table 26-4**), 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 26-7. 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
0100	RNG (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)
1000–1110	Reserved
1111	Reserved for header writeback

Table 26-8. 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 26.5.5.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4]),* on page 26-89).	
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.
18–0	—	Reserved.	

26.5.2 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 **Figure 26-4**. The channel determines how it will use 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 may 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 may 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 26-9. 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 26-1**).	
31–0	POINTER	**Pointer** A memory address.	

Examples of special cases include:

- Sometimes, a descriptor field may 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. 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 26.2.1.2**, *Link Tables*, on page 26-12. Scatter/gather capability is available for all pointers of all descriptor types, with the exception that the raid-or descriptor type does not allow scatter/gather.

26.5.3 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																															

A link table may 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 may contain any number of link tables.

Table 26-10. 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 26.5.5.2, **Channel Pointer Status Registers (CPSR[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 Status Register (see Section 26.5.5.2).	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.

Table 26-10. Link Table Field Definitions (Continued)

Fields	Description	Settings
— 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 26-1**).	
SEGPTR 31–0	**Segment Pointer** A memory address.	

26.5.4 SEC Controller

The following sections describe the registers and structures used by the SEC Controller.

26.5.4.1 Master Control Register (MCR)

MCR Master Control Register Offset 0xC1030

Bit	63–42	41–40	39–33	32
	—	PRIORITY	—	SWR
Type	R/W			
Reset	0x00000000			

Bit	31–24	23–16	15–8	7–0
Field	CHN3_EU_PR_CNT	CHN4_EU_PR_CNT	CHN3_BUS_PR_CNT	CHN4_BUS_PR_CNT
R/W	R/W			
Reset	0x00000000			

The MCR controls certain functions in the controller and provides a means for software to reset the SEC. **Table 26-11** describes the Master Control Register bit fields.

Table 26-11. Master Control Register Bit Field Descriptions

Bits	Reset	Description	Settings
— 63–42	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 MPC8548 internal arbiter. The SEC does not dynamically alter its priority level based on system congestion or SEC utilization, however software may change the SEC priority level in realtime.	00 Lowest Priority (default) 01 Next Lowest Priority 10 Next Highest Priority 11 Highest Priority
— 39–33	0	Reserved. Write to zero for future compatibility.	

Table 26-11. Master Control Register Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
SWR 32	0	**Software Reset** Writing 1 to this bit will cause a global software reset. Upon completion of the reset, this bit will be 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 has been denied access to a requested EU long enough to warrant immediate elevation to level 2 priority. **Note:** If set to zero, the CHN4_EU_PR_CTR must also be set to zero, and the controller assigns EUs on a purely round-robin basis. If set to non-zero, CHN4_EU_PR_CTR must be set to a different, non-zero value.	
CHN4_EU_PR_CNT 23–17	0	**Channel 4 EU Priority Counter** This counter is used by the controller to determine when Channel 4 has been denied access to a requested EU long enough to warrant immediate elevation to level 2 priority. **Note:** If set to zero, the CHN3_EU_PR_CTR must also be set to zero, and the controller assigns EUs on a purely round-robin basis. If set to non-zero, CHN3_EU_PR_CTR must also be set to a different, non-zero value.	
CHN3_BUS_PR_CNT 16–8	0	**Channel 3 Bus Priority Counter** This counter is used by the controller to determine when Channel 3 has been denied access to the bus long enough to warrant immediate elevation to level 2 priority. **Note:** If set to zero, the CHN4_BUS_PR_CTR must also be set to zero, and the controller assigns access to the bus on a purely round-robin basis. If set to non-zero, CHN4_BUS_PR_CTR must be set to a different, non-zero value.	
CHN4_BUS_PR_CNT 7–0	0	**Channel 4 Bus Priority Counter** This counter is used by the controller to determine when Channel 4 has been denied access to a needed on-chip resource long enough to warrant immediate elevation to level 2 priority. **Note:** If set to zero, the CHN3_BUS_PR_CTR must also be set to zero, and the controller assigns access to the bus on a purely round-robin basis. If set to non-zero, CHN3_BUS_PR_CTR must be set to a different, non-zero value.	

26.5.4.2 Controller Identification Register (CIDR)

CIDR — Controller Identification Register — Offset 0xC1020

Bits	63 … 0
	ID
Type	R
Reset	0x0030000200000000

The read-only ID register contains the same value as the IP Block Revision Register (see **Section 26.5.4.3**, *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.

26.5.4.3 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	0x00	0x02	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 2.1 block. **Table 26-12** describes the fields of the IP Block Revision Register.

Table 26-12. 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	0x00	**IP Major Revision Number** Major revision level value.	0x00 fixed value.
IP_MN 39–32	0x02	**IP Minor Revision Number** Minor revision level value.	0x02 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.

26.5.4.4 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	—			KEU	—	PKEU	—	RNG
Type	R							
Reset	0x00F			0x0	0xF	0x0	0xF	0x0

The EUASR is used to check the assignment status of an EU to a particular channel. When an EU is already assigned, it is inaccessible to any other channel. **Table 26-13** describes the fields of the EUASR.

Table 26-13. EUASR Bit Field Descriptions

Bits	Reset	Description	Settings
— 63–60	0xF	Reserved. Write to zero for future compatibility.	
AFEU 59–56	0x0	**AFEU** Indicates the AFEU channel assignment.	See **Table 26-14**.
— 55–52	0xF	Reserved. Write to zero for future compatibility.	
MDEU 51–48	0x0	**MDEU** Indicates the MDEU channel assignment.	See **Table 26-14**.
— 47–44	0xF	Reserved. Write to zero for future compatibility.	
AESU 43–40	0x0	**AESU** Indicates the AESU channel assignment.	See **Table 26-14**.
— 39–36	0xF	Reserved. Write to zero for future compatibility.	
DEU 35–32	0x0	**DEU** Indicates the DEU channel assignment.	See **Table 26-14**.
— 31–20	0x00F	Reserved. Write to zero for future compatibility.	
KEU 19–16	0x0	**KEU** Indicates the KEU channel assignment.	See **Table 26-14**.
— 15–12	0xF	Reserved. Write to ones for future compatibility.	
PKEU 11–8	0x0	**PKEU** Indicates the PKEU channel assignment.	See **Table 26-14**.
— 7–4	0xF	Reserved. Write to ones for future compatibility.	
RNG 3–0	0x0	**RNG** Indicates the RNG channel assignment.	See **Table 26-14**.

Table 26-14. Channel Assignment Value

Value	Channel
0x0	No channel assigned
0x1	Channel 1
0x2	Channel 2
0x3	Channel 3
0x4	Channel 4
0xA–0xE	Undefined
0xF	Unavailable

26.5.4.5 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	—															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	—						KEU		—		PKEU		—		RNG	
Subfield							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 26-15** describes the register field names in the CIER.

Table 26-15. CIER Bit Field Descriptions

Bits	Reset	Description	Settings
— 63–49	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.
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.
— 31–26	0	Reserved. Write to zero for future compatibility.	
KEU Err 25	0	**KEU Error Interrupt** Indicates whether the KEU generated an error.	0 Error interrupt disabled. 1 Error interrupt enabled.

Table 26-15. CIER Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
KEU Dn 24	0	**KEU Done** Indicates whether the KEU has completed its operation.	0 Done interrupt disabled. 1 Done interrupt enabled.
— 23–22	0	Reserved. Write to zero for future compatibility.	
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.	
RNG Err 17	0	**RNG Error Interrupt** Indicates whether the RNG generated an error.	0 Error interrupt disabled. 1 Error interrupt enabled.
RNG Dn 16	0	**RNG Done** Indicates whether the RNG 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.

Table 26-15. CIER Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
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.

26.5.4.6 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	—															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	—						KEU		—		PKEU		—		RNG	
Subfield							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 26.5.4.7, *Controller Interrupt Clear Register (CICR)*, on page 26-87. The CISR bit fields are described in **Table 26-16**.

Table 26-16. CISR Bit Field Descriptions

Bits	Reset	Description	Settings
— 63–49	0	Reserved. Write to zero for future compatibility.	
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.
— 31–26	0	Reserved. Write to zero for future compatibility.	
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.

Table 26-16. CISR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
— 23–22	0	Reserved. Write to zero for future compatibility.	
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.	
RNG Err 17	0	**RNG Error Interrupt** Indicates whether the RNG generated an error.	0 No error detected. 1 Error detected.
RNG Dn 16	0	**RNG Done** Indicates whether the RNG 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.

26.5.4.7 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	—															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	—						KEU		—		PKEU		—		RNG	
Subfield							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 26-17**.

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 26-17. CICR Bit Field Descriptions

Bits	Reset	Description	Settings
— 63–49	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.

Table 26-17. CICR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
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.
— 31–26	0	Reserved. Write to zero for future compatibility.	
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.
— 23–22	0	Reserved. Write to zero for future compatibility.	
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.
— 19–18	0	Reserved. Write to zero for future compatibility.	
RNG Err 17	0	**RNG Error Interrupt** Clears the RNG error status bit.	0 No action. 1 Clear status bit.
RNG Dn 16	0	**RNG Done** Clears the RNG 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.	

Table 26-17. CICR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
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.

26.5.5 Channel Registers and Structures

The following sections the registers and structures used by the four channels.

26.5.5.1 Channel Configuration Registers for Channels 1–4 (CCR[1–4])

CCR1 Channel Configuration Registers Offset 0xC1108
CCR2 Offset 0xC1208
CCR3 Offset 0xC1308
CCR4 Offset 0xC1408

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	—														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								BS	IWSE	—	EAE	CDWE	AWSE	NT	CDIE	—
Type	R/W															
Reset	0x0000															

Each CCR contains operational bits that permit configuration of the respective channel. **Table 26-18** describes each field of the CCR.

Table 26-18. CCR Bit Field Descriptions

Bits	Reset	Description	Settings
— 63–34	0	Reserved. Write to zero for future compatibility.	
CON 33	0	**Continue** 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 normal operation servicing the next descriptor pointer in the Fetch FIFO, if any is present.	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 by asserting the release output signals. The channel then resets all its registers, clears the R bit, and returns the channel state machine to the idle state. • 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 the EU. If a secondary EU has been reserved, the channel requests a write to reset that EU as well. The channel next asserts the appropriate release signal to notify the controller that the channel has finished with the reserved EU(s). The channel then resets all the registers, clears the R bit and returns the channel state machine to the idle state.	0 No special action. 1 Initiates a soft reset of the channel, clearing all of its internal state.
— 31–9	0	Reserved. Write to zero for future compatibility.	
BS 8	0	**Burst Size** This bit determines the burst size used by the SEC to access long text-data parcels in main memory.	0 Burst size is 64 bytes. 1 Burst size is 128 bytes.
IWSE 7	0	**ICV Writeback Status Enable** If this bit is set and the descriptor calls for ICV comparison, then at completion of descriptor processing, the channel writes back to the descriptor header all the required writeback information: DONE, ICCR0, and ICCR1 fields.	0 No special action. 1 Write back the required data after processing if ICV comparison is required.
— 6	0	Reserved. Write to zero for future compatibility.	
EAE 5	0	**Extend Address Enable** Selects whether to use 32-bit or 36-bit addressing.	0 Channel address bus is 32 bits. 1 Channel address bus is 36 bits.
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.

Table 26-18. CCR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
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 26.3.2**, *Channel Interrupts,* on page 26-22 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 will 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.

26.5.5.2 Channel Pointer Status Registers (CPSR[1–4])

CPSR1 Channel Pointer Status Registers Offset 0xC1110
CPSR2 Offset 0xC1210
CPSR3 Offset 0xC1310
CPSR4 Offset 0xC1410

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—			FF_COUNTER					—				G_STATE			
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—				S_STATE				CHN_STATE							
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—						MI	MO	PR	SR	PG	SG	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	SGDLZE	FPZE	IDH	IEUAR	EUE	GBE	GRLE	SBE	SRLE	DWORD_NUM			
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. **Table 26-19** describes the CPSR fields.

Table 26-19. CPSR Bit Field Descriptions

Bits	Reset	Description	Settings
— 63–61	0	Reserved. Write to zero for future compatibility.	
FF_ COUNTER 60–56	0	**Fetch File Counter** This 5-bit counter indicates the number of fetch pointers currently stored in the FIFO	
— 55–52	0	Reserved. Write to zero for future compatibility.	
G_STATE 51–48	0	**Gather State Machine State** Reflects the current state of the channel gather control state machine. The value in this field indicates the current stage of the channel performing a gather function. This information is included for completeness. The user does not typically require this information.	See **Table 26-20** for a detailed list of values and their meanings.
— 47–44	0	Reserved. Write to zero for future compatibility.	

Table 26-19. CPSR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
S_STATE 43–40	0	**Scatter State Machine State** Reflects the current state of the channel scatter control state machine. The value in this field indicates the current stage of the channel performing a scatter function. This information is included for completeness. The user does not typically require this information.	See **Table 26-20** for a detailed list of values and their meanings.
CHN_STATE 39–32	0	**Channel State Machine State** Reflects the current state of the channel control state machine. The value in this field indicates the current stage of the channel performing in the sequence of fetching and processing data descriptors. This information is included for completeness. The user does not typically require this information.	See **Table 26-21** for a detailed list of values and their meanings.
— 31–26	0	Reserved. Write to zero for future compatibility.	
MI 25	0	**Multi-EU Input** Indicates whether data input snooping is performed, as determined by the descriptor header.	0 Data input snooping disabled. 1 Data input snooping by secondary EU enabled.
MO 24	0	**Multi-EU Output** Indicates whether data output snooping is performed, as determined by the descriptor header.	0 Data output snooping disabled. 1 Data output snooping by secondary EU enabled.
PR 23	0	**Primary EU Assignment Request** This bit is set when descriptor processing is initiated by the channel and the Op_0 field in the descriptor header contains a valid EU identifier. The bit is cleared when the request is granted, which is reflected by the setting of the PG bit.	0 Primary EU assignment request is inactive. 1 Channel is requesting the assignment of a primary EU.
SR 22	0	**Secondary EU Assignment Request** This bit is set when descriptor processing is initiated by the channel and the Op_1 field in the descriptor header contains a valid EU identifier. The bit is cleared when the request is granted, which is reflected by the setting of the SG bit.	0 Secondary EU assignment request is inactive. 1 Channel is requesting the assignment of a security EU.
PG 21	0	**Primary EU Granted** Reflects the status of EU grant signal for the requested primary EU from the controller.	0 Primary EU grant signal is inactive. 1 Primary EU grant signal is active, indicating that the controller assigned the primary EU to this channel.
SG 20	0	**Secondary EU Granted** Reflects the status of EU grant signal for the requested secondary EU from the controller.	0 Secondary EU grant signal is inactive. 1 Secondary EU grant signal is active, indicating that the controller assigned the secondary EU to this channel.

Table 26-19. CPSR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
PRD 19	0	**Primary Reset Done** Reflects the state of the reset done signal from the assigned primary EU.	0 Assigned primary EU reset done signal inactive. 1 Assigned primary EU reset done signal active. Reset sequence complete and EU is ready to accept data.
SRD 18	0	**Secondary Reset Done** Reflects the state of the reset done signal from the assigned secondary EU.	0 Assigned secondary EU reset done signal inactive. 1 Assigned secondary EU reset done signal active. Reset sequence complete and EU is ready to accept data.
PD 17	0	**Primary EU Done** Reflects the state of the done interrupt from the assigned primary EU.	0 Assigned primary EU done interrupt inactive. 1 Assigned primary EU done interrupt active. EU processing complete and EU is ready to provide output data.
SD 16	0	**Secondary EU Done** Reflects the state of the done interrupt from the assigned secondary EU.	0 Assigned secondary EU done interrupt inactive. 1 Assigned secondary EU done interrupt active. EU processing complete and EU is ready to provide output data.
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.
SGDLZE 12	0	**Scatter/Gather Data Length Zero Error** Indicates that an all zero scatter/gather pointer was detected.	0 No error detected. 1 Error detected.
FPZE 11	0	**Fetch Pointer Zero Error** Indicates that an all zero fetch pointer was detected.	0 No error detected. 1 Error detected.

Table 26-19. CPSR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
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.
IEUAR 9	0	**Invalid EU Assignment Request** Set when an assigned EU does not match any of the EUs requested by the descriptor header.	0 No error detected. 1 Error detected.
EUE 8	0	**EU Error** Set when an EU assigned to this channel generates an error interrupt. The error may 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.
GBE 7	0	**Gather Boundary Error** Set when the gather pointer straddles both a primary and secondary EU data transfer.	0 No error detected. 1 Error detected.
GRLE 6	0	**Gather Run/Length Error** Set when the total data size covered by a gather link table does not match the total data size from the main descriptor.	0 No error detected. 1 Error detected.
SBE 5	0	**Scatter Boundary Error** Set when the scatter pointer straddles both a primary and secondary EU data transfer.	0 No error detected. 1 Error detected.
SRLE 4	0	**Scatter Run/Length Error** Set when the total data size covered by a scatter link table does not match the total data size from the main descriptor.	0 No error detected. 1 Error detected.
DWORD_NUM 3–0	0	**Pointer Number** Indicates the pointer that is currently being processed by the channel.	0x00 Header or Pointer 0 0x01 Pointer 1 0x02 Pointer 2 0x03 Pointer 3 0x04 Pointer 4 0x05 Pointer 5 0x06 Pointer 6 0x07 Complete/not yet started All other values are reserved.

Table 26-20. 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 26-21. CHN_STATE Field Values

Value	Channel State
0x00	IDLE
0x01	PROCESS_HEADER
0x02	FETCH_DESCRIPTOR
0x03	CHANNEL_DONE
0x04	CHANNEL_DONE_IRQ
0x05	CHANNEL_DONE_WRITEBACK
0x06	CHANNEL_DONE_NOTIFICATION
0x07	CHANNEL_ERROR
0x08	REQUEST_PRI_CHA
0x09	INC_DATA_PAIR_POINTER
0x0A	DELAY_DATA_PAIR_UPDATE
0x0B	EVALUATE_DATA_PAIRS
0x0C	WRITE_RESET_PRI
0x0D	RELEASE_PRI_CHA
0x0E	WRITE_RESET_SEC
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

Table 26-21. CHN_STATE Field Values (Continued)

Value	Channel State
0x19	WRITE_KEY_SIZE
0x1A	WRITE_CHA_GO
0x1B	DELAY_PRI_DONE
0x1E	WRITE_DATASIZE_SEC_SNOOPOUT
0x1F	TRANS_REQUEST_READ_SNOOPOUT
0x20	DELAY_SEC_DONE
0x21	TRANS_REQUEST_READ
0x22	EVALUATE_RESET
0x23	RESET_WRITE_RESET_PRI
0x24	RESET_RELEASE_PRI_CHA
0x25	RESET_WRITE_RESET_SEC
0x26	RESET_RELEASE_SEC_CHA
0x27	RESET_CHANNEL
0x28	WRITE_DATASIZE_PRI_POST
0x29	RESET_RELEASE_ALL
0x2A	RESET_RELEASE_ALL_DELAY
0x2B	REQUEST_SEC_CHA
0x2C	WRITE_DATASIZE_SEC
0x2D	WRITE_ICV_SIZE
0x2E	WRITE_SEC_CHA_GO_SNOOPOUT
0x2F	WRITE_PRI_CHA_GO_SNOOPIN
0x30	WRITE_SEC_CHA_GO_SNOOPIN
0x31	DELAY_1CYCLE
0x33	TRANS_EXTENT_READ
0x34	TRANS_EXTENT3
0x35	TRANS_EXTENT4
0x36	XOR_WRITE_READ_REG
0x37	DELAY_SEC_DONE_TLS
0x38	MAC_TO_CIPHER
0x39	MAC_TO_CIPHER_DONE
0x3A	READ_PRI_STATUS
0x3C	READ_SEC_STATUS
others	Reserved

26.5.5.3 Current Descriptor Pointer Register (CDPR)

CDPR1	Current Descriptor Pointer Registers	Offset 0xC1140
CDPR2		Offset 0xC1240
CDPR3		Offset 0xC1340
CDPR4		Offset 0xC1440

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	—												EPTR			
Type	R															
Reset	0x0000															

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	0x0000															

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	0x0000															

The CDPR contains the address of the descriptor that the channel is currently processing. The bits in the CDPR perform the functions described in **Table 26-22**.

Table 26-22. CDPR Bit Field Descriptions

Bits	Reset	Description
— 63–36	0	Reserved. Write to zero for future compatibility.
EPTR 35–32	0	**Extended Pointer** Concatenated as the upper 4 bits of the pointer address when extended mode is selected (EAE is high—see **Table 26-18** 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.

26.5.5.4 Channel Fetch FIFO (CFF)

CFF1 Channel Fetch FIFOs Offset 0xC1148
CFF2 Offset 0xC1248
CFF3 Offset 0xC1348
CFF4 Offset 0xC1448

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 26-18** 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 bits in the CFF perform the functions described in **Table 26-22**.

Table 26-23. 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 26-18** 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.

26.5.5.5 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 26-9. Descriptor Format

For more information about the fields in a descriptor, see **Section 26.2.1.1.1**, *Descriptor Structure*, on page 26-10.

Note: The DBs are located at the following locations:
Channel 1: Offsets: 0xC1180–0xC11BF
Channel 2: Offsets 0xC1280–0xC12BF
Channel 3: Offsets 0xC1380–0xC13BF
Channel 4: Offsets 0xC1480–0xC14BF

26.5.6 PKEU Registers

26.5.6.1 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 26-24** describes the PKEUMR fields.

Table 26-24. 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 26-29** for values. See **Section 26.4.1.11**, *PKEU Routines,* on page 26-27 for details.

Table 26-25. 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 26-25. 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: Exponential mod N (combines MOD_R2MODN, POLY_F2M_MULT1_MONT, and MOD_EXP)
0xFF	SPK_BUILD	Build PK data structure (data structure used by all elliptic curve routines)

26.5.6.2 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 26-26. 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.

26.5.6.3 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 may 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.

An illegal data size error will be generated as follows:

- All non-ECC routines with a data size > 256 generate an illegal data size error.
- All ECC routines with a data size > 64 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 26-27. 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.

26.5.6.4 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 97 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.

Table 26-28. 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.

26.5.6.5 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 26-29. 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 26.5.6.6**, *PKEU Status Register (PKEUSR),* on page 26-107).	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 26.5.6.6**, *PKEU Status Register (PKEUSR),* on page 26-107).	0 No reset 1 Full PKEU reset.

26.5.6.6 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	—									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 26-30. PKEUSR Field Descriptions

Name	Reset	Description	Settings
— 63–7	0	Reserved. Write to zero for future compatibility.	
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 may 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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84).	0 PKEU is not signaling error 1 PKEU is signaling error

Table 26-30. 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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84)	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.

26.5.6.7 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	—	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 26.4.1.8**, *PKEU Interrupt Mask Register*, on page 26-26).

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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84). 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 pointer Status Register (see **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4]),* on page 26-92) 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 26-31**.

Table 26-31. 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.
— 11	0	Reserved. Write to zero for future compatibility.	
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.	

26.5.6.8 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	—	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 26.4.1.7**, *PKEU Interrupt Status Register*, on page 26-25), 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 26-32. 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.
— 11	0	Reserved. Write to zero for future compatibility.	
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 26-32. 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.	

26.5.6.9 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.

26.5.6.10 PKEU Parameter Memories

The PKEU uses four 2048-bit memories to receive and store operands for the arithmetic operations the PKEU will be 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.

26.5.6.10.1 PKEU Parameter Memory A

This 2048 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 512 bit memories, and is used to specify particular curve parameters and input values.

26.5.6.10.2 PKEU Parameter Memory B

This 2048 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 512 bit memories, and is used to specify particular curve parameters and input values, as well as to store result values.

26.5.6.10.3 PKEU Parameter Memory E

This 2048 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 will cause address error to be reflected in the PKEU Interrupt Status Register.

26.5.6.10.4 PKEU Parameter Memory N

This 2048 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.

26.5.7 DEU Registers

26.5.7.1 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	—													CE	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 26-33. DEUMR Field Descriptions

Name	Reset	Description	Settings
— 63–3	0	Reserved. Write to zero for future compatibility.	
CE 2	0	**CBC/ECB** Specifies whether the DEU operates in cipher-block-chaining (CBC) or electronic code book (ECB) mode.	0 ECB mode. 1 CBC 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.

26.5.7.2 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 26-34. 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.

26.5.7.3 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 26-35. 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.

26.5.7.4 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 26-36**.

Table 26-36. 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 26.5.7.5**, *DEU Status Register (DEUSR),* on page 26-117).	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 26.5.7.5**, *DEU Status Register (DEUSR),* on page 26-117).	0 No reset 1 Full DEU reset.

26.5.7.5 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 26-37. 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 may 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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84).	0 DEU is not signaling error 1 DEU is signaling error

Table 26-37. 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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84)	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.

26.5.7.6 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 26.4.2.7**, *DEU Interrupt Mask Register*, on page 26-39). 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 26.5.4.6**, *Controller Interrupt Status Register (CISR)*, on page 26-84). 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 pointer Status Register (see **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4])*, on page 26-92) 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 26.5.7.4**, *DEU Reset Control Register (DEURCR)*, on page 26-116). The definition of each bit in the DEU Interrupt Status Register is listed in **Table 26-38**.

Table 26-38. 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 26-38. 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.

26.5.7.7 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 26.4.2.6**, *DEU Interrupt Status Register*, on page 26-39), 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 26-39. 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.

26.5.7.8 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 will be 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) will be 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.

26.5.7.9 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.

26.5.7.10 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 may 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.

26.5.7.11 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 have been 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

26.5.8 AESU Registers

26.5.8.1 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		FN	IM	RDK	CM		ED
Type	R/W															
Reset	0x0000															

The AESU Mode Register contains 7 bit fields that are 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 26-40** describes AESU Mode Register fields.

Note: The lowest 8 bits in this register are controlled by the MODE0 field in the descriptor header.

Table 26-40. AESUMR Field Descriptions

Name	Reset	Description	Settings
— 63–13	0	Reserved. Write to zero for future compatibility.	
SCM 12	0	**Sub-Cipher Mode** Specifies the number of sources to be XORd together for specific cipher modes. **Note:** This field is included for information purposes only. It is not under direct user control.	*XOR Cipher Mode*: 2 and 3 are valid. All other values reserved. *Other cipher modes*: 0 is valid. All other values reserved.
— 9–8	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 26-41** for details.
FM 5	0	**Final MAC** Determines whether to process the final message block and generate the final MAC tag during end-of-message processing (CCM mode only).	0 Do not generate final MAC tag. 1 Generate final MAC tag after CCM processing is complete.

Table 26-40. AESUMR Field Descriptions (Continued)

Name	Reset	Description	Settings
IM 4	0	**Initialize MAC** Used to initialize the AESU for a new message (CCM mode only).	0 Do not initialize (context is loaded by core processor). 1 Initialize new message with nonce.
RDK 3	0	**Restore Decrypt Key** Specifies that key data writes contain pre-expanded key (decrypt mode only). **Note:** In most networking applications, the decryption of an AES protected packet is performed as a single operation. However, if circumstances dictate that the decryption of a message should be split across multiple descriptors, the AESU allows the user to save the decrypt key, and the active AES context, to memory for later re-use. This saves the internal AESU processing overhead associated with regenerating the decryption key schedule (~12 AESU clock cycles for the first block of data to be decrypted.). For details, see **Section 26.4.3.1**, *AESU Mode Register,* on page 26-41	0 Expand the user key prior to decrypting the first block. 1 Do not expand the key. The expanded decryption key is written after the context switch.
CM 2–1	0	**Cipher Mode** Used in combination with the ECM field to define the AES operating mode.	See **Table 26-41** for details.
ED 0	0	**Encryption/Decryption** Specifies whether to encrypt or decrypt the data.	0 Perform decryption. 1 Perform encryption.

Table 26-41. AES Cipher Modes

Mode	ECM	CM
ECB	00	00
CBC	00	01
CTR	00	11
SRT	01	11
CCM (without ICV comparison)	10	00
CCM with ICV comparison	11	00
XOR	11	11
Reserved	all others	

Note: 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 26.4.3.9.3**, *Context for SRT Mode,* on page 26-44 for more information on how SRT mode reduces context loading overhead.

26.5.8.2 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 26-42. 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.

26.5.8.3 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	—															
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 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.

Table 26-43. DEUDSR Field Descriptions

Name	Reset	Description
— 63–12	0	Reserved. Write to zero for future compatibility.
Data Size 11–0	0	**Data Size** Stores the number of bits in the final message block. The required value is mode dependent: • ECB and CBC modes: Multiple of 128 bits. • CCM and CTR modes: Multiple of 8 bits. • XOR mode: Multiple of 256 bits (32 bytes). **Note:** Entry of an improper data size results in the setting of 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.

26.5.8.4 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 26-44. 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 26.5.8.5**, *AESU Status Register (AESUSR),* on page 26-129).	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 26.5.8.5**, *AESU Status Register (AESUSR),* on page 26-129).	0 No reset 1 Full AESU reset.

26.5.8.5 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 will result in an address error being reflected in the AESU Interrupt Status Register.

Table 26-45. 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 RNG is halted due to an error. **Note:** Because the error causing the RNG to stop operating may be masked before reaching the Interrupt Status Register, the RNG Interrupt Status Register is used to provide a second source of information regarding errors that prevent normal operation.	0 RNG not halted 1 RNG 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 26-45. 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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84).	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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84)	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.

26.5.8.6 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 26.4.3.7**, *AESU Interrupt Mask Register*, on page 26-43). 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 26.5.4.6**, *Controller Interrupt Status Register (CISR)*, on page 26-84). 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 pointer Status Register (see **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4])*, on page 26-92) 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 26-46**.

Table 26-46. 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 26.4.3.3**, *AESU Data Size Register,* on page 26-42 for details.	0 No data size error detected. 1 Data size error.
ME 7	0	**Mode Error** If set, indicates that invalid dRev. 3ata 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 26-46. 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.	

26.5.8.7 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 26.4.3.6**, *AESU Interrupt Status Register*, on page 26-43), 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 26-47. 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.	

26.5.8.8 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 may 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.

26.5.8.9 AESU Context Registers (AESUCR[1–7])

There are seven 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 (CBC, CTR and CCM modes). For CTR and CCM mode, all seven 64-bit Context Registers must be read to retrieve context and all seven must be written back to restore context. Effectively, the user must read the four empty place holder Context Registers in addition to the three Context Registers holding the Counter and Counter Modulus Exponent when in CTR mode. 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 with the lower bytes in the lowest 64-bit Context Register. The Context Registers are summarized in **Figure 26-10**.

Context Register (64 bits each)	Offset	Cipher Mode				
		ECB	CBC	CTR	SRT	CCM
1	0xC4100	—	IV1[1]	—	Counter[1]	IV[1]/MAC Tag
2	0xC4108	—	IV2[1]	—		
3	0xC4110	—	—	—	Counter Modulus Exponent (M)[1]	Encrypted MAC[2]/ Decrypted MAC/ Encrypted Counter
4	0xC4118	—	—	—	—	
5	0xC4120	—	—	Counter[1]	—	Counter[1]
6	0xC4128	—	—		—	
7	0xC4130	—	—	Counter Modulus Exponent[1]	—	Counter Modulus Exponent[1]/ header size/ MAC size[3]

Notes: **1.** Must be written at the start of a new message.
2. Must be written at the start of a new CCM decryption.
3. Header size/MAC size is only used if AES-CCM processing is suspended and resumed.

Figure 26-10. AESU Context Registers

Note: See **Section 26.4.3.9**, *AESU Context Registers*, on page 26-44 for details on how to use and program the Context Registers.

26.5.8.10 AESU Key Registers (AESUKR[1–3])

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 1. 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.

The Key Data Registers may be read when changing context in decrypt mode. To resume processing, the value read must be written back to the key registers and the restore decrypt key bit must be set in the Mode Register. This eliminates the overhead of expanding the key prior to starting decryption when switching context.

Note: The AESU key registers are located at the following offsets:
AESUKR1 = Offset 0xC4400.
AESUKR2 = Offset 0xC4408
AESUKR3 = Offset 0xC4410

26.5.8.11 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 bytes, or 8 bytes. When all 8 bytes of the staging register have been 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 have been 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 may 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 may be read any time the OFR signal is asserted (as indicated in the AESU Status Register). This will indicate 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. 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

26.5.9 MDEU Registers

26.5.9.1 MDEU Mode Register (MDEUMR)

MDEUMR — MDEU Mode Register — 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	—					STIB	NEW	—	CONT	CICV	SMAC	INIT	HMAC	PD	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 26-48** describes MDEU Mode Register fields.

Table 26-48. MDEUMR Field Descriptions

Name	Reset	Description	Settings
— 63–11	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.

Table 26-48. MDEUMR Field Descriptions (Continued)

Name	Reset	Description	Settings
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.
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 2	0	**Padding** For Old configurations (NEW = 0), the value of this bit must be programmed to be the inverse of the CONT bit. For New configurations (NEW = 1), this bit must be cleared (0).	*For 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 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.

26.5.9.2 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 64 bytes of key. MDEU generate a key size error if the value written to this register exceeds 64 bytes.

26.5.9.3 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	—															
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 has been 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 512-bit block size (that is, the least significant 8 bits must be written as 0). 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.

26.5.9.4 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 26-49**.

Table 26-49. 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.

Table 26-49. MDEURCR Field Descriptions (Continued)

Name	Reset	Description	Settings
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.

26.5.9.5 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 26-50. 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 may 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

Table 26-50. MDEUSR Field Descriptions (Continued)

Name	Reset	Description	Settings
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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84).	0 MDEU is not signaling error 1 MDEU 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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84)	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.

26.5.9.6 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 26.5.10.7**, *AFEU Interrupt Mask Register (AFEUIMR),* on page 26-157).

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 26.5.4.6**, *Controller Interrupt Status Register (CISR)*, on page 26-84). 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 Status Register (see **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4])*, on page 26-92) 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 26-51**.

Table 26-51. MDEUISR 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 MDEU.	0 No error detected. 1 Integrity check error.
— 13	0	Reserved. Write to zero for future compatibility.	
IE 12	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 11	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 10	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 9	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 8	0	**Data Size Error** If set, indicates that data with a size not a multiple of 512 bits was found when the MDEUMR[CONT} bit was cleared (0).	0 No data size error detected. 1 Data size error.

Table 26-51. MDEUISR Field Descriptions (Continued)

Name	Reset	Description	Settings
ME 7	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 6	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.
— 5–3	0	Reserved. Write to zero for future compatibility.	
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.	

26.5.9.7 MDEU Interrupt Mask Register (MDEUIMR)

MDEUIMR — MDEU 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	—	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 26.4.4.6**, *MDEU Interrupt Status Register*, on page 26-51), 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 26-52. MDEUIMR 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.
— 5–3	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.	

26.5.9.8 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 26.4.4.1**, *MDEU Mode Register*, on page 26-49). This register is cleared when the MDEU is reset or reinitialized.

26.5.9.9 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 may 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.

26.5.9.10 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: SHA-1and SHA-256 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 to 0. **Figure 26-11** shows how the MDEU Context Registers are initialized if the INIT bit is set in the MDEU Mode Register. All registers are initialized, regardless of mode selected, however only the appropriate Context Register values are used in hash generation per the mode selected. The user typically does not care about the MDEU Context Register initialization values; they are documented for completeness in the event the user reads these registers using core processor-controlled access. MDEU reset via the MDEU Reset Control Register (**Section 26.5.9.4**, *MDEU Reset Control Register (MDEURCR)*, on page 26-140) or SEC global software reset (**Section 26.5.4.1**, *Master Control Register (MCR)*, on page 26-77) does not clear these registers.

	63 — 32	31 — 0	Context Offset
Name	**A**	**B**	0xC_6100
MD-5	0x01234567	0x89ABCDEF	
SHA-1	0x67452301	0xEFCDAB89	
SHA-224	0xC1059ED8	0x367CD507	
SHA-256	0x6A09E667	0xBB67AE85	
Name	**C**	**D**	0xC_6108
MD-5	0xFEDCBA98	0x76543210	
SHA-1	0x98BADCFE	0x10325476	
SHA-224	0x3070DD17	0xF70E5939	
SHA-256	0x3C6EF372	0xA54FF53A	
Name	**E**	**F**	0xC_6110
MD-5	0xF0E1D2C3	0x8C68059B	
SHA-1	0xC3D2E1F0	0x9B05688C	
SHA-224	0xFFC00B31	0x68581511	
SHA-256	0x510E527F	0x9B05688C	
Name	**G**	**H**	0xC_6118
MD-5	0xABD9831F	0x19CDE05B	
SHA-1	0x1F83D9AB	0x5BE0CD19	
SHA-224	0x64F98FA7	0xBEFA4FA4	
SHA-256	0x1F83D9AB	0x5BE0CD19	
Name	**Message Length Count**		0xC_6120
Reset	0		

Figure 26-11. MDEU Context Register

26.5.9.11 MDEU Key Registers (MDEUKR[1–8])

The MDEU maintains eight 64-bit registers for writing an HMAC key. The IPAD and OPAD operations are performed automatically on the key data when required.

Note: SHA-1 and SHA-256 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.

26.5.9.12 MDEU Input FIFO

MDEU uses an input FIFO to hold data to be hashed. 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 by byte, 4 bytes, or 8 bytes. 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, this set is 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 bytes. Any last bytes remaining 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.

26.5.10 AFEU Registers

26.5.10.1 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 26-53** describes AFEU Mode Register fields.

Table 26-53. 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.

26.5.10.2 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.

26.5.10.3 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.

26.5.10.4 AFEU Reset Control Register (AFEURCR)

AFEURCR — AFEU 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 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 26.5.10.5**, *AFEU Status Register (AFEUSR)*, on page 26-154).

Table 26-54. 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.

26.5.10.5 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 26.5.10.6**, *AFEU Interrupt Status Register (AFEUISR)*, on page 26-155).

Table 26-55. 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 may 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 26.5.4.6**, *Controller Interrupt Status Register (CISR)*, on page 26-84).	0 AFEU is not signaling error 1 AFEU is signaling error

Table 26-55. 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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84).	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.

26.5.10.6 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 26.4.5.7**, *AFEU Interrupt Mask Register*, on page 26-56).

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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84). 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 Status Register (see **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4]),* on page 26-92) 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 26-56**.

Table 26-56. 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 26-56. 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.	

26.5.10.7 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 26.4.5.6**, *AFEU Interrupt Status Register*, on page 26-56), 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 26-57. AFEUIMR Field Descriptions

Name	Reset	Description	Settings
— 63–13	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.	

26.5.10.8 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 has been 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.

26.5.10.9 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 26.5.10.1**, *AFEU Mode Register (AFEUMR)*, on page 26-150). After context data is written, the context length must be written to the context/data length register (see **Section 26.5.10.3**, *AFEU Context/Data Size Register (AFEUCDSR)*, on page 26-152). 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 26.4.5.1**, *AFEU Mode Register*, on page 26-54). See **Section 26.5.10.12**, *AFEU FIFOs*, on page 26-160 for details about AFEU FIFO addressing.

Note: The AFEU Context Memory is in the range defined by offsets 0xC8100–0xC81FF.

26.5.10.10 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 will be 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.

26.5.10.11 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.

26.5.10.12 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 0xC8E00. 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 have been 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 have been 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.

26.5.11 KEU Registers

26.5.11.1 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 26-58** describes the KEU Mode Register bit fields.

Table 26-58. 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 host 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 host 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 26-58. 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

26.5.11.2 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.

26.5.11.3 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 host. 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.

26.5.11.4 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 26-59. 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 26.5.11.5**, *KEU Status Register (KEUSR),* on page 26-166).	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 26.5.11.5**, *KEU Status Register (KEUSR),* on page 26-166).	0 Normal operation. 1 Full KEU reset.

26.5.11.5 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 will result in an address error being reflected in the KEU Interrupt Status Register.

Table 26-60. 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 may 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 26-60. 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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84).	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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84).	0 KEU is not signaling done 1 KEU is signaling done
RD 0	0	**Reset Done** This status bit, when high, indicates that the RNG 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.

26.5.11.6 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 26.4.6.7**, *KEU Interrupt Mask Register*, on page 26-61).

If the AESU Interrupt Status Register is non-zero, the KEU halts and the KEU error interrupt signal is asserted to the controller (see **Section 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84). 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 Status Register (see **Section 26.5.5.2**, *Channel Pointer*

Status Registers (CPSR[1–4]), on page 26-92) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the host, 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 26.5.11.4**, *KEU Reset Control Register (KEURCR)*, on page 26-165).

The definition of each bit in the KEU Interrupt Status Register is listed in **Table 26-61**.

Table 26-61. 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.
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 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 26-61. 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.	

26.5.11.7 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 26.4.6.6**, *KEU Interrupt Status Register*, on page 26-60), 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 26-62**.

Table 26-62. 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.	

26.5.11.8 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 may 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.

26.5.11.9 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 has been 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.

26.5.11.10 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	CE															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	CA							—		CD	CB					
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	CC															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	CC															
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 26-63. 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.

26.5.11.11 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.

26.5.11.12 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	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 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.

26.5.11.13 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 host 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.

26.5.11.14 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 will occur. Reading from either of these registers results in an address error being reflected in the KEUISR.

26.5.11.15 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 will occur.

If the ‘F9 only’ mode is set, the integrity key data may be optionally written to KEUKDR[1–2]. This eliminates the need for the host 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.

26.5.11.16 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 host-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 0xCE800–0CEFFF.

26.5.12 RNG Registers

26.5.12.1 RNG Mode Register (RNGMR)

RNGMR — RNG 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 RNG 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.

26.5.12.2 RNG Data Size Register (RNGDSR)

RNGDSR — RNG 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 RNG Data Size Register is used to tell the RNG to begin generating random data. The actual content of the Data Size Register does not affect the operation of the RNG. After a reset and prior to the first write of data size, the RNG builds entropy without pushing data onto the FIFO. Once the Data Size Register is written, the RNG begins pushing data onto the FIFO. Data is pushed onto the FIFO every 256 cycles until the FIFO is full. The RNG then attempts to keep the FIFO full.

26.5.12.3 RNG Reset Control Register (RNGRCR)

RNGRCR — RNG 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 RNG. 26-64 describes RNGRCR fields.

Table 26-64. 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 RNG interrupts signalling done and error to reset. It further resets the state of the RNG Interrupt Status Register.	0 No reset. 1 Reset interrupt logic.
MI 1	0	**Module Initialization** Setting this reinitializes the RNG 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 RNG
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 RNG. All registers and internal states are returned to the defined reset state.	0 No reset 1 Full RNG reset.

26.5.12.4 RNG Status Register (RNGSR)

RNGSR — RNG 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	—	RD
Type	R															
Reset	0x0000															

This RNGSR contains 6 fields that reflect the state of the RNG internal signals. The RNGSR is read-only. Writing to this location results in an address error being reflected in the RNGISR. **Table 26-65** describes RNGSR fields.

Table 26-65. 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 RNG is halted due to an error. **Note:** Because the error causing the RNG to stop operating may be masked before reaching the Interrupt Status Register, the RNG Interrupt Status Register is used to provide a second source of information regarding errors that prevent normal operation.	0 RNG not halted 1 RNG 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 26.5.4.6**, *Controller Interrupt Status Register (CISR),* on page 26-84).	0 RNG is not signaling error 1 RNG is signaling error

Table 26-65. RNGSR Field Descriptions (Continued)

Name	Reset	Description	Settings
— 1	0	Reserved. Write to zero for future compatibility.	
RD 0	0	**Reset Done** This status bit, when high, indicates that the RNG 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.

26.5.12.5 RNG Interrupt Status Register (RNGISR)

RNGISR — RNG 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 RNG Interrupt Mask Register is zero (see **Section 26.4.7.6**, *RNG Interrupt Mask Register*, on page 26-65). If the RNG Interrupt Status Register is non-zero, the RNG halts and the RNG error interrupt signal is asserted to the controller (see **Section 26.4.7.5**, *RNG Interrupt Status Register*, on page 26-64). In addition, if the RNG 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 Status Register (see **Section 26.5.5.2**, *Channel Pointer Status Registers (CPSR[1–4])*, on page 26-92) 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 RNG Reset Control Register. **Table 26-66** describes RNGISR fields.

Table 26-66. 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 RNG 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** RNG 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.	

26.5.12.6 RNG Interrupt Mask Register (RNGIMR)

RNGIMR — RNG 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 RNG Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 26.4.7.5**, *RNG Interrupt Status Register*, on page 26-64), 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 26-67** describes RNG Interrupt Status Register fields.

Table 26-67. 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 RNG 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** RNG 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.	

26.5.12.7 RNG End_of_Message Register (RNGEOMR)

RNGEOMR — RNG 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 RNG End_of_message is a writable location but serves no function in the RNG. It is documented for the sake of consistency with the other EUs.

26.5.12.8 RNG Output FIFO

RNG 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. For core processor-controlled operation, a read from anywhere in the RNG FIFO address space dequeues data from the RNG output FIFO.

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.

Underflows caused by reading or writing the RNG output FIFO are reflected in the RNG Interrupt Status Register. Also, a write to the RNG output FIFO space will be reflected as an addressing error in the RNG Interrupt Status Register.

Note: The RNG output FIFO is located at offset 0xCA800–0CAFFF.