

MPC8360E

PowerQUICC II Pro Integrated

Communications Processor

Family Reference Manual

Supports
MPC8360E
MPC8358E

MPC8360ERM
Rev. 3
05/2010

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, ColdFire, PowerQUICC, StarCore, and Symphony are trademarks of Freescale Semiconductor, Inc. Reg. U.S. Pat. & Tm. Off. CoreNet, QorIQ, QUICC Engine, and VortiQa are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. IEEE nnn, nnn, and nnn are registered trademarks of the Institute of Electrical and Electronics Engineers, Inc. (IEEE). This product is not endorsed or approved by the IEEE. © 2010 Freescale Semiconductor, Inc.



Contents

Paragraph Number	Title	Page Number
About This Book		
Chapter 1		
Overview		
1.1	MPC8360E PowerQUICC II Pro Processor Overview	1-1
1.2	MPC8360E Architecture Overview	1-7
1.2.1	Power Architecture Core	1-7
1.2.2	QUICC Engine 2.0 Block	1-10
1.2.2.1	Examples of Chip-Level Pin Multiplexing	1-11
1.2.2.2	Differences Between the QUICC Engine Block and the MPC82xx/85xx CPM ...	1-12
1.2.2.3	Enhanced Features of the QUICC Engine Block Compared with the CPM	1-13
1.2.2.4	Software Migration from the MPC82xx/MPC85xx Family Devices	1-15
1.2.2.5	Serial Protocol Table	1-16
1.2.2.6	QUICC Engine Configurations	1-16
1.2.3	Security Engine	1-17
1.2.4	Dual DDR Memory Controllers	1-17
1.2.5	PCI Controller	1-18
1.2.5.1	PCI Bus Arbitration Unit	1-18
1.2.6	Local Bus Controller (LBC)	1-19
1.2.7	Integrated Programmable Interrupt Controller (IPIC)	1-20
1.2.8	Dual I ² C Interfaces	1-20
1.2.9	DMA Controller	1-21
1.2.10	Dual Universal Asynchronous Receiver/Transmitter (DUART)	1-21
1.2.11	System Timers	1-22
1.3	Application Examples	1-22
1.3.1	SME Router	1-22
1.3.2	DSLAM Line Card	1-23
1.3.3	NodeB/BTS—Network Interface Card	1-24

Chapter 2 Memory Map

2.1	Internal Memory Mapped Registers	2-1
2.2	Accessing IMMR Memory From the Local Processor	2-1
2.3	Complete IMMR Map	2-1
2.4	QUICC Engine Internal Memory Map	2-23

Contents

Paragraph Number	Title	Page Number
Chapter 3		
Signal Descriptions		
3.1	Signals Overview	3-1
3.2	Configuration Signals Sampled at Reset	3-18
3.3	Output Signal States During Reset	3-18
3.4	Parallel I/O Ports	3-20
3.4.1	Features	3-20
3.4.2	Port Registers	3-21
3.4.2.1	QUICC Engine Port Open-Drain Registers (CPODRA–CPODRG)	3-21
3.4.2.2	QUICC Engine Port Data Registers (CPDATA–CPDATG)	3-22
3.4.2.3	QUICC Engine Port Direction Registers (CPDIRxA–CPDIRxG)	3-23
3.4.2.4	QUICC Engine Port Pin Assignment Registers (CPPARxA–CPPARxG)	3-23
3.4.2.5	Communication Peripherals to QUICC Engine Mux Control Registers	3-24
3.4.2.6	QUICC Engine Port Output Hold Registers (CPOH1, CPOH2)	3-25
3.4.3	Port Block Diagram	3-27
3.4.4	Port Pins Functions	3-28
3.4.4.1	General-Purpose I/O Pins	3-28
3.4.4.2	Dedicated Pins	3-28
3.4.5	PCI Pins	3-28
3.4.6	QUICC Engine Ports Interrupts	3-29
3.4.7	Gigabit Ethernet Pins	3-29
3.4.7.1	RGMII pins	3-29
3.4.7.2	GMII and TBI pins	3-29
3.4.8	Ports Tables	3-30
Chapter 4		
Reset, Clocking, and Initialization		
4.1	External Signals	4-1
4.1.1	Reset Signals	4-1
4.1.2	Clock Signals	4-3
4.2	Functional Description	4-4
4.2.1	Reset Operations	4-4
4.2.1.1	Reset Causes	4-4
4.2.1.2	Reset Actions	4-5
4.2.2	Power-On Reset Flow	4-6
4.2.3	Hard Reset Flow	4-7
4.2.4	Soft Reset Flow	4-8
4.3	Reset Configuration	4-9
4.3.1	Reset Configuration Signals	4-9

Contents

Paragraph Number	Title	Page Number
4.3.1.1	Reset Configuration Word Source	4-9
4.3.1.2	CLKIN Division	4-10
4.3.1.3	Selecting Reset Configuration Input Signals	4-10
4.3.2	Reset Configuration Words	4-11
4.3.2.1	Reset Configuration Word Low Register (RCWLR).....	4-12
4.3.2.1.1	System PLL VCO Division	4-13
4.3.2.1.2	System PLL Configuration	4-14
4.3.2.1.3	QUICC Engine PLL Multiplication Factor	4-16
4.3.2.2	Reset Configuration Word High Register (RCWHR).....	4-17
4.3.2.2.1	PCI Host/Agent Configuration	4-18
4.3.2.2.2	Boot Memory Space (BMS)	4-19
4.3.2.2.3	Boot Sequencer Configuration	4-19
4.3.2.2.4	Boot ROM Location	4-20
4.3.2.2.5	Secondary DDR IO Enable.....	4-22
4.3.2.2.6	e300 Core True Little-Endian.....	4-22
4.3.2.2.7	LALE Configuration.....	4-22
4.3.2.2.8	LDP Configuration	4-23
4.3.3	Loading the Reset Configuration Words	4-23
4.3.3.1	Loading from Local Bus EEPROM.....	4-23
4.3.3.2	Loading from I2C EEPROM	4-26
4.3.3.2.1	Using the Boot Sequencer Reset Configuration	4-26
4.3.3.2.2	EEPROM Calling Address	4-26
4.3.3.2.3	EEPROM Data Format in Reset Configuration Mode	4-26
4.3.3.2.4	Reset Configuration Load Fail	4-29
4.3.3.3	Default Reset Configuration Words.....	4-29
4.3.3.3.1	Examples for Hard-Coded Reset Configuration Words Usage	4-30
4.4	Clocking	4-31
4.4.1	Clocking in PCI Host Mode.....	4-32
4.4.1.1	PCI Clock Outputs (PCI_CLK_OUT[0:2])	4-32
4.4.2	Clocking In PCI Agent Mode	4-32
4.4.3	System Clock Domains.....	4-32
4.5	Memory Map/Register Definitions	4-34
4.5.1	Reset Configuration Register Descriptions.....	4-34
4.5.1.1	Reset Configuration Word Low Register (RCWLR).....	4-34
4.5.1.2	Reset Configuration Word High Register (RCWHR).....	4-34
4.5.1.3	Reset Status Register (RSR)	4-35
4.5.1.4	Reset Mode Register (RMR)	4-36
4.5.1.5	Reset Protection Register (RPR)	4-37
4.5.1.6	Reset Control Register (RCR)	4-37
4.5.1.7	Reset Control Enable Register (RCER).....	4-38
4.5.2	Clock Configuration Registers.....	4-38

Contents

Paragraph Number	Title	Page Number
4.5.2.1	System PLL Mode Register (SPMR)	4-39
4.5.2.2	Output Clock Control Register (OCCR).....	4-40
4.5.2.3	System Clock Control Register (SCCR).....	4-41
4.5.3	Clock Control DDR Registers	4-41
4.5.3.1	MCK Enable Register (MCKENR _n)	4-42

Chapter 5 System Configuration

5.1	Introduction.....	5-1
5.2	Local Memory Map Overview and Example	5-1
5.2.1	Address Translation and Mapping	5-3
5.2.2	Window into Configuration Space.....	5-4
5.2.3	Local Access Windows.....	5-4
5.2.3.1	Local Access Register Memory Map	5-5
5.2.4	Local Access Register Descriptions	5-6
5.2.4.1	Internal Memory Map Registers Base Address Register (IMMRBAR).....	5-6
5.2.4.1.1	Updating IMMRBAR	5-6
5.2.4.2	Alternate Configuration Base Address Register (ALTCBAR).....	5-7
5.2.4.3	LBC Local Access Window <i>n</i> Base Address Registers (LBLAWBAR0–LBLAWBAR3)	5-8
5.2.4.3.1	LBLAWBAR0[BASE_ADDR] Reset Value	5-8
5.2.4.4	LBC Local Access Window <i>n</i> Attributes Registers (LBLAWAR0–LBLAWAR3).....	5-9
5.2.4.4.1	LBLAWAR0[EN] and LBLAWAR0[SIZE] Reset Value	5-9
5.2.4.5	PCI Local Access Window <i>n</i> Base Address Register (PCILAWBAR0–PCILAWBAR1)	5-10
5.2.4.5.1	PCILAWBAR0[BASE_ADDR] Reset Value.....	5-10
5.2.4.6	PCI Local Access Window <i>n</i> Attributes Registers (PCILAWAR0–PCILAWAR1)	5-11
5.2.4.6.1	PCILAWAR0[EN] and PCILAWAR0[SIZE] Reset Value	5-11
5.2.4.7	DDR Local Access Window <i>n</i> Base Address Registers (DDRLAWBAR0–DDRLAWBAR1).....	5-12
5.2.4.7.1	DDRLAWBAR0[BASE_ADDR] Reset Value.....	5-12
5.2.4.8	DDR Local Access Window <i>n</i> Attributes Registers (DDRLAWAR0–DDRLAWAR1).....	5-13
5.2.4.8.1	DDRLAWAR0[EN] and DDRLAWAR0[SIZE] Reset Value.....	5-13
5.2.4.9	Secondary DDR Local Access Window <i>n</i> Base Address Registers (SDDRLAWBAR0–SDDRLAWBAR1)	5-14
5.2.4.10	Secondary DDR Local Access Window <i>n</i> Attributes Registers (SDDRLAWAR0–SDDRLAWAR1)	5-14

Contents

Paragraph Number	Title	Page Number
5.2.5	Precedence of Local Access Windows	5-15
5.2.6	Configuring Local Access Windows	5-15
5.2.7	Distinguishing Local Access Windows from Other Mapping Functions	5-15
5.2.8	Outbound Address Translation and Mapping Windows	5-16
5.2.9	Inbound Address Translation and Mapping Windows	5-16
5.2.9.1	PCI Inbound Windows.....	5-16
5.2.10	Internal Memory Map.....	5-16
5.2.11	Accessing Internal Memory from External Masters.....	5-17
5.3	QUICC Engine Secondary Bus Access Windows	5-17
5.3.1	QUICC Engine Secondary Bus Access Windows Register Memory Map.....	5-17
5.3.2	QUICC Engine Secondary Bus Access Windows Registers	5-18
5.3.2.1	Local Bus Memory Controller Start Address Register (LBMCSAR).....	5-18
5.3.2.2	Secondary DDR Memory Controller Start Address Register (SDMCSAR).....	5-19
5.3.2.3	Local Bus Memory Controller End Address Register (LBMCEAR).....	5-19
5.3.2.4	Secondary DDR Memory Controller End Address Register (SDMCEAR).....	5-20
5.3.2.5	Local Bus Memory Controller Attributes Register (LBMCAR).....	5-20
5.3.2.6	Secondary DDR Memory Controller Attributes Register (SDMCAR).....	5-21
5.3.3	QUICC Engine Secondary Bus Windows Operation Description.....	5-21
5.3.4	QUICC Engine Secondary Bus Windows Example	5-22
5.4	System Configuration	5-23
5.4.1	External Signal Description	5-23
5.4.1.1	PCI_MODE Signal	5-23
5.4.2	System Configuration Register Memory Map.....	5-24
5.4.3	System Configuration Registers	5-25
5.4.3.1	System General Purpose Register Low (SGPRL)	5-25
5.4.3.2	System General Purpose Register High (SGPRH)	5-25
5.4.3.3	System Part and Revision ID Register (SPRIDR).....	5-26
5.4.3.3.1	SPRIDR[PARTID] Coding.....	5-26
5.4.3.4	System Priority and Configuration Register (SPCR)	5-27
5.4.3.5	System I/O Configuration Register Low (SICRL)	5-28
5.4.3.6	System I/O Configuration Register High (SICRH).....	5-30
5.4.3.7	Debug Configuration	5-32
5.4.3.7.1	DDR Debug Configuration.....	5-32
5.4.3.7.2	Secondary DDR Debug Configuration.....	5-33
5.4.3.7.3	Local Bus Debug Configuration.....	5-33
5.4.3.8	DDR Control Driver Register (DDRCDR).....	5-33
5.4.3.9	DDR Debug Status Register (DDRDSR)	5-35
5.5	Software Watchdog Timer (WDT).....	5-36
5.5.1	WDT Overview.....	5-36
5.5.2	WDT Features.....	5-36
5.5.3	WDT Modes of Operation	5-37

Contents

Paragraph Number	Title	Page Number
5.5.4	WDT Memory Map/Register Definition	5-37
5.5.4.1	System Watchdog Control Register (SWCRR)	5-38
5.5.4.2	System Watchdog Count Register (SWCNR)	5-39
5.5.4.3	System Watchdog Service Register (SWSRR).....	5-39
5.5.5	Functional Description.....	5-40
5.5.5.1	Software Watchdog Timer Unit	5-40
5.5.5.2	Modes of Operation	5-41
5.5.6	Initialization/Application Information	5-42
5.5.6.1	WDT Programming Guidelines	5-42
5.6	Real Time Clock Module (RTC).....	5-42
5.6.1	RTC Overview	5-42
5.6.2	RTC Features	5-43
5.6.3	RTC Modes of Operation.....	5-43
5.6.4	RTC External Signal Description	5-43
5.6.5	RTC Memory Map/Register Definition.....	5-44
5.6.5.1	Real Time Counter Control Register (RTCNR)	5-44
5.6.5.2	Real Time Counter Load Register (RTLDR).....	5-45
5.6.5.3	Real Time Counter Prescale Register (RTPSR)	5-46
5.6.5.4	Real Time Counter Register (RTCTR)	5-46
5.6.5.5	Real Time Counter Event Register (RTEVR).....	5-47
5.6.5.6	Real Time Counter Alarm Register (RTALR)	5-47
5.6.6	Functional Description.....	5-48
5.6.6.1	Real Time Counter Unit.....	5-48
5.6.6.2	RTC Operational Modes	5-48
5.6.7	RTC Programming Guidelines.....	5-49
5.7	Periodic Interval Timer (PIT)	5-49
5.7.1	PIT Overview.....	5-49
5.7.2	PIT Features	5-50
5.7.3	PIT Modes of Operation	5-50
5.7.4	PIT External Signal Description	5-50
5.7.5	PIT Memory Map/Register Definition	5-51
5.7.5.1	Periodic Interval Timer Control Register (PTCNR)	5-51
5.7.5.2	Periodic Interval Timer Load Register (PTLDR)	5-52
5.7.5.3	Periodic Interval Timer Prescale Register (PTPSR)	5-53
5.7.5.4	Periodic Interval Timer Counter Register (PTCTR).....	5-53
5.7.5.5	Periodic Interval Timer Event Register (PTEVR)	5-54
5.7.6	Functional Description.....	5-54
5.7.6.1	Periodic Interval Timer Unit.....	5-54
5.7.6.2	PIT Operational Modes.....	5-55
5.7.7	PIT Programming Guidelines	5-55
5.8	General-Purpose Timers (GTMs).....	5-56

Contents

Paragraph Number	Title	Page Number
5.8.1	GTM Overview	5-56
5.8.2	GTM Features	5-57
5.8.3	GTM Modes of Operation.....	5-57
5.8.3.1	Cascaded Modes	5-57
5.8.3.2	Clock Source Modes.....	5-58
5.8.3.3	Reference Modes	5-58
5.8.3.4	Capture Modes.....	5-58
5.8.4	GTM External Signal Description	5-58
5.8.5	GTM Memory Map/Register Definition.....	5-60
5.8.5.1	Global Timers Configuration Registers (GTCFR n).....	5-61
5.8.5.2	Global Timers Mode Registers (GTMDR1–GTMDR4)	5-65
5.8.5.3	Global Timers Reference Registers (GTRFR1–GTRFR4)	5-66
5.8.5.4	Global Timers Capture Registers (GTCPR1–GTCPR4).....	5-66
5.8.5.5	Global Timers Counter Registers (GTCNR1–GTCNR4)	5-67
5.8.5.6	Global Timers Event Registers (GTEVR1–GTEVR4)	5-67
5.8.5.7	Global Timers Prescale Registers (GTPSR1–GTPSR4).....	5-68
5.8.6	Functional Description.....	5-69
5.8.6.1	General-Purpose Timer Units	5-69
5.8.6.2	Reference Modes	5-69
5.8.6.3	Capture Modes.....	5-69
5.8.6.4	Cascaded Modes	5-70
5.8.7	Initialization/Application Information.....	5-72
5.8.7.1	Programming Guidelines	5-72
5.8.7.1.1	GTM Registers.....	5-72
5.9	Power Management Control (PMC).....	5-72
5.9.1	External Signal Description	5-73
5.9.2	PMC Memory Map/Register Definition.....	5-73
5.9.2.1	Power Management Controller Configuration Register (PMCCR).....	5-73
5.9.2.2	Power Management Controller Event Register (PM CER).....	5-74
5.9.2.3	Power Management Controller Mask Register (PMCMR)	5-75
5.9.3	Functional Description.....	5-75
5.9.3.1	Dynamic Power Management.....	5-76
5.9.3.2	Shutting Down Unused Blocks.....	5-76
5.9.3.3	Software-Controlled Power-Down States.....	5-76
5.9.3.3.1	Entering Low Power States—Core-Only Mode	5-76
5.9.3.3.2	Entering Low Power States—Core and System Mode.....	5-76
5.9.3.4	Exiting Core and System Low Power States	5-77
5.9.3.4.1	Exiting Low Power States—Core-Only Mode	5-77
5.9.3.4.2	Exiting Low Power States—Core and System Mode.....	5-77
5.9.4	Initialization/Application Information.....	5-78
5.9.4.1	Core Disable in Low Power Mode	5-78

Contents

Paragraph Number	Title	Page Number
Chapter 6		
Arbiter and Bus Monitor		
6.1	Arbiter Overview	6-1
6.1.1	Coherent System Bus Overview	6-1
6.2	Arbiter Memory Map/Register Definition	6-2
6.2.1	Arbiter Configuration Register (ACR)	6-2
6.2.2	Arbiter Timers Register (ATR)	6-4
6.2.3	Arbiter Event Register (AER).....	6-5
6.2.4	Arbiter Interrupt Definition Register (AIDR).....	6-6
6.2.5	Arbiter Mask Register (AMR).....	6-7
6.2.6	Arbiter Event Attributes Register (AEATR).....	6-8
6.2.7	Arbiter Event Address Register (AEADR).....	6-9
6.2.8	Arbiter Event Response Register (AERR).....	6-10
6.3	Functional Description.....	6-11
6.3.1	Arbitration Policy	6-11
6.3.1.1	Address Bus Arbitration with <u>PRIORITY</u> [0:1]	6-11
6.3.1.2	Address Bus Arbitration with <u>REPEAT</u>	6-12
6.3.1.3	Address Bus Arbitration After <u>ARTRY</u>	6-13
6.3.1.4	Address Bus Parking.....	6-13
6.3.1.5	Data Bus Arbitration.....	6-13
6.3.2	Bus Error Detection	6-13
6.3.2.1	Address Time Out.....	6-14
6.3.2.2	Data Time Out	6-14
6.3.2.3	Transfer Error	6-14
6.3.2.4	Address Only Transaction Type.....	6-14
6.3.2.5	Reserved Transaction Type.....	6-15
6.3.2.6	Illegal (eciwx/ecowx) Transaction Type.....	6-15
6.4	Initialization/Applications Information	6-16
6.4.1	Initialization Sequence.....	6-16
6.4.2	Error Handling Sequence.....	6-16

Chapter 7 e300 Processor Core Overview

7.1	e300c1 Overview	7-1
7.1.1	e300c1 Features	7-3
7.1.2	Instruction Unit.....	7-6
7.1.2.1	Instruction Queue and Dispatch Unit	7-6
7.1.2.2	Branch Processing Unit (BPU).....	7-6
7.1.3	Independent Execution Units.....	7-7

Contents

Paragraph Number	Title	Page Number
7.1.3.1	Integer Unit (IU).....	7-7
7.1.3.2	Floating-Point Unit (FPU).....	7-7
7.1.3.3	Load/Store Unit (LSU).....	7-7
7.1.3.4	System Register Unit (SRU).....	7-8
7.1.4	Completion Unit.....	7-8
7.1.5	Memory Subsystem Support.....	7-8
7.1.5.1	Memory Management Units (MMUs).....	7-8
7.1.5.2	Cache Units.....	7-9
7.1.6	Bus Interface Unit (BIU).....	7-10
7.1.7	System Support Functions.....	7-11
7.1.7.1	Power Management.....	7-11
7.1.7.2	Time Base/Decrementer.....	7-11
7.1.7.3	JTAG Test and Debug Interface.....	7-12
7.1.7.4	Clock Multiplier.....	7-12
7.2	PowerPC Architecture Implementation.....	7-12
7.3	Implementation-Specific Information.....	7-12
7.3.1	Register Model.....	7-13
7.3.1.1	UISA Registers.....	7-15
7.3.1.1.1	General-Purpose Registers (GPRs).....	7-15
7.3.1.1.2	Floating-Point Registers (FPRs).....	7-15
7.3.1.1.3	Condition Register (CR).....	7-15
7.3.1.1.4	Floating-Point Status and Control Register (FPSCR).....	7-15
7.3.1.1.5	User-Level SPRs.....	7-15
7.3.1.2	VEA Registers.....	7-16
7.3.1.3	OEA Registers.....	7-16
7.3.1.3.1	Machine State Register (MSR).....	7-16
7.3.1.3.2	Segment Registers (SRs).....	7-18
7.3.1.3.3	Supervisor-Level SPRs.....	7-18
7.3.2	Instruction Set and Addressing Modes.....	7-25
7.3.2.1	PowerPC Instruction Set and Addressing Modes.....	7-25
7.3.2.2	Implementation-Specific Instruction Set.....	7-26
7.3.3	Cache Implementation.....	7-27
7.3.3.1	PowerPC Cache Characteristics.....	7-27
7.3.3.2	Implementation-Specific Cache Organization.....	7-27
7.3.3.3	Instruction and Data Cache Way-Locking.....	7-29
7.3.4	Interrupt Model.....	7-29
7.3.4.1	PowerPC Interrupt Model.....	7-29
7.3.4.2	Implementation-Specific Interrupt Model.....	7-30
7.3.5	Memory Management.....	7-33
7.3.5.1	PowerPC Memory Management.....	7-33
7.3.5.2	Implementation-Specific Memory Management.....	7-33

Contents

Paragraph Number	Title	Page Number
7.3.6	Instruction Timing	7-34
7.3.7	Core Interface	7-35
7.3.7.1	Memory Accesses.....	7-36
7.3.7.2	Signals.....	7-36
7.3.8	Debug Features	7-37
7.3.8.1	Breakpoint Signaling	7-37
7.4	Differences Between Cores.....	7-38

Chapter 8 Integrated Programmable Interrupt Controller (IPIC)

8.1	IPIC Introduction	8-1
8.2	IPIC Features	8-4
8.3	IPIC Modes of Operation	8-4
8.3.1	Core Enable Mode	8-4
8.3.2	Core Disable Mode.....	8-5
8.4	IPIC External Signal Description	8-5
8.4.1	IPIC External Signals Overview.....	8-5
8.4.2	IPIC Detailed Signal Descriptions.....	8-5
8.5	IPIC Memory Map/Register Definition.....	8-6
8.5.1	System Global Interrupt Configuration Register (SICFR)	8-8
8.5.2	System Global Interrupt Vector Register (SIVCR).....	8-9
8.5.3	System Internal Interrupt Pending Registers (SIPNR_H and SIPNR_L).....	8-11
8.5.4	System Internal Interrupt Group A Priority Register (SIPRR_A).....	8-13
8.5.5	System Internal Interrupt Group D Priority Register (SIPRR_D).....	8-14
8.5.6	System Internal Interrupt Mask Register (SIMSR_H and SIMSR_L)	8-15
8.5.7	System Internal Interrupt Control Register (SICNR)	8-16
8.5.8	System External Interrupt Pending Register (SEPNR).....	8-18
8.5.9	System Mixed Interrupt Group A Priority Register (SMPRR_A).....	8-18
8.5.10	System Mixed Interrupt Group B Priority Register (SMPRR_B)	8-19
8.5.11	System External Interrupt Mask Register (SEMSR)	8-20
8.5.12	System External Interrupt Control Register (SECNR).....	8-21
8.5.13	System Error Status Register (SERSR)	8-22
8.5.14	System Error Mask Register (SERMR).....	8-23
8.5.15	System Error Control Register (SERCR)	8-24
8.5.16	System Internal Interrupt Force Registers (SIFCR_H and SIFCR_L)	8-25
8.5.17	System External Interrupt Force Register (SEFCR).....	8-26
8.5.18	System Error Force Register (SERFR).....	8-26
8.5.19	System Critical Interrupt Vector Register (SCVCR)	8-27
8.5.20	System Management Interrupt Vector Register (SMVCR)	8-27
8.5.21	QUICC Engine Ports Interrupt Event Register (CEPIER)	8-28

Contents

Paragraph Number	Title	Page Number
8.5.22	QUICC Engine Ports Interrupt Mask Register (CEPIMR).....	8-29
8.5.23	QUICC Engine Ports Interrupt Control Register (CEPICR)	8-30
8.6	Functional Description.....	8-31
8.6.1	Interrupt Types	8-31
8.6.2	Interrupt Configuration	8-31
8.6.3	Internal Interrupts Group Relative Priority.....	8-33
8.6.4	Mixed Interrupts Group Relative Priority.....	8-33
8.6.5	Highest Priority Interrupt.....	8-34
8.6.6	Interrupt Source Priorities.....	8-34
8.6.7	Masking Interrupt Sources.....	8-37
8.6.8	Interrupt Vector Generation and Calculation	8-38
8.6.9	Machine Check Interrupts.....	8-38
8.6.10	QUICC Engine Ports Interrupts.....	8-39

Chapter 9 DDR Memory Controller

9.1	DDR Introduction	9-1
9.2	DDR Features	9-2
9.2.1	DDR Modes of Operation.....	9-3
9.3	DDR External Signal Descriptions	9-3
9.3.1	DDR Signals Overview	9-4
9.3.2	DDR Detailed Signal Descriptions	9-6
9.3.2.1	Memory Interface Signals.....	9-6
9.3.2.2	Clock Interface Signals.....	9-9
9.3.2.3	Debug Signals.....	9-10
9.4	DDR Memory Map/Register Definition	9-10
9.4.1	DDR Register Descriptions	9-11
9.4.1.1	Chip Select Memory Bounds (CS _n _BNDS).....	9-12
9.4.1.2	Chip Select Configuration (CS _n _CONFIG).....	9-12
9.4.1.3	DDR SDRAM Timing Configuration 3 (TIMING_CFG_3).....	9-14
9.4.1.4	DDR SDRAM Timing Configuration 0 (TIMING_CFG_0).....	9-15
9.4.1.5	DDR SDRAM Timing Configuration 1 (TIMING_CFG_1).....	9-17
9.4.1.6	DDR SDRAM Timing Configuration 2 (TIMING_CFG_2).....	9-19
9.4.1.7	DDR SDRAM Control Configuration (DDR_SDRAM_CFG).....	9-21
9.4.1.8	DDR SDRAM Control Configuration 2 (DDR_SDRAM_CFG_2).....	9-23
9.4.1.9	DDR SDRAM Mode Configuration (DDR_SDRAM_MODE).....	9-25
9.4.1.10	DDR SDRAM Mode 2 Configuration (DDR_SDRAM_MODE_2).....	9-26
9.4.1.11	DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL).....	9-26
9.4.1.12	DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)	9-29
9.4.1.13	DDR SDRAM Data Initialization (DDR_DATA_INIT)	9-29

Contents

Paragraph Number	Title	Page Number
9.4.1.14	DDR SDRAM Clock Control (DDR_SDRAM_CLK_CNTL)	9-30
9.4.1.15	DDR Initialization Address (DDR_INIT_ADDR)	9-30
9.4.1.16	DDR IP Block Revision 1 (DDR_IP_REV1)	9-31
9.4.1.17	DDR IP Block Revision 2 (DDR_IP_REV2)	9-31
9.4.1.18	Memory Data Path Error Injection Mask High (DATA_ERR_INJECT_HI)	9-32
9.4.1.19	Memory Data Path Error Injection Mask Low (DATA_ERR_INJECT_LO)	9-32
9.4.1.20	Memory Data Path Error Injection Mask ECC (ERR_INJECT)	9-33
9.4.1.21	Memory Data Path Read Capture High (CAPTURE_DATA_HI)	9-33
9.4.1.22	Memory Data Path Read Capture Low (CAPTURE_DATA_LO)	9-34
9.4.1.23	Memory Data Path Read Capture ECC (CAPTURE_ECC)	9-34
9.4.1.24	Memory Error Detect (ERR_DETECT)	9-34
9.4.1.25	Memory Error Disable (ERR_DISABLE)	9-35
9.4.1.26	Memory Error Interrupt Enable (ERR_INT_EN)	9-36
9.4.1.27	Memory Error Attributes Capture (CAPTURE_ATTRIBUTES)	9-37
9.4.1.28	Memory Error Address Capture (CAPTURE_ADDRESS)	9-38
9.4.1.29	Single-Bit ECC Memory Error Management (ERR_SBE)	9-39
9.5	Functional Description	9-39
9.5.1	DDR SDRAM Interface Operation	9-44
9.5.1.1	Supported DDR SDRAM Organizations	9-44
9.5.2	DDR SDRAM Address Multiplexing	9-46
9.5.3	JEDEC Standard DDR SDRAM Interface Commands	9-51
9.5.4	DDR SDRAM Interface Timing	9-53
9.5.4.1	Clock Distribution	9-56
9.5.5	DDR SDRAM Mode-Set Command Timing	9-57
9.5.6	DDR SDRAM Registered DIMM Mode	9-58
9.5.7	DDR SDRAM Write Timing Adjustments	9-58
9.5.8	DDR SDRAM Refresh	9-59
9.5.8.1	DDR SDRAM Refresh Timing	9-60
9.5.8.2	DDR SDRAM Refresh and Power-Saving Modes	9-61
9.5.8.2.1	Self-Refresh in Sleep Mode	9-62
9.5.9	DDR Data Beat Ordering	9-63
9.5.10	Page Mode and Logical Bank Retention	9-63
9.5.11	Error Checking and Correcting (ECC)	9-64
9.5.12	Error Management	9-66
9.6	Initialization/Application Information	9-67
9.6.1	Programming Differences between Memory Types	9-68
9.6.2	DDR SDRAM Initialization Sequence	9-71

Contents

Paragraph Number	Title	Page Number
Chapter 10		
Local Bus Controller		
10.1	LBC Introduction	10-1
10.1.1	LBC Features	10-2
10.1.2	Modes of Operation	10-3
10.1.2.1	LBC Bus Clock and Clock Ratios	10-3
10.1.2.2	Source ID Debug Mode	10-3
10.2	LBC External Signal Descriptions.....	10-4
10.3	LBC Memory Map/Register Definition.....	10-9
10.3.1	LBC Register Descriptions	10-10
10.3.1.1	Base Registers (BR0–BR7)	10-11
10.3.1.2	Option Registers (OR0–OR7).....	10-12
10.3.1.2.1	Address Mask	10-12
10.3.1.2.2	Option Registers (OR n)—GPCM Mode	10-13
10.3.1.2.3	Option Registers (OR n)—UPM Mode	10-16
10.3.1.2.4	Option Registers (OR n)—SDRAM Mode	10-17
10.3.1.3	UPM Memory Address Register (MAR).....	10-18
10.3.1.4	UPM Mode Registers (M n MR)	10-19
10.3.1.5	Memory Refresh Timer Prescaler Register (MRTPR)	10-21
10.3.1.6	UPM Data Register (MDR)	10-22
10.3.1.7	Local Bus SDRAM Machine Mode Register (LSDMR).....	10-22
10.3.1.8	UPM Refresh Timer (LURT).....	10-24
10.3.1.9	SDRAM Refresh Timer (LSRT).....	10-25
10.3.1.10	Transfer Error Status Register (LTESR).....	10-26
10.3.1.11	Transfer Error Check Disable Register (LTEDR).....	10-27
10.3.1.12	Transfer Error Interrupt Enable Register (LTEIR)	10-28
10.3.1.13	Transfer Error Attributes Register (LTEATR)	10-29
10.3.1.14	Transfer Error Address Register (LTEAR).....	10-30
10.3.1.15	Local Bus Configuration Register (LBCR)	10-30
10.3.1.16	Clock Ratio Register (LCRR).....	10-31
10.4	Functional Description.....	10-32
10.4.1	Basic Architecture.....	10-33
10.4.1.1	Address and Address Space Checking	10-33
10.4.1.2	External Address Latch Enable Signal (LALE)	10-33
10.4.1.3	Data Transfer Acknowledge (TA)	10-35
10.4.1.4	Data Buffer Control (LBCTL).....	10-35
10.4.1.5	Atomic Operation	10-35
10.4.1.6	Parity Generation and Checking (LDP).....	10-36
10.4.1.7	Bus Monitor	10-36
10.4.2	General-Purpose Chip-Select Machine (GPCM).....	10-36

Contents

Paragraph Number	Title	Page Number
10.4.2.1	Timing Configuration	10-38
10.4.2.2	Chip-Select Assertion Timing	10-41
10.4.2.2.1	Programmable Wait State Configuration	10-42
10.4.2.2.2	Chip-Select and Write Enable Negation Timing	10-42
10.4.2.2.3	Relaxed Timing	10-43
10.4.2.2.4	Output Enable (LOE) Timing	10-46
10.4.2.2.5	Extended Hold Time on Read Accesses	10-46
10.4.2.3	External Access Termination (LGTA)	10-47
10.4.2.4	Boot Chip-Select Operation.....	10-49
10.4.3	SDRAM Machine	10-49
10.4.3.1	Supported SDRAM Configurations.....	10-49
10.4.3.2	SDRAM Power-On Initialization	10-50
10.4.3.3	Intel PC133 and JEDEC-Standard SDRAM Interface Commands	10-51
10.4.3.4	Page Hit Checking	10-52
10.4.3.5	Page Management.....	10-52
10.4.3.6	SDRAM Address Multiplexing	10-52
10.4.3.7	SDRAM Device-Specific Parameters.....	10-53
10.4.3.7.1	Precharge-to-Activate Interval.....	10-54
10.4.3.7.2	Activate-to-Read/Write Interval	10-54
10.4.3.7.3	Column Address to First Data Out—CAS Latency.....	10-55
10.4.3.7.4	Last Data In to Precharge—Write Recovery	10-55
10.4.3.7.5	Refresh Recovery Interval (RFRC)	10-56
10.4.3.7.6	External Address and Command Buffers (BUFCMD).....	10-56
10.4.3.8	SDRAM Interface Timing	10-57
10.4.3.9	SDRAM Read/Write Transactions.....	10-59
10.4.3.10	SDRAM MODE-SET Command Timing.....	10-59
10.4.3.11	SDRAM Refresh.....	10-59
10.4.3.11.1	SDRAM Refresh Timing	10-60
10.4.4	User-Programmable Machines (UPMs).....	10-60
10.4.4.1	UPM Requests	10-61
10.4.4.1.1	Memory Access Requests.....	10-62
10.4.4.1.2	UPM Refresh Timer Requests	10-63
10.4.4.1.3	Software Requests—RUN Command	10-63
10.4.4.1.4	Exception Requests.....	10-64
10.4.4.2	Programming the UPMs	10-64
10.4.4.2.1	UPM Programming Example (Two Sequential Writes to the RAM Array)	10-65
10.4.4.2.2	UPM Programming Example (Two Sequential Reads from the RAM Array)	10-65
10.4.4.3	UPM Signal Timing	10-66
10.4.4.4	UPM RAM Array	10-67

Contents

Paragraph Number	Title	Page Number
10.4.4.4.1	UPM RAM Words	10-67
10.4.4.4.2	Chip-Select Signal Timing (CST _n)	10-70
10.4.4.4.3	Byte Select Signal Timing (BST _n)	10-71
10.4.4.4.4	General-Purpose Signals (GnTn, GOn)	10-72
10.4.4.4.5	Loop Control (LOOP)	10-72
10.4.4.4.6	Repeat Execution of Current RAM Word (REDO)	10-72
10.4.4.4.7	Address Multiplexing (AMX)	10-73
10.4.4.4.8	Data Valid and Data Sample Control (UTA)	10-73
10.4.4.4.9	LGPL[0:5] Signal Negation (LAST)	10-74
10.4.4.4.10	Wait Mechanism (WAEN)	10-74
10.4.4.5	Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge	10-75
10.4.4.6	Extended Hold Time on Read Accesses	10-75
10.4.4.7	Memory System Interface Example Using UPM	10-76
10.5	Initialization/Application Information	10-82
10.5.1	Interfacing to Peripherals in Multiplexed Address/Data Mode	10-82
10.5.1.1	Multiplexed Address/Data Bus and Unmultiplexed Address Signals	10-82
10.5.1.2	Peripheral Hierarchy on the Local Bus	10-83
10.5.1.3	Peripheral Hierarchy on the Local Bus for Very High Bus Speeds	10-83
10.5.1.4	GPCM Timings	10-84
10.5.2	Bus Turnaround	10-85
10.5.2.1	Address Phase after Previous Read	10-85
10.5.2.2	Read Data Phase after Address Phase	10-85
10.5.2.3	Read-Modify-Write Cycle for Parity Protected Memory Banks	10-86
10.5.2.4	UPM Cycles with Additional Address Phases	10-86
10.5.3	Interface to Different Port-Size Devices	10-86
10.5.4	Interfacing to SDRAM	10-88
10.5.4.1	Basic SDRAM Capabilities of the Local Bus	10-88
10.5.4.2	Maximum Amount of SDRAM Supported	10-89
10.5.4.3	SDRAM Machine Limitations	10-90
10.5.4.3.1	Analysis of Maximum Row Number Due to Bank Select Multiplexing	10-90
10.5.4.3.2	Bank Select Signals	10-90
10.5.4.3.3	128-Mbyte SDRAM	10-91
10.5.4.3.4	256-Mbyte SDRAM	10-93
10.5.4.3.5	512-Mbyte SDRAM	10-93
10.5.4.4	Parity Support for SDRAM	10-95
10.5.5	Interfacing to ZBT SRAM	10-96
10.5.5.1	Interfacing to MSC8122 DSI	10-97
10.5.5.2	DSI in Asynchronous SRAM-Like Mode	10-98
10.5.5.3	DSI in Synchronous Mode	10-100
10.5.5.3.1	Synchronous Single Write	10-103
10.5.5.3.2	Synchronous Single Read	10-104

Contents

Paragraph Number	Title	Page Number
10.5.5.3.3	Synchronous Burst Write.....	10-105
10.5.5.3.4	Synchronous Burst Read	10-106
10.5.5.4	Broadcast Accesses.....	10-106

Chapter 11 Sequencer

11.1	Sequencer Overview	11-1
11.1.1	Sequencer Features	11-2
11.2	Sequencer External Signal Description	11-2
11.3	Sequencer Memory Map/Register Definition.....	11-2
11.4	Sequencer Register Descriptions	11-3
11.4.1	PCI Outbound Translation Address Registers (POTAR _n).....	11-3
11.4.2	PCI Outbound Base Address Registers (POBAR _n)	11-3
11.4.3	PCI Outbound Comparison Mask Registers (POCMR _n)	11-4
11.4.4	Power Management Control Register (PMCR)	11-5
11.4.5	Discard Timer Control Register (DTCR)	11-6
11.5	Functional Description.....	11-6
11.5.1	Transaction Forwarding	11-6
11.5.1.1	Transactions from the Coherency System Bus (CSB) Port	11-7
11.5.1.2	Transactions from the PCI Port	11-7
11.5.1.3	Transactions from the DMA Port	11-7
11.5.2	PCI Outbound Address Translation.....	11-7
11.5.3	Transaction Ordering	11-8

Chapter 12 DMA/Messaging Unit

12.1	DMA Features.....	12-1
12.2	DMA External Signal Description.....	12-2
12.2.1	DMA Detailed Signal Descriptions	12-2
12.3	DMA Memory Map/Register Definition	12-3
12.4	DMA Register Descriptions.....	12-4
12.4.1	Outbound Message Interrupt Status Register (OMISR).....	12-4
12.4.2	Outbound Message Interrupt Mask Register (OMIMR).....	12-5
12.4.3	Inbound Message Registers (IMR0–IMR1)	12-6
12.4.4	Outbound Message Registers (OMR0–OMR1).....	12-6
12.4.5	Doorbell Registers	12-7
12.4.5.1	Outbound Doorbell Register (ODR).....	12-7
12.4.5.2	Inbound Doorbell Register (IDR).....	12-8
12.4.6	Inbound Message Interrupt Status Register (IMISR)	12-8

Contents

Paragraph Number	Title	Page Number
12.4.7	Inbound Message Interrupt Mask Register (IMIMR).....	12-9
12.4.8	DMA Registers	12-10
12.4.8.1	DMA Mode Register (DMAMR _n)	12-10
12.4.8.2	DMA Status Register (DMASR _n)	12-12
12.4.8.3	DMA Current Descriptor Address Register (DMACDAR _n)	12-13
12.4.8.4	DMA Source Address Register (DMASAR _n).....	12-14
12.4.8.5	DMA Destination Address Register (DMADAR _n).....	12-14
12.4.8.6	DMA Byte Count Register (DMABCR _n)	12-15
12.4.8.7	DMA Next Descriptor Address Register (DMANDAR _n).....	12-15
12.4.8.8	DMA General Status Register (DMAGSR).....	12-16
12.5	Functional Description.....	12-16
12.5.1	Message Unit	12-16
12.5.1.1	Messaging Registers (IMR0–IMR1, OMR0–OMR1)	12-16
12.5.1.2	Doorbell Registers (IDR and ODR)	12-17
12.5.2	DMA Controller.....	12-17
12.5.3	DMA Operation	12-17
12.5.3.1	External Control.....	12-18
12.5.3.2	DMA Coherency.....	12-19
12.5.3.3	Halt and Error Conditions.....	12-20
12.5.4	DMA Segment Descriptors.....	12-20
12.5.4.1	Descriptor in Big-Endian Mode.....	12-21
12.5.4.2	Descriptor in Little-Endian Mode.....	12-22
12.6	Initialization/Application Information	12-22
12.6.1	Initialization Steps in Direct Mode	12-22
12.6.2	Initialization Steps in Chaining Mode	12-22
12.6.3	Initialization Steps in Direct Mode with External Control	12-23
12.6.4	Initialization Steps in Chaining Mode with External Control	12-23

Chapter 13 PCI Bus Interface

13.1	PCI Introduction	13-2
13.1.1	PCI Features.....	13-3
13.1.2	PCI Modes of Operation	13-3
13.1.2.1	PCI Enable Configuration.....	13-3
13.1.2.2	Host/Agent Mode Configuration	13-4
13.1.2.3	PCI Clock Output Enable Configuration	13-4
13.1.2.4	PCI Arbiter Configuration	13-4
13.2	PCI External Signal Description.....	13-4
13.3	PCI Memory Map/Register Definitions.....	13-11
13.3.1	PCI Configuration Access Registers.....	13-13

Contents

Paragraph Number	Title	Page Number
13.3.1.1	PCI_CONFIG_ADDRESS	13-13
13.3.1.2	PCI_CONFIG_DATA	13-14
13.3.1.3	PCI Interrupt Acknowledge Register (PCI_INT_ACK).....	13-15
13.3.2	PCI Memory-Mapped Control and Status Registers	13-15
13.3.2.1	PCI Error Status Register (PCI_ESR)	13-15
13.3.2.2	PCI Error Capture Disable Register (PCI_ECDR).....	13-16
13.3.2.3	PCI Error Enable Register (PCI_EER).....	13-17
13.3.2.4	PCI Error Attributes Capture Register (PCI_EATCR)	13-18
13.3.2.5	PCI Error Address Capture Register (PCI_EACR)	13-19
13.3.2.6	PCI Error Extended Address Capture Register (PCI_EEACR)	13-20
13.3.2.7	PCI Error Data Low Capture Register (PCI_EDLCR).....	13-20
13.3.2.8	PCI General Control Register (PCI_GCR).....	13-20
13.3.2.9	PCI Error Control Register (PCI_ECR)	13-21
13.3.2.10	PCI General Status Register (PCI_GSR).....	13-22
13.3.2.11	PCI Inbound Translation Address Registers (PITAR _n).....	13-22
13.3.2.12	PCI Inbound Base Address Registers (PIBAR _n).....	13-23
13.3.2.13	PCI Inbound Extended Base Address Registers (PIEBAR _n)	13-24
13.3.2.14	PCI Inbound Window Attribute Registers (PIWAR _n).....	13-24
13.3.3	PCI Configuration Space Registers	13-25
13.3.3.1	Vendor ID Configuration Register.....	13-27
13.3.3.2	Device ID Configuration Register	13-27
13.3.3.3	PCI Command Configuration Register.....	13-28
13.3.3.4	PCI Status Configuration Register.....	13-29
13.3.3.5	Revision ID Configuration Register	13-30
13.3.3.6	Standard Programming Interface Configuration Register	13-30
13.3.3.7	Subclass Code Configuration Register	13-31
13.3.3.8	Base Class Code Configuration Register.....	13-31
13.3.3.9	Cache Line Size Configuration Register	13-32
13.3.3.10	Latency Timer Configuration Register	13-32
13.3.3.11	Header Type Configuration Register	13-33
13.3.3.12	BIST Control Configuration Register.....	13-33
13.3.3.13	PIMMR Base Address Configuration Register	13-33
13.3.3.14	GPL Base Address Register 0.....	13-34
13.3.3.15	GPL Base Address Registers 1–2	13-34
13.3.3.16	GPL Extended Base Address Registers 1–2	13-35
13.3.3.17	Subsystem Vendor ID Configuration Register	13-36
13.3.3.18	Subsystem Device ID Configuration Register.....	13-36
13.3.3.19	Capabilities Pointer Configuration Register.....	13-36
13.3.3.20	Interrupt Line Configuration Register	13-37
13.3.3.21	Interrupt Pin Configuration Register	13-37
13.3.3.22	Minimum Grant Configuration Register	13-37

Contents

Paragraph Number	Title	Page Number
13.3.3.23	Maximum Latency Configuration Register	13-38
13.3.3.24	PCI Function Configuration Register	13-38
13.3.3.25	PCI Arbiter Control Register (PCIACR)	13-39
13.3.3.26	Hot Swap Register Block.....	13-40
13.4	Functional Description.....	13-41
13.4.1	PCI Bus Arbitration	13-41
13.4.1.1	Bus Parking.....	13-41
13.4.1.2	Arbitration Algorithm.....	13-41
13.4.1.3	Broken Master Lock-Out.....	13-42
13.4.1.4	Master Latency Timer.....	13-43
13.4.2	Bus Commands	13-43
13.4.3	PCI Protocol Fundamentals	13-44
13.4.3.1	Basic Transfer Control.....	13-44
13.4.3.2	Addressing	13-44
13.4.3.3	Device Selection	13-45
13.4.3.4	Byte Enable Signals.....	13-45
13.4.3.5	Bus Driving and Turnaround	13-46
13.4.3.6	Bus Transactions.....	13-46
13.4.3.7	Read and Write Transactions	13-46
13.4.3.8	Transaction Termination	13-48
13.4.4	Other Bus Operations.....	13-50
13.4.4.1	Fast Back-to-Back Transactions	13-51
13.4.4.2	Dual Address Cycles.....	13-51
13.4.4.3	Data Streaming	13-51
13.4.4.4	Host Mode Configuration Access.....	13-52
13.4.4.5	Agent Mode Configuration Access	13-52
13.4.4.6	Special Cycle Command.....	13-53
13.4.4.7	Interrupt Acknowledge	13-53
13.4.5	Error Functions	13-54
13.4.5.1	Parity.....	13-54
13.4.5.2	Error Reporting.....	13-54
13.4.6	PCI Inbound Address Translation.....	13-56
13.4.7	CompactPCI Hot Swap Specification Support.....	13-57
13.4.8	Byte Ordering	13-57
13.4.8.1	Byte Order for Configuration Transactions	13-58
13.5	Initialization/Application Information	13-59
13.5.1	Initialization Sequence for Host Mode	13-59
13.5.2	Initialization Sequence for Agent Mode	13-59

Chapter 14 Security Engine (SEC) 2.4

Contents

Paragraph Number	Title	Page Number
14.1	SEC Overview	14-1
14.2	Architecture Overview	14-2
14.2.1	Data Packet Descriptors	14-4
14.2.2	Execution Units (EUs)	14-5
14.2.2.1	Public Key Execution Unit (PKEU)	14-5
14.2.2.1.1	Elliptic Curve Operations	14-5
14.2.2.1.2	Modular Exponentiation Operations	14-6
14.2.2.2	Data Encryption Standard Execution Unit (DEU)	14-6
14.2.2.3	ARC Four Execution Unit (AFEU)	14-7
14.2.2.4	Advanced Encryption Standard Execution Unit (AESU)	14-7
14.2.2.5	Message Digest Execution Unit (MDEU)	14-7
14.2.2.6	Random Number Generator (RNG)	14-7
14.2.3	Crypto-Channels	14-8
14.2.4	SEC Controller	14-9
14.2.4.1	Host-Controlled Access	14-9
14.2.4.2	Channel-Controlled Access	14-10
14.2.5	Bus Interface	14-10
14.3	Configuration of Internal Memory Space	14-10
14.4	Descriptor Overview	14-15
14.4.1	Descriptor Structure	14-15
14.4.2	Descriptor Format—Header Dword	14-16
14.4.2.1	Selecting Execution Units—EU_SEL0 and EU_SEL1	14-17
14.4.2.2	Selecting Descriptor Type—DESC_TYPE	14-18
14.4.3	Descriptor Format—Pointer Dwords	14-19
14.4.4	Link Table Format	14-21
14.4.4.1	Link Table Example	14-23
14.4.5	Descriptor Types	14-24
14.5	Execution Units	14-26
14.5.1	Public Key Execution Unit (PKEU)	14-26
14.5.1.1	PKEU Mode Register (PKEUMR)	14-26
14.5.1.2	PKEU Key Size Register (PKEUKSR)	14-28
14.5.1.3	PKEU AB Size Register (PKEUABS)	14-28
14.5.1.4	PKEU Data Size Register (PKEUDSR)	14-29
14.5.1.5	PKEU Reset Control Register (PKEURCR)	14-29
14.5.1.6	PKEU Status Register (PKEUSR)	14-30
14.5.1.7	PKEU Interrupt Status Register (PKEUISR)	14-31
14.5.1.8	PKEU Interrupt Control Register (PKEUICR)	14-32
14.5.1.9	PKEU EU-Go Register (PKEUEUG)	14-33
14.5.1.10	PKEU Parameter Memories	14-34
14.5.1.10.1	PKEU Parameter Memory A	14-34
14.5.1.10.2	PKEU Parameter Memory B	14-34

Contents

Paragraph Number	Title	Page Number
14.5.1.10.3	PKEU Parameter Memory E	14-34
14.5.1.10.4	PKEU Parameter Memory N	14-34
14.5.2	Data Encryption Standard Execution Unit (DEU).....	14-34
14.5.2.1	DEU Mode Register (DEUMR)	14-35
14.5.2.2	DEU Key Size Register (DEUKSR).....	14-36
14.5.2.3	DEU Data Size Register (DEUDSR).....	14-36
14.5.2.4	DEU Reset Control Register (DEURCR).....	14-37
14.5.2.5	DEU Status Register (DEUSR)	14-37
14.5.2.6	DEU Interrupt Status Register (DEUISR)	14-38
14.5.2.7	DEU Interrupt Control Register (DEUICR).....	14-40
14.5.2.8	DEU EU-Go Register (DEUEUG)	14-41
14.5.2.9	DEU IV Register (DEUIV)	14-42
14.5.2.10	DEU Key Registers (DEUK1–DEUK3).....	14-42
14.5.2.11	DEU FIFOs	14-42
14.5.3	ARC Four Execution Unit (AFEU)	14-42
14.5.3.1	AFEU Mode Register (AFEUMR).....	14-43
14.5.3.2	Host-Provided Context through Prevent Permute	14-43
14.5.3.2.1	Dump Context.....	14-43
14.5.3.3	AFEU Key Size Register (AFEUKSR)	14-44
14.5.3.4	AFEU Context/Data Size Register (AFEUDSR)	14-45
14.5.3.5	AFEU Reset Control Register (AFEURCR)	14-46
14.5.3.6	AFEU Status Register (AFEUSR).....	14-46
14.5.3.7	AFEU Interrupt Status Register (AFEUISR).....	14-47
14.5.3.8	AFEU Interrupt Control Register (AFEUICR).....	14-49
14.5.3.9	AFEU End of Message Register (AFEUEMR).....	14-50
14.5.3.10	AFEU Context	14-50
14.5.3.10.1	AFEU Context Memory	14-50
14.5.3.10.2	AFEU Context Memory Pointer Register	14-51
14.5.3.11	AFEU Key Registers (AFEUK0, AFEUK1)	14-51
14.5.3.11.1	AFEU FIFOs.....	14-51
14.5.4	Message Digest Execution Unit (MDEU)	14-51
14.5.4.1	MDEU Mode Register (MDEUMR)	14-51
14.5.4.2	Recommended Settings for MDEUMR.....	14-54
14.5.4.3	MDEU Key Size Register (MDEUKSR)	14-55
14.5.4.4	MDEU Data Size Register (MDEUDSR).....	14-56
14.5.4.5	MDEU Reset Control Register (MDEURCR).....	14-56
14.5.4.6	MDEU Status Register (MDEUSR)	14-57
14.5.4.7	MDEU Interrupt Status Register (MDEUISR).....	14-58
14.5.4.8	MDEU Interrupt Control Register (MDEUICR).....	14-59
14.5.4.9	MDEU ICV Size Register (MDEUICVSR)	14-60
14.5.4.10	MDEU EU-Go Register (MDEUEUG)	14-61

Contents

Paragraph Number	Title	Page Number
14.5.4.11	MDEU Context Registers	14-61
14.5.4.12	MDEU Key Registers	14-62
14.5.4.13	MDEU FIFO	14-63
14.5.5	Random Number Generator (RNG).....	14-63
14.5.5.1	RNG Mode Register (RNGMR).....	14-63
14.5.5.2	RNG Data Size Register (RNGDSR)	14-64
14.5.5.3	RNG Reset Control Register (RNGRCR)	14-65
14.5.5.4	RNG Status Register (RNGSR).....	14-65
14.5.5.5	RNG Interrupt Status Register (RNGISR).....	14-66
14.5.5.6	RNG Interrupt Control Register (RNGICR).....	14-67
14.5.5.7	RNG EU-Go Register (RNGEUG).....	14-68
14.5.5.8	RNG FIFO	14-68
14.5.6	Advanced Encryption Standard Execution Units (AESU)	14-68
14.5.6.1	AESU Mode Register (AESUMR).....	14-68
14.5.6.2	AESU Key Size Register (AESUKSR)	14-71
14.5.6.3	AESU Data Size Register (AESUDSR)	14-71
14.5.6.4	AESU Reset Control Register (AESURCR)	14-72
14.5.6.5	AESU Status Register (AESUSR).....	14-73
14.5.6.6	AESU Interrupt Status Register (AESUISR).....	14-74
14.5.6.7	AESU Interrupt Control Register (AESUICR).....	14-75
14.5.6.8	AESU End of Message Register (AESUEMR)	14-76
14.5.6.9	AESU Context Registers	14-77
14.5.6.9.1	Context for CBC Mode.....	14-78
14.5.6.9.2	Context for Counter Mode.....	14-78
14.5.6.9.3	Context for SRT Mode	14-78
14.5.6.9.4	Context for CCM Mode.....	14-79
14.5.6.9.5	AESU Key Registers	14-81
14.5.6.9.6	AESU FIFOs.....	14-81
14.6	Crypto-Channels	14-81
14.6.1	Crypto-Channel Registers.....	14-82
14.6.1.1	Crypto-Channel Configuration Register (CCCR)	14-82
14.6.1.2	Crypto-Channel Pointer Status Register (CCPSR).....	14-85
14.6.1.3	Crypto-Channel Current Descriptor Pointer Register (CDPR)	14-90
14.6.1.4	Fetch FIFO (FF).....	14-90
14.6.1.5	Descriptor Buffer (DB).....	14-91
14.6.2	Interrupts.....	14-92
14.6.2.1	Channel Done Interrupt	14-92
14.6.2.2	Channel Error Interrupt.....	14-92
14.6.2.3	Channel Reset	14-93
14.7	Controller	14-93
14.7.1	Controller Registers	14-93

Contents

Paragraph Number	Title	Page Number
14.7.2	EU Assignment Status Register (EUASR)	14-93
14.7.2.1	Interrupt Mask Register (IMR)	14-94
14.7.2.2	Interrupt Status Register (ISR)	14-96
14.7.2.3	Interrupt Clear Register (ICR)	14-96
14.7.2.4	ID Register	14-98
14.7.2.5	IP Block Revision Register	14-98
14.7.2.6	Master Control Register (MCR)	14-99
14.7.2.7	EU Access	14-100
14.7.2.8	Multiple EU Assignment	14-100
14.7.2.9	Multiple Channels	14-101
14.7.2.10	Priority Arbitration	14-101
14.7.2.11	Round-Robin Snapshot Arbiters	14-101
14.8	Bus Interface	14-102
14.8.1	Bus Access	14-102
14.8.1.1	Master Read	14-102
14.8.1.2	Master Write	14-103
14.8.1.2.1	Slave Access	14-103
14.8.2	Bus Arbitration Priority	14-103
14.8.3	Snooping by Caches	14-103
14.8.4	Interrupts	14-103
14.9	Power-Saving Mode	14-104

Chapter 15 I²C Interfaces

15.1	I ² C Introduction	15-1
15.1.1	I ² C Features	15-2
15.1.2	I ² C Modes of Operation	15-2
15.2	I ² C External Signal Descriptions	15-3
15.2.1	I ² C Signal Overview	15-3
15.2.2	I ² C Detailed Signal Descriptions	15-3
15.3	I ² C Memory Map/Register Definition	15-4
15.3.1	I ² C Register Descriptions	15-5
15.3.1.1	I ² C _n Address Register (I2CnADR)	15-5
15.3.1.2	I ² C _n Frequency Divider Register (I2CnFDR)	15-6
15.3.1.3	I ² C _n Control Register (I2CnCR)	15-7
15.3.1.4	I ² C _n Status Register (I2CnSR)	15-8
15.3.1.5	I ² C _n Data Register (I2CnDR)	15-9
15.3.1.6	Digital Filter Sampling Rate Register (I2CnDFSRR)	15-10
15.4	Functional Description	15-10
15.4.1	Transaction Protocol	15-10

Contents

Paragraph Number	Title	Page Number
15.4.1.1	START Condition	15-11
15.4.1.2	Slave Address Transmission	15-11
15.4.1.3	Repeated START Condition	15-12
15.4.1.4	STOP Condition.....	15-12
15.4.1.5	Protocol Implementation Details	15-12
15.4.1.5.1	Transaction Monitoring—Implementation Details.....	15-12
15.4.1.5.2	Control Transfer—Implementation Details	15-13
15.4.1.6	Address Compare—Implementation Details	15-14
15.4.2	Arbitration Procedure	15-14
15.4.2.1	Arbitration Control	15-14
15.4.3	Handshaking	15-15
15.4.4	Clock Control.....	15-15
15.4.4.1	Clock Synchronization.....	15-15
15.4.4.2	Input Synchronization and Digital Filter	15-15
15.4.4.2.1	Input Signal Synchronization	15-15
15.4.4.2.2	Filtering of SCL _n and SDA _n Lines	15-16
15.4.4.3	Clock Stretching	15-16
15.4.5	Boot Sequencer Mode.....	15-16
15.4.5.1	Using the Boot Sequencer for Reset Configuration	15-16
15.4.5.2	EEPROM Calling Address	15-17
15.4.5.3	EEPROM Data Format	15-17
15.4.5.4	Boot Sequencer Done Indication	15-20
15.5	Initialization/Application Information	15-20
15.5.1	Interrupt Service Routine Flowchart.....	15-20
15.5.2	Initialization Sequence.....	15-22
15.5.3	Generation of START	15-22
15.5.4	Post-Transfer Software Response	15-22
15.5.5	Generation of STOP.....	15-23
15.5.6	Generation of Repeated START	15-23
15.5.7	Generation of SCL _n When SDA _n is Negated	15-23
15.5.8	Slave Mode Interrupt Service Routine.....	15-23
15.5.8.1	Slave Transmitter and Received Acknowledge	15-24
15.5.8.2	Loss of Arbitration and Forcing of Slave Mode.....	15-24

Chapter 16 DUART

16.1	DUART Overview	16-1
16.1.1	DUART Features	16-2
16.1.2	DUART Modes of Operation.....	16-3
16.2	DUART External Signal Descriptions	16-3

Contents

Paragraph Number	Title	Page Number
16.2.1	DUART Signal Overview	16-3
16.2.2	DUART Detailed Signal Descriptions	16-3
16.3	DUART Memory Map/Register Definition	16-4
16.3.1	DUART Register Descriptions	16-6
16.3.1.1	Receiver Buffer Registers (URBR1 and URBR2, URBR3 and URBR4)	16-6
16.3.1.2	Transmitter Holding Registers (UTHR1 and UTHR2, UTHR3 and UTHR4)	16-6
16.3.1.3	Divisor Most and Least Significant Byte Registers (UDMB and UDLB)	16-7
16.3.1.4	Interrupt Enable Registers (UIER1 and UIER2, UIER3 and UIER4)	16-8
16.3.1.5	Interrupt ID Registers (UIIR1 and UIIR2, UIIR3 and UIIR4)	16-9
16.3.1.6	FIFO Control Registers (UFCR1 and UFCR2, UFCR3 and UFCR4)	16-10
16.3.1.7	Alternate Function Registers (UAFR1 and UAFR2, UAFR3 and UAFR4)	16-11
16.3.1.8	Line Control Registers (ULCR1 and ULCR2, ULCR3 and ULCR4)	16-12
16.3.1.9	MODEM Control Registers (UMCR1 and UMCR2, UMCR3 and UMCR4)	16-14
16.3.1.10	Line Status Registers (ULSR1 and ULSR2, ULSR3 and ULSR4)	16-15
16.3.1.11	MODEM Status Registers (UMSR1 and UMSR2, UMSR3 and UMSR4)	16-16
16.3.1.12	Scratch Registers (USCR1 and USCR2, USCR3 and USCR4)	16-17
16.3.1.13	DMA Status Registers (UDSR1 and UDSR2, UDSR3 and UDSR4)	16-17
16.4	Functional Description	16-18
16.4.1	Serial Interface	16-19
16.4.1.1	START Bit	16-19
16.4.1.2	Data Transfer	16-20
16.4.1.3	Parity Bit	16-20
16.4.1.4	STOP Bit	16-20
16.4.2	Baud-Rate Generator Logic	16-20
16.4.3	Local Loopback Mode	16-21
16.4.4	Errors	16-21
16.4.4.1	Framing Error	16-21
16.4.4.2	Parity Error	16-21
16.4.4.3	Overrun Error	16-21
16.4.5	FIFO Mode	16-21
16.4.5.1	FIFO Interrupts	16-22
16.4.5.2	DMA Mode Select	16-22
16.4.5.3	Interrupt Control Logic	16-22
16.5	DUART Initialization/Application Information	16-23

Chapter 17 JTAG/Testing Support

Contents

Paragraph Number	Title	Page Number
17.1	JTAG Overview	17-1
17.2	JTAG Signals	17-1
17.2.1	JTAG External Signal Descriptions	17-2
17.3	JTAG Registers and Scan Chains	17-3

Chapter 18 Delay Lock Loop (DLL)

18.1	DLL Introduction	18-1
18.2	DLL Overview	18-1
18.2.1	DLL Features	18-2
18.2.2	DLL Modes of Operation	18-2
18.2.3	DLL External Signals	18-2
18.3	DLL Initialization and Application Information	18-2
18.4	DLL Memory Map/Register Definition.....	18-3
18.4.1	DLL Override Register (DLLOVR)	18-4
18.4.2	DLL Status Register (DLLSR)	18-5
18.4.3	DLL Clock Register (DLLCK).....	18-5

Chapter 19 QUICC Engine Block on the MPC8360E

19.1	QUICC Engine Block	19-1
19.2	QUICC Engine Implementation Details for the MPC8360E.....	19-2
19.2.1	QUICC Engine System Interface.....	19-4
19.2.1.1	System Interface—Serial DMA.....	19-4
19.2.1.2	System Interface—Data Paths	19-4
19.2.1.3	System Interface—SDMA and Bus Error	19-5
19.2.1.4	System Interface—SDMA and Reset	19-5
19.2.1.5	System Interface—Arbitration over the System Bus.....	19-5
19.2.1.6	System Interface—Arbitration Over the Secondary Bus.....	19-6
19.2.2	QUICC Engine Block Control.....	19-6
19.2.2.1	QUICC Engine Block Control—CERCR[CIR]	19-6
19.2.2.2	QUICC Engine Block Control—QUICC Engine Microcode Revision	19-7
19.2.2.3	QUICC Engine Block Control—External Requests Device Specific Information	19-7
19.2.3	QUICC Engine Multiplexing and Timers.....	19-8
19.2.3.1	QUICC Engine Multiplexing and Timers—CMXUCR n	19-8
19.2.3.2	QUICC Engine Multiplexing and Timers—Global Timer Module (GTM).....	19-8
19.3	General-Purpose Timers (GTM).....	B-11
19.3.1	Overview.....	B-11

Contents

Paragraph Number	Title	Page Number
19.3.2	Features	B-12
19.3.3	Modes of Operation	B-13
19.3.3.1	Cascaded Modes	B-13
19.3.3.2	Clock Source Modes	B-13
19.3.3.3	Reference Modes	B-13
19.3.3.4	Capture Modes	B-14
19.3.4	External Signal Description	B-14
19.3.4.1	Overview	B-14
19.3.4.2	Detailed Signal Descriptions	B-15
19.3.5	Memory Map/Register Definition	B-15
19.3.5.1	Global Timers Configuration Registers (GTCFR _n)	B-16
19.3.5.2	Global Timers Mode Registers (GTMDR1–GTMDR4)	B-20
19.3.5.3	Global Timers Reference Registers (GTRFR1–GTRFR4)	B-21
19.3.5.4	Global Timers Capture Registers (GTCPR1–GTCPR4)	B-21
19.3.5.5	Global Timers Counter Registers (GTCNR1–GTCNR4)	B-22
19.3.5.6	Global Timers Event Registers (GTEVR1–GTEVR4)	B-22
19.3.5.7	Global Timers Prescale Registers (GTPSR1–GTPSR4)	B-23
19.3.6	Functional Description	B-24
19.3.6.1	General-Purpose Timer Units	B-24
19.3.6.2	Reference Modes	B-24
19.3.6.3	Capture Modes	B-24
19.3.6.4	Cascaded Modes	B-25
19.3.7	Initialization/Application Information	B-27
19.3.7.1	Programming Guidelines	B-27
19.3.7.1.1	GTM Registers	B-27

Figures

**Figure
Number**

Title

**Page
Number**

Figures

Figure Number	Title	Page Number
1-1	MPC8360E Block Diagram	1-2
1-2	MPC8360E Integrated e300c1 Core Block Diagram.....	1-9
1-3	QUICC Engine Block Architectural Block Diagram.....	1-11
1-4	Integrated Security Engine Functional Blocks.....	1-17
1-5	QUICC Engine Block in the MPC8360E Used as a SME Router	1-23
1-6	QUICC Engine Block in the MPC8360E Used in an DSLAM Line Card	1-24
1-7	Wireless Infrastructure—NodeB/BTS Network Interface Card	1-24
2-1	QUICC Engine 2.0 High-Level Memory Map	2-24
3-1	MPC8360E Signal Groupings.....	3-2
3-2	QUICC Engine Port Open-Drain Registers (CPODRA–CPODRG)	3-21
3-3	QUICC Engine Port Data Registers (CPDATA–CPDATG)	3-22
3-4	QUICC Engine Port Direction 1 Registers (CPDIR1A–CPDIR1G)	3-23
3-5	QUICC Engine Port Direction 2 Registers (CPDIR2A–CPDIR2G)	3-23
3-6	QUICC Engine Port Pin Assignment 1 Registers (CPPAR1A–CPPAR1G).....	3-24
3-7	QUICC Engine Port Pin Assignment 2 Registers (CPPAR2A–CPPAR2G).....	3-24
3-8	Communication Peripherals to QUICC Engine Mux Registers (CPCE1R–CPCE4R).....	3-24
3-9	QUICC Engine Port Output Hold Register 1 (CPOH1)	3-25
3-10	QUICC Engine Port Output Hold Register 2 (CPOH2)	3-26
3-11	Port Functional Block Diagram	3-27
3-12	Primary, Secondary and Third Option Programming	3-31
4-1	Power-On Reset Flow	4-7
4-2	Hard Reset Flow.....	4-8
4-3	Reset Configuration Word Low Register (RCWLR).....	4-12
4-4	Reset Configuration Word High Register (RCWHR).....	4-17
4-5	Loading Reset Configuration Words from Local Bus.....	4-25
4-6	Loading Reset Configuration Words from Local Bus (continued)	4-25
4-7	EEPROM Data Format for Reset Configuration Words Preload Command	4-27
4-8	EEPROM Contents	4-28
4-9	Clock Subsystem Block Diagram	4-31
4-10	Reset Status Register (RSR).....	4-35
4-11	Reset Mode Register (RMR).....	4-36
4-12	Reset Protection Register (RPR).....	4-37
4-13	Reset Control Register (RCR).....	4-37
4-14	Reset Control Enable Register (RCER)	4-38
4-15	System PLL Mode Register	4-39
4-16	Output Clock Control Register (OCCR).....	4-40
4-17	System Clock Control Register (SCCR).....	4-41
4-18	MCK Enable Register (MCKENR _n)	4-42
5-1	Local Memory Map Example	5-2
5-2	Internal Memory Map Registers' Base Address Register (IMMRBAR).....	5-7
5-3	Alternate Configuration Base Address Register (ALTCBAR)	5-7

Figures

Figure Number	Title	Page Number
5-4	LBC Local Access Window <i>n</i> Base Address Registers (LBLAWBAR0–LBLAWBAR3)	5-8
5-5	LBC Local Access Window <i>n</i> Attributes Registers (LBLAWAR0–LBLAWAR3)	5-9
5-6	PCI Local Access Window <i>n</i> Base Address Registers (PCILAWBAR0–PCILAWBAR1)	5-10
5-7	PCI Local Access Window <i>n</i> Attributes Registers (PCILAWAR0–PCILAWAR1)	5-11
5-8	DDR Local Access Window <i>n</i> Base Address Registers (DDRLAWBAR0–DDRLAWBAR1)	5-12
5-9	DDR Local Access Window <i>n</i> Attributes Registers (DDRLAWAR0–DDRLAWAR1)	5-13
5-10	Secondary DDR Local Access Window <i>n</i> Base Address Registers (SDDRLAWBAR0–SDDRLAWBAR1)	5-14
5-11	Secondary DDR Local Access Window <i>n</i> Attributes Registers (SDDRLAWAR0–SDDRLAWAR1)	5-14
5-12	Local Bus Memory Controller Start Address Register (LBMCSAR)	5-18
5-13	Secondary DDR Memory Controller Start Address Register (SDMCSAR)	5-19
5-14	Local Bus Memory Controller End Address Register (LBMCEAR)	5-19
5-15	Secondary DDR Memory Controller End Address Register (SDMCEAR)	5-20
5-16	Local Bus Memory Controller Attributes Register (LBMCAR)	5-20
5-17	Secondary DDR Memory Controller Attributes Register (SDMCAR)	5-21
5-18	Local Memory Map Example	5-22
5-19	System General Purpose Register Low (SGPRL)	5-25
5-20	System General Purpose Register High (SGPRH)	5-25
5-21	System Part and Revision ID Register (SPRIDR)	5-26
5-22	System Priority Configuration Register (SPCR)	5-27
5-23	System I/O Configuration Register Low (SICRL)	5-29
5-24	System I/O Configuration Register High (SICRH)	5-30
5-25	DDR Control Driver Register (DDRCDR)	5-34
5-26	DDR Debug Status Register (DDRDSR)	5-35
5-27	Software Watchdog Timer High-Level Block Diagram	5-36
5-28	System Watchdog Control Register (SWCRR)	5-38
5-29	System Watchdog Count Register (SWCNR)	5-39
5-30	System Watchdog Service Register (SWSRR)	5-39
5-31	Software Watchdog Timer Service State Diagram	5-40
5-32	Software Watchdog Timer Functional Block Diagram	5-41
5-33	Real Time Clock Module High Level Block Diagram	5-43
5-34	Real Time Counter Control Register (RTCNR)	5-44
5-35	Real Time Counter Load Register (RTLDR)	5-45
5-36	Real Time Counter Prescale Register (RTPSR)	5-46
5-37	Real Time Counter Register (RTCTR)	5-46
5-38	Real Time Counter Event Register (RTEVR)	5-47

Figures

Figure Number	Title	Page Number
5-39	Real Time Counter Alarm Register (RTALR)	5-47
5-40	Real Time Clock Module Functional Block Diagram	5-48
5-41	Periodic Interval Timer High Level Block Diagram.....	5-50
5-42	Periodic Interval Timer Control Register (PTCNR)	5-51
5-43	Periodic Interval Timer Load Register (PTLDR).....	5-52
5-44	Periodic Interval Timer Prescale Register (PTPSR)	5-53
5-45	Periodic Interval Timer Counter Register (PTCTR)	5-53
5-46	Periodic Interval Timer Event Register (PTEVR).....	5-54
5-47	Periodic Interval Timer Functional Block Diagram.....	5-55
5-48	Global Timers Block Diagram	5-56
5-49	Global Timers Configuration Register 1 (GTCFR1).....	5-62
5-50	Global Timers Configuration Register 2 (GTCFR2).....	5-63
5-51	Global Timers Mode Registers (GTMDR1–GTMDR4).....	5-65
5-52	Global Timers Reference Registers (GTRFR1–GTRFR4).....	5-66
5-53	Global Timers Capture Registers (GTCPR1–GTCPR4)	5-66
5-54	Global Timers Counter Registers (GTCNR1–GTCNR4).....	5-67
5-55	Global Timers Event Registers (GTEVR1–GTEVR4).....	5-67
5-56	Global Timers Prescale Registers (GTPSR1–GTPSR4).....	5-68
5-57	Timers Non-Cascaded Mode Block Diagram	5-70
5-58	Timer Pair-Cascaded Mode Block Diagram	5-71
5-59	Timers Super-Cascaded Mode Block Diagram.....	5-71
5-60	Power Management Controller Configuration Register	5-73
5-61	Power Management Controller Event Register	5-74
5-62	Power Management Controller Mask Register.....	5-75
6-1	Arbiter Configuration Register (ACR)	6-2
6-2	Arbiter Timers Register (ATR)	6-4
6-3	Arbiter Event Register (AER).....	6-5
6-4	Arbiter Interrupt Definition Register (AIDR).....	6-6
6-5	Arbiter Mask Register (AMR)	6-7
6-6	Arbiter Event Attributes Register (AEATR).....	6-8
6-7	Arbiter Event Address Register (AEADR).....	6-9
6-8	Arbiter Event Response Register (AERR).....	6-10
6-9	Address Bus Arbitration.....	6-11
6-10	An Example of Priority-Based Arbitration Algorithm	6-12
7-1	e300c1 Core Block Diagram.....	7-2
7-2	e300 Programming Model—Registers.....	7-14
7-3	Machine State Register (MSR)	7-16
7-4	e300c1 Data Cache Organization.....	7-28
7-5	Core Interface.....	7-36
8-1	Interrupt Sources Block Diagram	8-3
8-2	System Global Interrupt Configuration Register (SICFR)	8-8

Figures

Figure Number	Title	Page Number
8-3	System Global Interrupt Vector Register (SIVCR).....	8-9
8-4	System Internal Interrupt Pending Register (SIPNR_H).....	8-11
8-5	System Internal Interrupt Pending Register (SIPNR_L).....	8-12
8-6	System Internal Interrupt Group A Priority Register (SIPRR_A).....	8-14
8-7	System Internal Interrupt Group D Priority Register (SIPRR_D).....	8-14
8-8	System Internal Interrupt Mask Register (SIMSR_H).....	8-15
8-9	System Internal Interrupt Mask Register (SIMSR_L).....	8-16
8-10	System Internal Interrupt Control Register (SICNR).....	8-17
8-11	System External Interrupt Pending Register (SEPNR).....	8-18
8-12	System Mixed Interrupt Group A Priority Register (SMPRR_A).....	8-18
8-13	System Mixed Interrupt Group B Priority Register (SMPRR_B).....	8-19
8-14	System External Interrupt Mask Register (SEMSR).....	8-20
8-15	System External Interrupt Control Register (SECNR).....	8-21
8-16	System Error Status Register (SERSR).....	8-22
8-17	System Error Mask Register (SERMR).....	8-24
8-18	System Error Control Register (SERCR).....	8-24
8-19	System Internal Interrupt Force Register (SIFCR_H).....	8-25
8-20	System Internal Interrupt Force Register (SIFCR_L).....	8-25
8-21	System External Interrupt Force Register (SEFCR).....	8-26
8-22	System Error Status Register (SERFR).....	8-26
8-23	System Critical Interrupt Vector Register (SCVCR).....	8-27
8-24	System Management Interrupt Vector Register (SMVCR).....	8-28
8-25	QUICC Engine Ports Interrupt Event Register (CEPIER).....	8-29
8-26	QUICC Engine Ports Interrupt Mask Register (CEPIMR).....	8-30
8-27	QUICC Engine Ports Interrupt Control Register (CEPICR).....	8-30
8-28	Interrupt Structure.....	8-32
8-29	DDR Interrupt Request Masking.....	8-38
9-1	DDR Memory Controller Simplified Block Diagram.....	9-2
9-2	Chip Select Bounds Registers (CS _n _BNDS).....	9-12
9-3	Chip Select Configuration Register (CS _n _CONFIG).....	9-13
9-4	DDR SDRAM Timing Configuration 3 (TIMING_CFG_3).....	9-14
9-5	DDR SDRAM Timing Configuration 0 (TIMING_CFG_0).....	9-15
9-6	DDR SDRAM Timing Configuration 1 (TIMING_CFG_1).....	9-17
9-7	DDR SDRAM Timing Configuration 2 Register (TIMING_CFG_2).....	9-19
9-8	DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG).....	9-21
9-9	DDR SDRAM Control Configuration Register 2 (DDR_SDRAM_CFG_2).....	9-23
9-10	DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE).....	9-25
9-11	DDR SDRAM Mode 2 Configuration Register (DDR_SDRAM_MODE_2).....	9-26
9-12	DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL).....	9-26
9-13	DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL).....	9-29
9-14	DDR SDRAM Data Initialization Configuration Register (DDR_DATA_INIT).....	9-29

Figures

Figure Number	Title	Page Number
9-15	DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL)	9-30
9-16	DDR Initialization Address Configuration Register (DDR_INIT_ADDR)	9-30
9-17	DDR IP Block Revision 1 (DDR_IP_REV1)	9-31
9-18	DDR IP Block Revision 2 (DDR_IP_REV2)	9-31
9-19	Memory Data Path Error Injection Mask High Register (DATA_ERR_INJECT_HI)	9-32
9-20	Memory Data Path Error Injection Mask Low Register (DATA_ERR_INJECT_LO).....	9-32
9-21	Memory Data Path Error Injection Mask ECC Register (ERR_INJECT).....	9-33
9-22	Memory Data Path Read Capture High Register (CAPTURE_DATA_HI).....	9-33
9-23	Memory Data Path Read Capture Low Register (CAPTURE_DATA_LO)	9-34
9-24	Memory Data Path Read Capture ECC Register (CAPTURE_ECC).....	9-34
9-25	Memory Error Detect Register (ERR_DETECT).....	9-35
9-26	Memory Error Disable Register (ERR_DISABLE).....	9-35
9-27	Memory Error Interrupt Enable Register (ERR_INT_EN).....	9-36
9-28	Memory Error Attributes Capture Register (CAPTURE_ATTRIBUTES).....	9-37
9-29	Memory Error Address Capture Register (CAPTURE_ADDRESS)	9-38
9-30	Single-Bit ECC Memory Error Management Register (ERR_SBE)	9-39
9-31	DDR Memory Controller Block Diagram	9-40
9-32	Typical Dual Data Rate SDRAM Internal Organization.....	9-41
9-33	Typical DDR SDRAM Interface Signals	9-41
9-34	Example 256-Mbyte DDR SDRAM Configuration With ECC	9-43
9-35	DDR SDRAM Burst Read Timing—ACTTORW = 3, \overline{MCAS} Latency = 2	9-55
9-36	DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTOR	9-55
9-37	DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTORW = 3.....	9-56
9-38	DDR SDRAM Clock Distribution Example for x8 DDR SDRAMs	9-57
9-39	DDR SDRAM Mode-Set Command Timing	9-57
9-40	Registered DDR SDRAM DIMM Burst Write Timing	9-58
9-41	Write Timing Adjustments Example for Write Latency = 1	9-59
9-42	DDR SDRAM Bank Staggered Auto Refresh Timing.....	9-60
9-43	DDR SDRAM Power-Down Mode	9-61
9-44	DDR SDRAM Self-Refresh Entry Timing	9-62
9-45	DDR SDRAM Self-Refresh Exit Timing	9-62
10-1	Local Bus Controller Block Diagram	10-1
10-2	Base Registers (BR _n)	10-11
10-3	Option Registers (OR _n) in GPCM Mode.....	10-13
10-4	Option Registers (OR _n) in UPM Mode	10-16
10-5	Option Registers (OR _n) in SDRAM Mode.....	10-17
10-6	UPM Memory Address Register (MAR)	10-18
10-7	UPM Mode Registers (M _n MR)	10-19
10-8	Memory Refresh Timer Prescaler Register (MRTPR).....	10-21
10-9	UPM Data Register (MDR)	10-22

Figures

Figure Number	Title	Page Number
10-10	Local Bus SDRAM Machine Mode Register (LSDMR)	10-22
10-11	UPM Refresh Timer (LURT)	10-24
10-12	LSRT SDRAM Refresh Timer (LSRT).....	10-25
10-13	Transfer Error Status Register (LTESR)	10-26
10-14	Transfer Error Check Disable Register (LTEDR).....	10-27
10-15	Transfer Error Interrupt Enable Register (LTEIR).....	10-28
10-16	Transfer Error Attributes Register (LTEATR)	10-29
10-17	Transfer Error Address Register (LTEAR)	10-30
10-18	Local Bus Configuration Register.....	10-30
10-19	Clock Ratio Register (LCRR)	10-31
10-20	Basic Operation of Memory Controllers in the LBC	10-33
10-21	Example of 8-Bit GPCM Writing 32 Bytes to Address 0x5420	10-34
10-22	Basic LBC Bus Cycle with LALE, TA, and $\overline{\text{LCS}}_n$	10-35
10-23	Local Bus to GPCM Device Interface	10-37
10-24	GPCM Basic Read Timing (XACS = 0, ACS = 1x, TRLX = 0, CLKDIV = 4,8)	10-37
10-25	GPCM Basic Write Timing (XACS = 0, ACS = 00, CSNT = 1, SCY = 1, TRLX = 0, CLKDIV = 4, 8)	10-42
10-26	GPCM Relaxed Timing Read (XACS = 0, ACS = 1x, SCY = 1, CSNT = 0, TRLX = 1, CLKDIV = 4, 8)	10-43
10-27	GPCM Relaxed Timing Back-to-Back Writes (XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1, CLKDIV = 4, 8).....	10-44
10-28	GPCM Relaxed Timing Write (XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TRLX = 1, CLKDIV = 4, 8)	10-45
10-29	GPCM Relaxed Timing Write (XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TRLX = 1, CLKDIV = 4, 8)	10-45
10-30	GPCM Read Followed by Read (TRLX = 0, EHTR = 0, Fastest Timing).....	10-46
10-31	GPCM Read Followed by Write (TRLX = 0, EHTR = 1, One-Cycle Extended Hold Time on Reads)	10-47
10-32	External Termination of GPCM Access (PLL Enabled Mode)	10-48
10-33	External Termination of GPCM Access (PLL Bypass Mode).....	10-48
10-34	Connection to a 32-Bit SDRAM with 12 Address Lines.....	10-50
10-35	SDRAM Address Multiplexing	10-53
10-36	PRETOACT = 2 (2 Clock Cycles).....	10-54
10-37	ACTTORW = 2 (2 Clock Cycles).....	10-54
10-38	CL = 2 (2 Clock Cycles)	10-55
10-39	WRC = 2 (2 Clock Cycles)	10-55
10-40	RFRC = 4 (6 Clock Cycles)	10-56
10-41	BUFCMD = 1, LCRR[BUFCMDC] = 2.....	10-56
10-42	SDRAM Single-Beat Read, Page Closed, CL = 3	10-57
10-43	SDRAM Single-Beat Read, Page Hit, CL = 3	10-57

Figures

Figure Number	Title	Page Number
10-44	SDRAM Two-Beat Burst Read, Page Closed, CL = 3.....	10-57
10-45	SDRAM Four-Beat Burst Read, Page Miss, CL = 3.....	10-57
10-46	SDRAM Single-Beat Write, Page Hit.....	10-58
10-47	SDRAM Three-Beat Write, Page Closed.....	10-58
10-48	SDRAM Read-after-Read Pipelined, Page Hit, CL = 3.....	10-58
10-49	SDRAM Write-after-Write Pipelined, Page Hit.....	10-58
10-50	SDRAM Read-after-Write Pipelined, Page Hit	10-59
10-51	SDRAM MODE-SET Command.....	10-59
10-52	SDRAM Bank-Staggered Auto Refresh Timing.....	10-60
10-53	User-Programmable Machine Functional Block Diagram.....	10-61
10-54	RAM Array Indexing	10-62
10-55	Memory Refresh Timer Request Block Diagram	10-63
10-56	UPM Clock Scheme for LCRR[CLKDIV] = 2.....	10-66
10-57	UPM Clock Scheme for LCRR[CLKDIV] = 4 or 8	10-66
10-58	UPM RAM Array and Signal Generation.....	10-67
10-59	UPM RAM Word Field Descriptions.....	10-67
10-60	LCSn Signal Selection	10-71
10-61	LBS Signal Selection	10-71
10-62	UPM Read Access Data Sampling.....	10-74
10-63	Effect of LUPWAIT Signal.....	10-75
10-64	Single-Beat Read Access to FPM DRAM.....	10-77
10-65	Single-Beat Write Access to FPM DRAM	10-78
10-66	Burst Read Access to FPM DRAM Using LOOP (Two Beats Shown).....	10-79
10-67	Refresh Cycle (CBR) to FPM DRAM	10-80
10-68	Exception Cycle	10-81
10-69	Multiplexed Address/Data Bus	10-82
10-70	Local Bus Peripheral Hierarchy.....	10-83
10-71	Local Bus Peripheral Hierarchy for Very High Bus Speeds	10-84
10-72	GPCM Address Timings	10-84
10-73	GPCM Data Timings.....	10-85
10-74	Interface to Different Port-Size Devices	10-87
10-75	128-Mbyte SDRAM Diagram.....	10-91
10-76	Parity Support for SDRAM.....	10-95
10-77	Interface to ZBT SRAM	10-96
10-78	Interface to MSC8122 DSI in Asynchronous Mode	10-98
10-79	Asynchronous Write to MSC8122 DSI.....	10-99
10-80	Asynchronous Read from MSC8122 DSI.....	10-100
10-81	Interface to MSC8122 DSI in Synchronous Mode	10-101
10-82	UPM Synchronization Cycle	10-102
10-83	Synchronous Single Write to MSC8122 DSI.....	10-103
10-84	Synchronous Single Read from MSC8122 DSI.....	10-104

Figures

Figure Number	Title	Page Number
10-85	Synchronous Burst Write to MSC8122 DSI	10-105
10-86	Synchronous Burst Read from MSC8122 DSI	10-106
11-1	I/O Sequencer Block Diagram	11-1
11-2	PCI Outbound Translation Address Registers (POTAR _n)	11-3
11-3	PCI Outbound Base Address Registers (POBAR _n)	11-3
11-4	PCI Outbound Comparison Mask Registers (POCMR _n)	11-4
11-5	Power Management Control Register (PMCR)	11-5
11-6	Discard Timer Control Register (DTCR)	11-6
11-7	Outbound PCI Memory Address Translation	11-8
12-1	DMA/Messaging Unit Block Diagram	12-1
12-2	Outbound Message Interrupt Status Register (OMISR)	12-4
12-3	Outbound Message Interrupt Mask Register (OMIMR)	12-5
12-4	Inbound Message Registers (IMR0, IMR1)	12-6
12-5	Outbound Message Registers (OMR0–OMR1)	12-6
12-6	Outbound Doorbell Register (ODR)	12-7
12-7	Inbound Doorbell Register (IDR)	12-8
12-8	Inbound Message Interrupt Status Register (IMISR)	12-8
12-9	Inbound Message Interrupt Mask Register (IMIMR)	12-9
12-10	DMA Mode Register (DMAMR _n)	12-10
12-11	DMA Status Register (DMASR _n)	12-12
12-12	DMA Current Descriptor Address Register (DMACDAR _n)	12-13
12-13	DMA Source Address Register (DMASAR _n)	12-14
12-14	DMA Destination Address Register (DMADAR _n)	12-14
12-15	DMA Byte Count Register (DMABCR _n)	12-15
12-16	DMA Next Descriptor Address Register (DMANDAR _n)	12-15
12-17	DMA General Status Register (DMAGSR)	12-16
12-18	DMA Controller Block Diagram	12-17
12-19	DMA Chain of Segment Descriptors	12-21
13-1	PCI Controller Block Diagram	13-2
13-2	PCI Interface External Signals	13-5
13-3	PCI_CONFIG_ADDRESS Register	13-13
13-4	PCI_CONFIG_DATA	13-15
13-5	PCI Error Status Register (PCI_ESR)	13-15
13-6	PCI Error Capture Disable Register (PCI_ECDR)	13-16
13-7	PCI Error Enable Register (PCI_EER)	13-17
13-8	PCI Error Attributes Capture Register (PCI_EATCR)	13-18
13-9	PCI Error Address Capture Register (PCI_EACR)	13-19
13-10	PCI Error Extended Address Capture Register (PCI_EEACR)	13-20
13-11	PCI Error Data Low Capture Register (PCI_EDLCR)	13-20
13-12	PCI General Control Register (PCI_GCR)	13-21
13-13	PCI Error Control Register (PCI_ECR)	13-21

Figures

Figure Number	Title	Page Number
13-14	PCI General Status Register (PCI_GSR)	13-22
13-15	PCI Inbound Translation Address Registers (PITARn)	13-23
13-16	PCI Inbound Base Address Registers (PIBARn)	13-23
13-17	PCI Inbound Extended Base Address Registers (PIEBARn)	13-24
13-18	PCI Inbound Window Attribute Registers (PIWARn)	13-24
13-19	Vendor ID Configuration Register	13-27
13-20	Device ID Configuration Register	13-27
13-21	PCI Command Configuration Register	13-28
13-22	PCI Status Configuration Register	13-29
13-23	Revision ID Configuration Register	13-30
13-24	Standard Programming Interface Configuration Register	13-30
13-25	Subclass Code Configuration Register	13-31
13-26	Base Class Code Configuration Register	13-31
13-27	Cache Line Size Configuration Register	13-32
13-28	Latency Timer Configuration Register	13-32
13-29	Header Type Configuration Register	13-33
13-30	BIST Control Configuration Register	13-33
13-31	PIMMR Base Address Configuration Register	13-33
13-32	GPL Base Address Register 0	13-34
13-33	GPL Base Address Registers 1–2	13-35
13-34	GPL Extended Base Address Registers 1–2	13-35
13-35	Subsystem Vendor ID Configuration Register	13-36
13-36	Subsystem Device ID Configuration Register	13-36
13-37	Capabilities Pointer Configuration Register	13-37
13-38	Interrupt Line Configuration Register	13-37
13-39	Interrupt Pin Register	13-37
13-40	Minimum Grant Configuration Register	13-38
13-41	Maximum Latency Configuration Register	13-38
13-42	PCI Function Configuration Register	13-38
13-43	PCI Arbiter Control Register (PCIACR)	13-39
13-44	Hot Swap Register Block	13-40
13-45	PCI Arbitration Example	13-42
13-46	Single Beat Read Example	13-47
13-47	Burst Read Example	13-47
13-48	Single Beat Write Example	13-48
13-49	Burst Write Example	13-48
13-50	Target-Initiated Terminations	13-49
13-51	PCI Parity Operation	13-55
13-52	Inbound PCI Memory Address Translation	13-56
13-53	Address Invariant Byte Ordering—4 bytes Outbound	13-57
13-54	Address Invariant Byte Ordering—4 bytes Inbound	13-58

Figures

Figure Number	Title	Page Number
13-55	Address Invariant Byte Ordering—8 bytes Outbound.....	13-58
13-56	Address Invariant Byte Ordering—2 bytes Inbound	13-58
13-57	CFG_DATA Byte Ordering.....	13-59
14-1	SEC Connected to MPC8360E Internal Bus.....	14-3
14-2	SEC Functional Modules	14-3
14-4	Header Dword	14-16
14-3	Descriptor Format	14-16
14-5	Pointer Dword	14-20
14-6	Link Table Entry Format	14-21
14-7	Descriptors, Link Tables, and Data Parcels	14-23
14-8	PKEU Mode Register.....	14-26
14-9	PKEU Key Size Register	14-28
14-10	PKEU AB Size Register	14-29
14-11	PKEU Data Size Register (PKEUDSR).....	14-29
14-12	PKEU Reset Control Register (PKEURCR).....	14-29
14-13	PKEU Status Register (PKEUSR)	14-30
14-14	PKEU Interrupt Status Register (PKEUISR).....	14-31
14-15	PKEU Interrupt Control Register (PKEUICR).....	14-32
14-16	PKEU EU-Go Register (PKEUEUG)	14-34
14-17	DEU Mode Register (DEUMR).....	14-35
14-18	DEU Key Size Register (DEUKSR).....	14-36
14-19	DEU Data Size Register (DEUDSR).....	14-37
14-20	DEU Reset Control Register (DEURCR)	14-37
14-21	DEU Status Register (DEUSR).....	14-38
14-22	DEU Interrupt Status Register (DEUISR)	14-38
14-23	DEU Interrupt Control Register (DEUICR)	14-40
14-24	DEU EU-Go Register (DEUEUG)	14-42
14-25	AFEU Mode Register (AFEUMR)	14-43
14-26	AFEU Key Size Register (AFEUKSR)	14-44
14-27	AFEU Data Size Register	14-45
14-28	AFEU Reset Control Register (AFEURCR).....	14-46
14-29	AFEU Status Register	14-46
14-30	AFEU Interrupt Status Register (AFEUISR).....	14-47
14-31	AFEU Interrupt Control Register (AFEUICR).....	14-49
14-32	AFEU End of Message Register (AFEUEMR)	14-50
14-33	MDEU Mode Register (MDEUMR) in ‘Old’ Configuration	14-52
14-34	MDEU Mode Register (MDEUMR) in ‘New’ Configuration.....	14-53
14-35	MDEU Key Size Register (MDEUKSR).....	14-55
14-36	MDEU Data Size Register (MDEUDSR).....	14-56
14-37	MDEU Reset Control Register (MDEURCR).....	14-56
14-38	MDEU Status Register.....	14-57

Figures

Figure Number	Title	Page Number
14-39	MDEU Interrupt Status Register (MDEUISR)	14-58
14-40	MDEU Interrupt Control Register (MDEUICR)	14-59
14-41	MDEU ICV Size Register	14-61
14-42	MDEU EU-Go Register	14-61
14-43	MDEU Context Register	14-62
14-44	RNG Mode Register	14-64
14-45	RNG Data Size Register	14-64
14-46	RNG Reset Control Register	14-65
14-47	RNG Status Register	14-65
14-48	RNG Interrupt Status Register	14-66
14-49	RNG Interrupt Control Register	14-67
14-50	RNG EU-Go Register	14-68
14-51	AESU Mode Register	14-69
14-52	AESU Key Size Register	14-71
14-53	AESU Data Size Register	14-72
14-54	AESU Reset Control Register	14-72
14-55	AESU Status Register	14-73
14-56	AESU Interrupt Status Register	14-74
14-57	AESU Interrupt Control Register	14-75
14-58	AESU End of Message Register	14-77
14-59	AESU Context Register	14-77
14-60	Crypto-Channel Configuration Register (CCCR)	14-82
14-61	Header Dword Writeback Format	14-84
14-62	Crypto-Channel Pointer Status Register	14-85
14-63	Crypto-Channel Current Descriptor Pointer Register	14-90
14-64	Fetch FIFO	14-91
14-65	Data Packet Descriptor Buffer	14-92
14-66	EU Assignment Status Register (EUASR)	14-94
14-67	Interrupt Mask Register	14-95
14-68	Interrupt Status Register	14-96
14-69	Interrupt Clear Register	14-97
14-70	ID Register	14-98
14-71	IP Block Revision Register	14-98
14-72	Master Control Register	14-99
15-1	I ² C Block Diagram	15-1
15-2	I ² C _n Address Register (I2C _n ADR)	15-5
15-3	I ² C _n Frequency Divider Register (I2C _n FDR)	15-6
15-4	I ² C _n Control Register (I2C _n CR)	15-7
15-5	I ² C _n Status Register (I2C _n SR)	15-8
15-6	I ² C _n Data Register (I2C _n DR)	15-9
15-7	I ² C _n Digital Filter Sampling Rate Register (I2C _n DFSRR)	15-10

Figures

Figure Number	Title	Page Number
15-8	I ² C Interface Transaction Protocol.....	15-11
15-9	EEPROM Contents	15-18
15-10	EEPROM Data Format for One Register Preload Command.....	15-19
15-11	Example I ² C Interrupt Service Routine Flowchart	15-21
16-1	UART Block Diagram	16-2
16-2	Receiver Buffer Registers (URBR1 and URBR2).....	16-6
16-3	Transmitter Holding Registers (UTHR1 and UTHR2, UTHR3 and UTHR4)	16-6
16-4	Divisor Most Significant Byte Registers (UDMB1 and UDMB2, UDMB3 and UDMB4).....	16-7
16-5	Divisor Least Significant Byte Registers (UDLB1 and UDLB2, UDLB3 and UDLB4).....	16-7
16-6	Interrupt Enable Registers (UIER1 and UIER2, UIER3 and UIER4).....	16-8
16-7	Interrupt ID Registers (UIIR1 and UIIR2, UIIR3 and UIIR4)	16-9
16-8	FIFO Control Registers (UFCR1 and UFCR2, UFCR3 and UFCR4).....	16-11
16-9	Alternate Function Register (UAFR).....	16-11
16-10	Line Control Register (ULCR1 and ULCR2, ULCR3 and ULCR4).....	16-12
16-11	Modem Control Register (UMCR1 and UMCR2, UMCR3 and UMCR4)	16-14
16-12	Line Status Register (ULSR1 and ULSR2, ULSR3 and ULSR4)	16-15
16-13	Modem Status Register (UMSR1 and UMSR2, UMSR3 and UMSR4).....	16-16
16-14	Scratch Register (USCR)	16-17
16-15	DMA Status Register (UDSR).....	16-17
16-16	UART Bus Interface Transaction Protocol Example	16-19
17-1	JTAG Interface Block Diagram	17-1
18-1	DLL Block Diagram	18-1
18-2	DLL Application Example.....	18-3
18-3	DLL Override Register	18-4
18-4	DLL Status Register.....	18-5
18-5	DLL Clock Register	18-5
19-1	QUICC Engine Block Architectural Block Diagram.....	19-2
19-2	Data Paths	19-5
19-3	CMXUCR1[HBM1] and CMXUCR1[HBM3] location on the MPC8360E.....	19-8
A-1	MPC8358 Block Diagram.....	A-7

Tables

**Table
Number**

Title

**Page
Number**

Tables

Table Number	Title	Page Number
1-1	Chip-Level Pin Multiplexing Examples	1-12
1-2	QUICC Engine 2.0 Protocols.....	1-16
2-1	IMMR Memory Map	2-2
2-2	Memory Map.....	2-3
2-3	QUICC Engine High-Level Memory Map	2-25
3-1	MPC8360E Signal Reference by Functional Block.....	3-3
3-2	MPC8360E Alphabetical Signal Reference.....	3-11
3-3	MPC8360E Reset Configuration Signals.....	3-18
3-4	Output Signal States During System Reset.....	3-19
3-5	Signals for Multiplexing	3-20
3-6	CPODRx Field Descriptions.....	3-22
3-7	CPDATx Field Descriptions.....	3-22
3-8	CPDIRxy Field Descriptions	3-23
3-9	CPPARxy Field Descriptions.....	3-24
3-10	CPCExR Field Descriptions.....	3-25
3-11	CPOH1 Bit Setting.....	3-25
3-12	CPOH2 Bit Setting.....	3-26
3-13	Output Delay Options	3-26
3-14	RGMII Pins	3-29
3-15	GMII and TBI Pins	3-30
3-16	Port A Dedicated Pin Assignment	3-31
3-17	Port B Dedicated Pin Assignment.....	3-39
3-18	Port C Dedicated Pin Assignment.....	3-47
3-19	Port D Dedicated Pin Assignment	3-53
3-20	Port E Dedicated Pin Assignment.....	3-60
3-21	Port F Dedicated Pin Assignment	3-69
3-22	Port G Dedicated Pin Assignment	3-77
4-1	System Control Signals.....	4-1
4-2	External Clock Signals.....	4-3
4-3	Reset Causes	4-4
4-4	Reset Actions	4-5
4-5	Reset Configuration Words Source.....	4-10
4-6	Division.....	4-10
4-7	Selecting Reset Configuration Input Signals	4-11
4-8	RCWLR Bit Settings.....	4-12
4-9	System PLL VCO Division.....	4-13
4-10	System PLL Ratio	4-14
4-11	SPMF Maximum Values	4-14
4-12	QUICC Engine PLL Multiplication Factor.....	4-16
4-13	Reset Configuration Word High Bit Settings.....	4-17
4-14	PCI Host/Agent Configuration.....	4-19

Tables

Table Number	Title	Page Number
4-15	Boot Memory Space.....	4-19
4-16	Boot Sequencer Configuration.....	4-20
4-17	Boot ROM Location.....	4-21
4-18	Secondary DDR Configuration.....	4-22
4-19	e300 Core True Little-Endian	4-22
4-20	LALE Configuration	4-22
4-21	LDP Configuration.....	4-23
4-22	Local Bus Configuration EEPROM Addresses	4-23
4-23	Local Bus Reset Configuration Words Data Structure.....	4-24
4-24	Hard-Coded Reset Configuration Word Low Fields Values	4-29
4-25	Hard-Coded Reset Configuration Word High Field Values	4-30
4-26	Examples For Hard-Coded Reset Configuration Words Usage.....	4-30
4-27	Configurable Clock Units	4-33
4-28	Reset Configuration and Status Registers Memory Map.....	4-34
4-29	Reset Status Register Field Descriptions	4-35
4-30	RMR Field Descriptions	4-36
4-31	RPR Bit Descriptions	4-37
4-32	RCR Bit Settings.....	4-38
4-33	RCER Bit Settings	4-38
4-34	Clock Configuration Registers Memory Map.....	4-38
4-35	System PLL Mode Register Bit Settings	4-39
4-36	OCCR Bit Settings	4-40
4-37	SCCR Bit Descriptions	4-41
4-38	Clock Control DDR Register Address Map.....	4-42
4-39	MCKENR _n Field Descriptions	4-42
5-1	Local Access Windows Target Interface.....	5-1
5-2	Local Access Windows Example.....	5-2
5-3	Format of Window Definitions	5-3
5-4	Local Access Register Memory Map.....	5-5
5-5	IMMRBAR Bit Settings.....	5-7
5-6	ALTCBAR Bit Settings.....	5-8
5-7	LBLAWBAR0–LBLAWBAR3 Bit Settings.....	5-8
5-8	LBLAWBAR0[BASE_ADDR] Reset Value	5-8
5-9	LBLAWAR0–LBLAWAR3 Bit Settings.....	5-9
5-10	LBLAWAR0[EN] Reset Value.....	5-9
5-11	PCILAWBAR0–PCILAWBAR1 Bit Settings	5-10
5-12	PCILAWBAR0[BASE_ADDR] Reset Value	5-10
5-13	PCILAWAR0–PCILAWAR1 Bit Settings.....	5-11
5-14	PCILAWAR0[EN] Reset Value.....	5-11
5-15	DDRLAWBAR0–DDRLAWBAR1 Bit Settings.....	5-12
5-16	DDRLAWBAR0[BASE_ADDR] Reset Value	5-12

Tables

Table Number	Title	Page Number
5-17	DDRLAWAR0–DDRLAWAR1 Bit Settings	5-13
5-18	DDRLAWAR0[EN] Reset Value	5-14
5-19	SDDLAWBAR0–SDDLAWBAR1 Bit Settings.....	5-14
5-20	SDDLAWAR0–SDDLAWAR1 Bit Settings.....	5-15
5-21	Overlapping Local Access Windows	5-15
5-22	QUICC Engine Secondary Bus Access Windows Register Memory Map.....	5-18
5-23	LBMCSAR Bit Settings.....	5-18
5-24	SDMCSAR Bit Settings.....	5-19
5-25	LBMCEAR Bit Settings.....	5-19
5-26	SDMCEAR Bit Settings.....	5-20
5-27	LBMCAR Bit Settings.....	5-20
5-28	SDMCAR Bit Settings.....	5-21
5-29	Local Access Windows Example.....	5-22
5-30	QUICC Engine Secondary Bus Windows Settings.....	5-23
5-31	System Configuration Signal Properties	5-23
5-32	System Configuration Signal—Detailed Description	5-24
5-33	System Configuration Register Memory Map	5-24
5-34	SGPRL Bit Settings	5-25
5-35	SGPRH Bit Settings	5-25
5-36	SPRIDR Bit Settings	5-26
5-37	PARTID Coding	5-26
5-38	REVID Coding	5-26
5-39	SPCR Bit Settings	5-27
5-40	SICRL Bit Settings.....	5-29
5-41	SICRH Bit Settings	5-31
5-42	SICRH[29–31] Bit Settings	5-32
5-43	DDRCDR Field Descriptions.....	5-34
5-44	DDRDSR Field Descriptions	5-35
5-45	WDT Register Address Map.....	5-37
5-46	SWCRR Bit Settings.....	5-38
5-47	SWCNR Bit Settings.....	5-39
5-48	SWSRR Bit Settings	5-40
5-49	RTC Signal Properties.....	5-43
5-50	RTC External Signal—Detailed Signal Description	5-44
5-51	RTC Register Address Map	5-44
5-52	RTCNR Bit Settings.....	5-45
5-53	RTLDR Bit Settings	5-45
5-54	RTPSR Bit Settings.....	5-46
5-55	RTCTR Bit Settings	5-46
5-56	RTEVR Bit Settings	5-47
5-57	RTALR Bit Settings	5-47

Tables

Table Number	Title	Page Number
5-58	PIT Signal Properties	5-50
5-59	PIT External Signal—Detailed Signal Descriptions	5-51
5-60	PIT Register Address Map	5-51
5-61	PTCNR Bit Settings	5-52
5-62	PTLDR Bit Settings	5-52
5-63	PTPSR Bit Settings	5-53
5-64	PTCTR Bit Settings	5-53
5-65	PTEVR Bit Settings	5-54
5-66	GTM Signal Properties	5-59
5-67	GTM External Signals—Detailed Signal Descriptions	5-59
5-68	GTM Register Address Map	5-60
5-69	GTCFR1 Bit Settings	5-62
5-70	GTCFR2 Bit Settings	5-64
5-71	GTMDR Bit Settings	5-65
5-72	GTRFR Bit Settings	5-66
5-73	GTCPR _n Bit Settings	5-67
5-74	GTCNR Bit Settings	5-67
5-75	GTEVR _n Bit Settings	5-68
5-76	GTPSR _n Bit Settings	5-68
5-77	System Control Signals—Detailed Signal Descriptions	5-73
5-78	Power Management Controller Registers Memory Map	5-73
5-79	PMCCR Bit Settings	5-74
5-80	PMCER Bit Settings	5-75
5-81	PMCMR Bit Settings	5-75
6-1	Arbiter Register Map	6-2
6-2	ACR Field Descriptions	6-3
6-3	ATR Field Descriptions	6-4
6-4	AER Field Descriptions	6-5
6-5	AIDR Field Descriptions	6-6
6-6	AMR Field Descriptions	6-7
6-7	AEATR Field Descriptions	6-8
6-8	AEADR Field Descriptions	6-10
6-9	AERR Field Descriptions	6-10
6-10	Address Only Transaction Type Encoding	6-14
6-11	Reserved Transaction Type Encoding	6-15
6-12	Illegal Transaction Type Encoding	6-15
7-1	MSR Bit Descriptions	7-16
7-2	e300 HID0 Bit Descriptions	7-20
7-3	Using HID0[ECLK] and HID0[SBCLK] to Configure <i>clk_out</i>	7-22
7-4	HID1 Bit Descriptions	7-23
7-5	e300HID2 Bit Descriptions	7-23

Tables

Table Number	Title	Page Number
7-6	Interrupt Classifications	7-30
7-7	Exceptions and Interrupts.....	7-31
7-8	Differences Between e300 and G2_LE Cores	7-38
8-1	IPIC Signal Properties.....	8-5
8-2	IPIC External Signals—Detailed Signal Descriptions.....	8-5
8-3	IPIC Register Address Map	8-6
8-4	QUICC Engine Ports Interrupts Register Address Map	8-7
8-5	SICFR Field Descriptions	8-8
8-6	SIVCR Field Descriptions	8-9
8-7	IVEC/CVEC/MVEC Field Definition	8-10
8-8	SIPNR_H/SIFCR_H/SIMSR_H Bit Assignments.....	8-11
8-9	SIPNR_H Field Descriptions	8-12
8-10	SIPNR_L/SIFCR_L/SIMSR_L Bit Assignments	8-12
8-11	SIPNR_L Field Descriptions	8-13
8-12	SIPRR_A Field Descriptions	8-14
8-13	SIPRR_D Field Descriptions	8-15
8-14	SIMSR_H Field Descriptions	8-16
8-15	SIMSR_L Field Descriptions.....	8-16
8-16	SICNR Field Descriptions	8-17
8-17	SEPNR Field Descriptions.....	8-18
8-18	SMPRR_A Field Descriptions	8-19
8-19	SMPRR_B Field Descriptions	8-20
8-20	SEMSR Field Descriptions	8-21
8-21	SECNR Field Descriptions	8-22
8-22	SERSR/SERM/R/SERFR Bit Assignments.....	8-23
8-23	SERSR Field Descriptions	8-23
8-24	SERM/R Field Descriptions.....	8-24
8-25	SERC/R Field Descriptions.....	8-24
8-26	SIFCR_H Field Descriptions	8-25
8-27	SIFCR_L Field Descriptions.....	8-25
8-28	SEFCR Field Descriptions	8-26
8-29	SERFR Field Descriptions	8-27
8-30	SCVCR Field Descriptions	8-27
8-31	SMVCR Field Descriptions	8-28
8-32	CEPIER Bit Settings	8-29
8-33	CEPIMR Bit Settings.....	8-30
8-34	CEPICR Bit Settings.....	8-31
8-35	Interrupt Source Priority Levels.....	8-34
8-36	QUICC Engine Ports Interrupt Lines.....	8-39
9-1	DDR Memory Interface Signal Summary	9-4
9-2	Memory Address Signal Mappings.....	9-5

Tables

Table Number	Title	Page Number
9-3	Memory Interface Signals—Detailed Signal Descriptions	9-6
9-4	Clock Signals—Detailed Signal Descriptions	9-9
9-5	DDR Memory Controller Memory Map	9-10
9-6	CS _n _BNDS Field Descriptions	9-12
9-7	CS _n _CONFIG Field Descriptions	9-13
9-8	TIMING_CFG_3 Field Descriptions	9-14
9-9	TIMING_CFG_0 Field Descriptions	9-15
9-10	TIMING_CFG_1 Field Descriptions	9-17
9-11	TIMING_CFG_2 Field Descriptions	9-19
9-12	DDR_SDRAM_CFG Field Descriptions	9-21
9-13	DDR_SDRAM_CFG_2 Field Descriptions	9-24
9-14	DDR_SDRAM_MODE Field Descriptions	9-25
9-15	DDR_SDRAM_MODE_2 Field Descriptions	9-26
9-16	DDR_SDRAM_MD_CNTL Field Descriptions	9-27
9-17	Settings of DDR_SDRAM_MD_CNTL Fields	9-28
9-18	DDR_SDRAM_INTERVAL Field Descriptions	9-29
9-19	DDR_DATA_INIT Field Descriptions	9-29
9-20	DDR_SDRAM_CLK_CNTL Field Descriptions	9-30
9-21	DDR_INIT_ADDR Field Descriptions	9-30
9-22	DDR_IP_REV1 Field Descriptions	9-31
9-23	DDR_IP_REV2 Field Descriptions	9-31
9-24	DATA_ERR_INJECT_HI Field Descriptions	9-32
9-25	DATA_ERR_INJECT_LO Field Descriptions	9-32
9-26	ERR_INJECT Field Descriptions	9-33
9-27	CAPTURE_DATA_HI Field Descriptions	9-33
9-28	CAPTURE_DATA_LO Field Descriptions	9-34
9-29	CAPTURE_ECC Field Descriptions	9-34
9-30	ERR_DETECT Field Descriptions	9-35
9-31	ERR_DISABLE Field Descriptions	9-36
9-32	ERR_INT_EN Field Descriptions	9-36
9-33	CAPTURE_ATTRIBUTES Field Descriptions	9-37
9-34	CAPTURE_ADDRESS Field Descriptions	9-38
9-35	ERR_SBE Field Descriptions	9-39
9-36	Byte Lane to Data Relationship	9-44
9-37	Supported DDR1 SDRAM Device Configurations	9-45
9-38	Supported DDR2 SDRAM Device Configurations	9-46
9-39	DDR1 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled	9-46
9-40	DDR1 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled	9-47
9-41	DDR2 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled	9-48
9-42	DDR2 Address Multiplexing for 32-Bit Data Bus with Interleaving and Partial Array Self-Refresh Disabled	9-49

Tables

Table Number	Title	Page Number
9-43	Example of Address Multiplexing for 64-Bit Data Bus Interleaving between Two Banks with Partial Array Self Refresh Disabled.....	9-50
9-44	Example of Address Multiplexing for 64-Bit Data Bus Interleaving between Four Banks with Partial Array Self Refresh Disabled	9-51
9-45	DDR SDRAM Command Table.....	9-52
9-46	DDR SDRAM Interface Timing Intervals	9-53
9-47	DDR SDRAM Power-Saving Modes Refresh Configuration.....	9-61
9-48	Memory Controller—Data Beat Ordering	9-63
9-49	DDR SDRAM ECC Syndrome Encoding	9-64
9-50	DDR SDRAM ECC Syndrome Encoding (Check Bits)	9-65
9-51	Memory Controller Errors	9-67
9-52	Memory Interface Configuration Register Initialization Parameters.....	9-67
9-53	Programming Differences between Memory Types	9-68
10-1	Signal Properties—Summary.....	10-4
10-2	Local Bus Controller Detailed Signal Descriptions.....	10-5
10-3	Local Bus Controller Memory Map.....	10-9
10-4	BR _n Field Descriptions.....	10-11
10-5	Memory Bank Sizes in Relation to Address Mask	10-13
10-6	OR _n —GPCM Field Descriptions	10-14
10-7	OR _n —UPM Field Descriptions	10-16
10-8	OR _n —SDRAM Field Descriptions	10-17
10-9	MAR Field Descriptions	10-18
10-10	M _n MR Field Descriptions	10-19
10-11	MRTPR Field Descriptions.....	10-22
10-12	MDR Field Description.....	10-22
10-13	LSDMR Field Descriptions	10-22
10-14	LURT Field Descriptions	10-25
10-15	LSRT Field Descriptions.....	10-25
10-16	LTESR Field Descriptions	10-26
10-17	LTEDR Field Descriptions.....	10-27
10-18	LTEIR Field Descriptions	10-28
10-19	LTEATR Field Descriptions.....	10-29
10-20	LTEAR Field Descriptions.....	10-30
10-21	LBCR Field Descriptions.....	10-30
10-22	LCRR Field Descriptions.....	10-31
10-23	GPCM Write Control Signal Timing for LCRR[CLKDIV] = 4 or 8.....	10-38
10-24	GPCM Read Control Signal Timing for LCRR[CLKDIV] = 4 or 8.....	10-39
10-25	GPCM Write Control Signal Timing for LCRR[CLKDIV] = 2	10-40
10-26	GPCM Read Control Signal Timing for LCRR[CLKDIV] = 2.....	10-41
10-27	Boot Bank Field Values After Reset	10-49
10-28	SDRAM Interface Commands	10-51

Tables

Table Number	Title	Page Number
10-29	UPM Routines Start Addresses	10-62
10-30	UPM RAM Word Field Descriptions	10-67
10-31	MnMR Loop Field Use	10-72
10-32	UPM Address Multiplexing	10-73
10-33	Data Bus Requirements For Read Cycle	10-87
10-34	Micron SDRAM Devices	10-89
10-35	LADn Signal Connections to 128-Mbyte SDRAM	10-91
10-36	Logical Address Bus Partitioning	10-92
10-37	SDRAM Device Address Port During Address Phase	10-92
10-38	SDRAM Device Address Port During READ/WRITE Command	10-92
10-39	Register Settings for 128-Mbytes SDRAMs	10-93
10-40	Logical Address Partitioning	10-93
10-41	SDRAM Device Address Port During Address Phase	10-94
10-42	SDRAM Device Address Port During READ/WRITE Command	10-94
10-43	Register Settings for 512-Mbyte SDRAMs	10-94
10-44	UPM Synchronization Cycles	10-102
11-1	Sequencer Memory Map	11-2
11-2	POTARn Field Descriptions	11-3
11-3	POBARn Field Descriptions	11-4
11-4	POCMRn Field Descriptions	11-4
11-5	PMCR Field Descriptions	11-5
11-6	DTCR Field Descriptions	11-6
12-1	DMA Interface Signals—Detailed Signal Descriptions	12-2
12-2	Module Memory Map	12-3
12-3	OMISR Field Descriptions	12-5
12-4	OMIMR Field Descriptions	12-5
12-5	IMR0 and IMR1 Field Descriptions	12-6
12-6	OMR0 and OMR1 Field Descriptions	12-6
12-7	ODR Field Descriptions	12-7
12-8	IDR Field Descriptions	12-8
12-9	IMISR Field Descriptions	12-9
12-10	IMIMR Field Descriptions	12-9
12-11	DMAMRn Field Descriptions	12-10
12-12	DMASRn Field Descriptions	12-13
12-13	DMACDARn Field Descriptions	12-13
12-14	DMASARn Field Descriptions	12-14
12-15	DMASARn Field Descriptions	12-14
12-16	DMABCRn Field Descriptions	12-15
12-17	DMANDARn Field Descriptions	12-15
12-18	DMA Segment Descriptor Fields	12-20
13-1	PCI Controller Modes	13-3

Tables

Table Number	Title	Page Number
13-2	Signal Properties	13-4
13-3	PCI Interface Signals—Detailed Signal Descriptions	13-5
13-4	PCI Configuration Access Registers.....	13-11
13-5	PCI Memory-Mapped Registers	13-12
13-6	PCI_CONFIG_ADDRESS Field Descriptions	13-13
13-7	PCI_CONFIG_DATA Field Descriptions.....	13-15
13-8	PCI_ESR Field Descriptions.....	13-16
13-9	PCI_ECDR Field Descriptions	13-17
13-10	PCI_EER Field Descriptions	13-17
13-11	PCI_EATCR Field Descriptions	13-18
13-12	PCI_EACR Field Description.....	13-19
13-13	PCI_EEACR Field Description	13-20
13-14	PCI_EDLCR Field Description	13-20
13-15	PCI_GCR Field Descriptions.....	13-21
13-16	PCI_ECR Field Descriptions	13-22
13-17	PCI_GSR Field Descriptions	13-22
13-18	PITAR _n Field Descriptions	13-23
13-19	PIBAR _n Field Descriptions	13-23
13-20	PIEBAR _n Field Descriptions.....	13-24
13-21	PIWAR _n Field Descriptions.....	13-24
13-22	PCI Configuration Space Registers.....	13-26
13-23	Vendor ID Configuration Register Field Descriptions.....	13-27
13-24	Device ID Configuration Register Field Descriptions.....	13-27
13-25	PCI Command Configuration Register Field Descriptions.....	13-28
13-26	PCI Status Configuration Register Field Descriptions.....	13-29
13-27	Revision ID Configuration Register Field Descriptions	13-30
13-28	Standard Programming Interface Configuration Register Field Descriptions	13-30
13-29	Subclass Code Configuration Register Field Descriptions	13-31
13-30	Class Code Configuration Register Field Descriptions	13-31
13-31	Cache Line Size Configuration Register Field Descriptions	13-32
13-32	Latency Timer Configuration Register Field Descriptions	13-32
13-33	PIMMR Base Address Configuration Register Field Descriptions	13-33
13-34	GPL Base Address Register 0 Field Descriptions	13-34
13-35	GPL Base Address Register 1,2 Field Descriptions	13-35
13-36	GPL Extended Base Address Registers 1–2 Field Descriptions.....	13-35
13-37	Subsystem Vendor ID Configuration Register Field Descriptions	13-36
13-38	Subsystem Device ID Configuration Register Field Descriptions.....	13-36
13-39	Interrupt Line Configuration Register Field Descriptions	13-37
13-40	PCI Function Configuration Register Field Descriptions	13-38
13-41	PCI Arbiter Control Register (PCIACR) Field Descriptions.....	13-39
13-42	Hot Swap Register Block Field Descriptions	13-40

Tables

Table Number	Title	Page Number
13-43	PCI Command Definitions	13-43
13-44	Special Cycle Commands	13-53
14-1	Example Data Packet Descriptor	14-4
14-2	SEC Base Address Map	14-10
14-3	SEC Address Map	14-11
14-4	Header Dword Bit Definitions	14-17
14-5	EU_SEL1 and EU_SEL2 Values	14-18
14-6	Descriptor Types	14-18
14-7	Pointer Dword Field Definitions	14-20
14-8	Link Table Field Definitions	14-21
14-9	Descriptor Pointer Dword Usage	14-25
14-10	Mode Field Description	14-27
14-11	PKEURCR Field Descriptions	14-30
14-12	PKEU Status Register Field Descriptions	14-30
14-13	PKEUISR Field Descriptions	14-31
14-14	PKEUICR Field Descriptions	14-33
14-15	DEUMR Field Descriptions	14-35
14-16	DEUKSR Field Descriptions	14-36
14-17	DEURCR Field Descriptions	14-37
14-18	DEUSR Field Descriptions	14-38
14-19	DEUISR Field Descriptions	14-39
14-20	DEUICR Field Descriptions	14-40
14-21	AFEUMR Field Descriptions	14-44
14-22	AFEURCR Field Descriptions	14-46
14-23	AFEUSR Field Descriptions	14-47
14-24	AFEUISR Field Descriptions	14-48
14-25	AFEUICR Field Descriptions	14-49
14-26	MDEUMR in ‘Old’ Configuration	14-52
14-27	MDEUMR in ‘New’ Configuration	14-53
14-28	Mode Register—HMAC or SSL-MAC Generated by Single Descriptor	14-55
14-29	Mode Register—HMAC Generated across a Sequence of Descriptors	14-55
14-30	MDEU Reset Control Register Field Descriptions	14-56
14-31	MDEU Status Register Field Descriptions	14-57
14-32	MDEUISR Field Descriptions	14-58
14-33	MDEUICR Field Descriptions	14-60
14-34	RNG Mode Register Definitions	14-64
14-35	RNG Reset Control Register Field Descriptions	14-65
14-36	RNG Status Register Field Descriptions	14-66
14-37	RNG Interrupt Status Register Field Descriptions	14-66
14-38	RNG Interrupt Control Register Field Descriptions	14-67
14-39	AESU Mode Register Field Descriptions	14-69

Tables

Table Number	Title	Page Number
14-40	AES Cipher Modes	14-70
14-41	AESU Reset Control Register Field Descriptions	14-72
14-42	AESU Status Register Field Descriptions	14-73
14-43	AESU Interrupt Status Register Field Descriptions	14-74
14-44	AESU Interrupt Control Register Field Descriptions	14-75
14-45	Counter Modulus	14-78
14-46	CCCR Field Descriptions	14-83
14-47	Header Dword Writeback Field Descriptions	14-84
14-48	Crypto-Channel Pointer Status Register Signals	14-85
14-49	G_STATE Field Values	14-87
14-50	S_STATE Field Values	14-87
14-51	CHN_STATE Field Values	14-88
14-52	Crypto-Channel Pointer Status Register Error Field Definitions	14-89
14-53	Channel Pointer Status Register PTR_DW Field Values	14-89
14-54	Crypto-Channel Current Descriptor Pointer Register Signals	14-90
14-55	Fetch FIFO Field Descriptions	14-91
14-56	Channel Assignment Value	14-94
14-57	Interrupt Mask, Status, and Clear Register Fields	14-97
14-58	IP Block Revision Register Fields	14-98
14-59	Master Control Register Signals	14-99
15-1	I ² C Interface Signal Descriptions	15-3
15-2	I ² C Interface Signals—Detailed Signal Descriptions	15-4
15-3	I ² C Memory Map	15-4
15-4	I2CnADR Field Descriptions	15-5
15-5	I2Cn FDR Field Descriptions	15-6
15-6	I2CnCR Field Descriptions	15-7
15-7	I2CnSR Field Descriptions	15-8
15-8	I2CnDR Field Descriptions	15-9
15-9	I2CnDFSRR Field Descriptions	15-10
16-1	DUART Signal Overview	16-3
16-2	DUART Signals—Detailed Signal Descriptions	16-3
16-3	DUART Register Summary	16-5
16-4	URBR Field Descriptions	16-6
16-5	UTHR Field Descriptions	16-7
16-6	UDMB Field Descriptions	16-7
16-7	UDLB Field Descriptions	16-7
16-8	Baud Rate Examples	16-8
16-9	UIER Field Descriptions	16-9
16-10	UIIR Field Descriptions	16-10
16-11	UIIR IID Bits Summary	16-10
16-12	UFCR Field Descriptions	16-11

Tables

Table Number	Title	Page Number
16-13	UAFR Field Descriptions.....	16-12
16-14	ULCR Field Descriptions.....	16-13
16-15	Parity Selection Using ULCR[PEN], ULCR[SP], and ULCR[EPS]	16-14
16-16	UMCR Field Descriptions	16-14
16-17	ULSR Field Descriptions	16-15
16-18	UMSR Field Descriptions.....	16-16
16-19	USCR Field Descriptions.....	16-17
16-20	UDSR Field Descriptions.....	16-17
16-21	UDSR[TXRDY] Set Conditions.....	16-18
16-22	UDSR[TXRDY] Cleared Conditions.....	16-18
16-23	UDSR[RXRDY] Set Conditions.....	16-18
16-24	UDSR[RXRDY] Cleared.....	16-18
17-1	JTAG Test Signals Summary	17-2
17-2	JTAG Test—Detailed Signal Descriptions.....	17-2
18-1	DLL Macro External Signals	18-2
18-2	DLL Register Address Map	18-4
18-3	DLLOVR Field Description.....	18-4
18-4	DLLSR Field Description	18-5
18-5	DLLCK Field Description	18-6
19-1	QERM Chapters and MPC8360E Implementation	19-3
19-2	CERCR Field Descriptions	19-6
19-3	CEURNR Field Descriptions	19-7
19-4	Interrupt Routing and External Pins to QUICC Engine Block in MPC8360.....	19-7

Tables

**Table
Number**

Title

**Page
Number**

About This Book

The primary objective of this reference manual is to define the functionality of the MPC8360E PowerQUICC II Pro communications processor. The MPC8360E integrates a processor built on Power Architecture™ technology, with system logic required for networking, storage, and general purpose embedded applications. The e300 processor core is a low-power implementation of the family of reduced instruction set computing (RISC) embedded processors that implement Power Architecture technology. This book is intended as a companion to the *e300 Power Architecture Core Family Reference Manual*.

A new communications complex—the QUICC Engine block—forms the heart of the networking capability of the MPC8360E. The QUICC Engine contains several peripheral controllers and integrates two 32-bit RISC controllers. Each RISC controller can control multiple peripherals and they work together to provide increased aggregated system bandwidth for higher throughput applications. Protocol support is provided by the main workhorses of the device—the unified communication controllers (UCCs) and multi-channel communication controller (MCC).

Audience

It is assumed that the reader understands operating systems, microprocessor system design, and the basic principles of RISC processing.

Organization

Following is a summary and a brief description of the major parts of this reference manual:

Part I describes the many features of the MPC8360E integrated processor at an overview level. The following chapters are included:

- [Chapter 1, “Overview,”](#) provides a high-level description of features and functionality of the MPC8360E integrated processor. It describes the MPC8360E, its interfaces, and its programming model. The functional operation of the MPC8360E with emphasis on peripheral functions is also described.
- [Chapter 2, “Memory Map,”](#) describes the MPC8360E memory map, including the memory map of the QUICC Engine block. An overview of the local address map is followed by a complete listing of all internal memory mapped registers with cross references to the sections describing each.
- [Chapter 3, “Signal Descriptions,”](#) provides a listing of all the external signals, cross-references for signals that serve multiple functions, output signal states at reset, and tables containing QUICC Engine block multiplexing options.

Part II includes the following chapters:

- [Chapter 4, “Reset, Clocking, and Initialization,”](#) describes the hard and soft resets, the power-on reset sequence, power-on reset (POR) configuration, clocking, and initialization of the MPC8360E.

- [Chapter 5, “System Configuration,”](#) describes several functions of the MPC8360E that control the local access windows, system configuration, protection and general utilities.

Part III describes the core and I/O interfaces of the MPC8360E integrated processor. The following chapters are included:

- [Chapter 6, “Arbiter and Bus Monitor,”](#) provides an overview of the arbiter in the MPC8360E device. It also describes configuration, control, and status registers of the arbiter.
- [Chapter 7, “e300 Processor Core Overview,”](#) provides an overview of the basic functionality of the e300 processor core and briefly describes how the functional units interact.
- [Chapter 8, “Integrated Programmable Interrupt Controller \(IPIC\),”](#) describes the IPIC interrupt protocol, various types of interrupt sources controlled by the IPIC unit, and the IPIC registers with some programming guidelines. It also provides a definition of the external interrupt signals and their functions. In addition, the interrupt configuration, control, and status registers are described in this chapter.
- [Chapter 9, “DDR Memory Controller,”](#) describes the DDR SDRAM memory controller of the MPC8360E. This fully programmable controller supports most DDR memories available today, including both buffered and unbuffered devices. The 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. Special features like ECC error injection support rapid system debug.
- [Chapter 10, “Local Bus Controller,”](#) describes the local bus controller (LBC) of the MPC8360E. It describes the external signals and the memory-mapped registers as well as a functional description of the general-purpose chip-select machine (GPCM), synchronous DRAM (SDRAM) machine, and user-programmable machines (UPMs) of the LBC. Also, it includes an initialization and applications information section with many specific examples of its use.
- [Chapter 11, “Sequencer,”](#) describes how the I/O sequencer switches transactions among its ports, using a buffer pool to minimize blocking. It also provides address translation on outbound PCI transactions.
- [Chapter 12, “DMA/Messaging Unit,”](#) describes the four-channel high speed general-purpose DMA controller of the MPC8360E. The channels share buffer space in the IOS to facilitate the gathering and sending of data. The DMA/messaging unit supports communication between two processors on different buses, for example, a local processor and a processor on a PCI bus. This communication unit operates with generic messages and doorbell registers. This block also provides a DMA controller that transfers blocks of data independent of the local processor or PCI hosts.
- [Chapter 13, “PCI Bus Interface,”](#) describes the PCI interface, which complies with the *PCI Local Bus Specification*, Rev. 2.3. This chapter provides a basic description of PCI bus operations. The specific emphasis is directed at how this device implements the PCI specification.
- [Chapter 14, “Security Engine \(SEC\) 2.4,”](#) describes the SEC 2.4 that is designed to offload computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the e300 core of the MPC8360E. It is optimized to process all the algorithms associated with IPsec, IKE, SSL/TLS, iSCSI, SRTP, and IEEE Std. 802.11i.TM



- [Chapter 15, “I2C Interfaces,”](#) describes the inter-IC (IIC or I²C) bus controller of the MPC8360E. This synchronous, serial, bidirectional, multiple-master bus allows two-wire connection of devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The MPC8360E powers up in boot sequencer mode which allows the I²C controller to initialize configuration registers.
- [Chapter 16, “DUART,”](#) describes the (dual) universal asynchronous receiver/transmitters (UARTs) which feature a PC16552D-compatible programming model. These independent UARTs are provided specifically to support system debugging.
- [Chapter 17, “JTAG/Testing Support,”](#) describes the joint test action group (JTAG) interface of the MPC8360E to facilitate boundary-scan testing. The JTAG interface complies with the IEEE Std. 1149.1™ boundary-scan specification.
- [Chapter 18, “Delay Lock Loop \(DLL\),”](#) describes the theory of operation of the delay lock loop (DLL) module in the integrated device. Additionally, the configuration, control, and status registers are described.

Part IV describes the QUICC Engine block of the MPC8360E integrated processor. The following chapters are included:

- [Chapter 19, “QUICC Engine Block on the MPC8360E,”](#) serves as both a general overview of the QUICC Engine block and a guide to the specific implementation of the QUICC Engine block on the MPC8360E. The *QUICC Engine Block Reference Manual with Protocol Interworking (QEIWRM)* describes all the functional units of the QUICC Engine block and must be used in conjunction with this device manual.
- [Appendix A, “MPC8358E,”](#) illustrates the MPC8358E and in particular the differences between it and the MPC8360E as described in this manual.
- [Appendix B, “Revision History,”](#) lists the major differences between revisions of this reference manual.

Suggested Reading

This section lists additional reading that provides background for the information in this manual as well as general information about the architecture.

General Information

The following documentation, published by Morgan-Kaufmann Publishers, 340 Pine Street, Sixth Floor, San Francisco, CA, provides useful information about the PowerPC architecture and computer architecture in general:

- *The PowerPC Architecture: A Specification for a New Family of RISC Processors*, Second Edition, by International Business Machines, Inc.
- *Computer Architecture: A Quantitative Approach*, Third Edition, by John L. Hennessy and David A. Patterson
- *Computer Organization and Design: The Hardware/Software Interface*, Third Edition, by David A. Patterson and John L. Hennessy

Related Documentation

Freescale documentation is available from the sources listed on the back cover of this manual; the document order numbers are included in parentheses for ease in ordering:

- *Programming Environments Manual for 32-Bit Implementations of the PowerPC™ Architecture (MPCFPE32B)*—Describes resources defined by the PowerPC architecture.
- Reference manuals (formerly called user’s manuals)—These books provide details about individual implementations.
- Addenda/errata to reference or user’s manuals—Because some processors have follow-on parts an addendum is provided that describes the additional features and functionality changes. These addenda are intended for use with the corresponding reference or user’s manuals.
- Hardware specifications—Hardware specifications provide specific data regarding bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations.
- Technical summaries—Each device has a technical summary that provides an overview of its features. This document is roughly equivalent to the overview (Chapter 1) of an implementation’s reference or user’s manual.
- Application notes—These short documents address specific design issues useful to programmers and engineers working with Freescale processors.

Additional literature is published as new processors become available. For a current list of documentation, refer to www.freescale.com.

Conventions

This document uses the following notational conventions:

cleared/set	When a bit takes the value zero, it is said to be cleared; when it takes a value of one, it is said to be set.
mnemonics	Instruction mnemonics are shown in lowercase bold.
<i>italics</i>	<p>Italics indicate variable command parameters, for example, bcctrx.</p> <p>Book titles in text are set in italics</p> <p>Internal signals are set in lowercase italics, for example, <u>core int</u></p>
0x0	Prefix to denote hexadecimal number
0b0	Prefix to denote binary number
rA, rB	Instruction syntax used to identify a source GPR
rD	Instruction syntax used to identify a destination GPR
REG[FIELD]	Abbreviations for registers are shown in uppercase text. Specific bits, fields, or ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.
x	In some contexts, such as signal encodings, an unitalicized x indicates a don’t care.
<i>x</i>	An italicized <i>x</i> indicates an alphanumeric variable.

<i>n</i>	An italicized <i>n</i> indicates a numeric variable.
¬	NOT logical operator
&	AND logical operator
	OR logical operator
	Concatenation, for example TCR[WPEXT] TCR[WP]
—	Indicates a reserved bit field in an e300 register. Although these bits can be written to as ones or zeros, they are always read as zeros.

Signal Conventions

$\overline{\text{OVERBAR}}$	An overbar indicates that a signal is active-low.
<i>lowercase italics</i>	Lowercase italics is used to indicate internal signals.
lowercase_plaintext	Lowercase plain text is used to indicate signals that are used for configuration.

Acronyms and Abbreviations

Table i contains acronyms and abbreviations used in this document.

Table i. Acronyms and Abbreviated Terms

Term	Meaning
ADB	Allowable disconnect boundary
ATM	Asynchronous transfer mode
ATMU	Address translation and mapping unit
BD	Buffer descriptor
BIST	Built-in self test
BRI	Basic rate interface
BTB	Branch target buffer
BUID	Bus unit ID
CAM	Content-addressable memory
CSB	Coherent system bus
CCSR	Configuration control and status register
CEPT	Conférence des administrations Européenes des postes et télécommunications (European Conference of Postal and Telecommunications Administrations)
COL	Collision
CRC	Cyclic redundancy check
CRS	Carrier sense
DDR	Double data rate
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
DUART	Dual universal asynchronous receiver/transmitter
EA	Effective address
ECC	Error checking and correction
EEST	Enhanced Ethernet serial transceiver
EHPI	Enhanced host port interface
EPROM	Erasable programmable read-only memory
FCS	Frame-check sequence
GCI	General circuit interface
GMII	Gigabit media independent interface
GPCM	General-purpose chip-select machine
GPIO	General-purpose I/O
GPR	General-purpose register
GUI	Graphical user interface
HDLC	High-level data link control
I ² C	Inter-integrated circuit
IDL	Inter-chip digital link
IEEE	Institute of Electrical and Electronics Engineers
IPG	Interpacket gap
ISDN	Integrated services digital network
ITLB	Instruction translation lookaside buffer
IU	Integer unit
JTAG	Joint Test Action Group
LAE	Local access error
LAW	Local access window
LBC	Local bus controller
LIFO	Last-in-first-out
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
LSU	Load/store unit
MAC	Multiply accumulate, media access control
MDI	Medium-dependent interface
MESI	Modified/exclusive/shared/invalid—cache coherency protocol
MII	Media independent interface
MMU	Memory management unit

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
MSB	Most-significant byte
msb	Most-significant bit
NMSI	Nonmultiplexed serial interface
No-op	No operation
OSI	Open systems interconnection
PCI	Peripheral component interconnect
PCI/X	Abbreviation used to describe operation for both the PCI and PCI-X bus functionality
PCI-X	PCI extended
PCMCIA	Personal Computer Memory Card International Association
PCS	Physical coding sublayer
PIC	Programmable interrupt controller
PMA	Physical medium attachment
PMD	Physical medium dependent
POR	Power-on reset
PRI	Primary rate interface
RGMII	Reduced gigabit media independent interface
RISC	Reduced instruction set computing
RIO	Abbreviation occasionally used to refer to the RapidIO interface
RTOS	Real-time operating system
RWITM	Read with intent to modify
RWM	Read modify write
Rx	Receive
RxBD	Receive buffer descriptor
SCP	Serial control port
SDLC	Synchronous data link control
SDMA	Serial DMA
SFD	Start frame delimiter
SI	Serial interface
SIU	System interface unit
SMC	Serial management controller
SNA	Systems network architecture
SPI	Serial peripheral interface
SPR	Special-purpose register
SRAM	Static random access memory
TAP	Test access port

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
TBI	Ten-bit interface
TDM	Time-division multiplexed
TLB	Translation lookaside buffer
TSA	Time-slot assigner
Tx	Transmit
TxBD	Transmit buffer descriptor
UART	Universal asynchronous receiver/transmitter
UPM	User-programmable machine
USB	Universal serial bus
UTP	Unshielded twisted pair
VA	Virtual address
ZBT	Zero bus turnaround

Part I

Overview

Part I describes the many features of the MPC8360E integrated processor at an overview level. The following chapters are included:

- [Chapter 1, “Overview,”](#) provides a high-level description of features and functionality of the MPC8360E integrated processor. It describes the MPC8360E, its interfaces, and its programming model. The functional operation of the MPC8360E with emphasis on peripheral functions is also described.
- [Chapter 2, “Memory Map,”](#) describes the MPC8360E memory map, including the memory map of the QUICC Engine block. An overview of the local address map is followed by a complete listing of all internal memory mapped registers with cross references to the sections describing each.
- [Chapter 3, “Signal Descriptions,”](#) provides a listing of all the external signals, cross-references for signals that serve multiple functions, output signal states at reset, and tables containing QUICC Engine block multiplexing options.

Chapter 1

Overview

This chapter provides an overview of the MPC8360E PowerQUICC™ II Pro processor features, including a block diagram showing the major functional components. The MPC8360E is a cost-effective, highly integrated communications processor that addresses the needs of the networking, wireless infrastructure, and telecommunications markets. Target applications include next generation DSLAMs, network interface cards for 3G basestations (Node Bs), routers, media gateways, and high end integrated access devices (IADs). The MPC8360E extends current PowerQUICC II offerings, adding higher CPU performance, additional functionality, faster interfaces and interworking between various communication protocols while addressing the requirements related to time-to-market, price, power consumption, and board real estate.

Although this document is written from the perspective of the MPC8360E, most of the material applies to the MPC8358E as well. For information on differences between the MPC8360E and the MPC8358E, see [Appendix A, “MPC8358E.”](#)

1.1 MPC8360E PowerQUICC II Pro Processor Overview

[Figure 1-1](#) shows the major functional units within the MPC8360E. One major component of the MPC8360E is the e300c1 core which includes 32 Kbytes of instruction and 32 Kbytes of data cache and is compatible with the PowerPC 603e instruction set. Another is the new QUICC Engine™ block, which provides termination, interworking and switching between a wide range of communication protocols including ATM, Ethernet, HDLC, TDM, and POS. The QUICC Engine block's enhanced interworking eases the transition and reduces investment costs from ATM to IP based systems. Other major features include dual DDR/DDR2 SDRAM memory controllers, a 32-bit PCI controller, a flexible local bus, and a dedicated security engine.

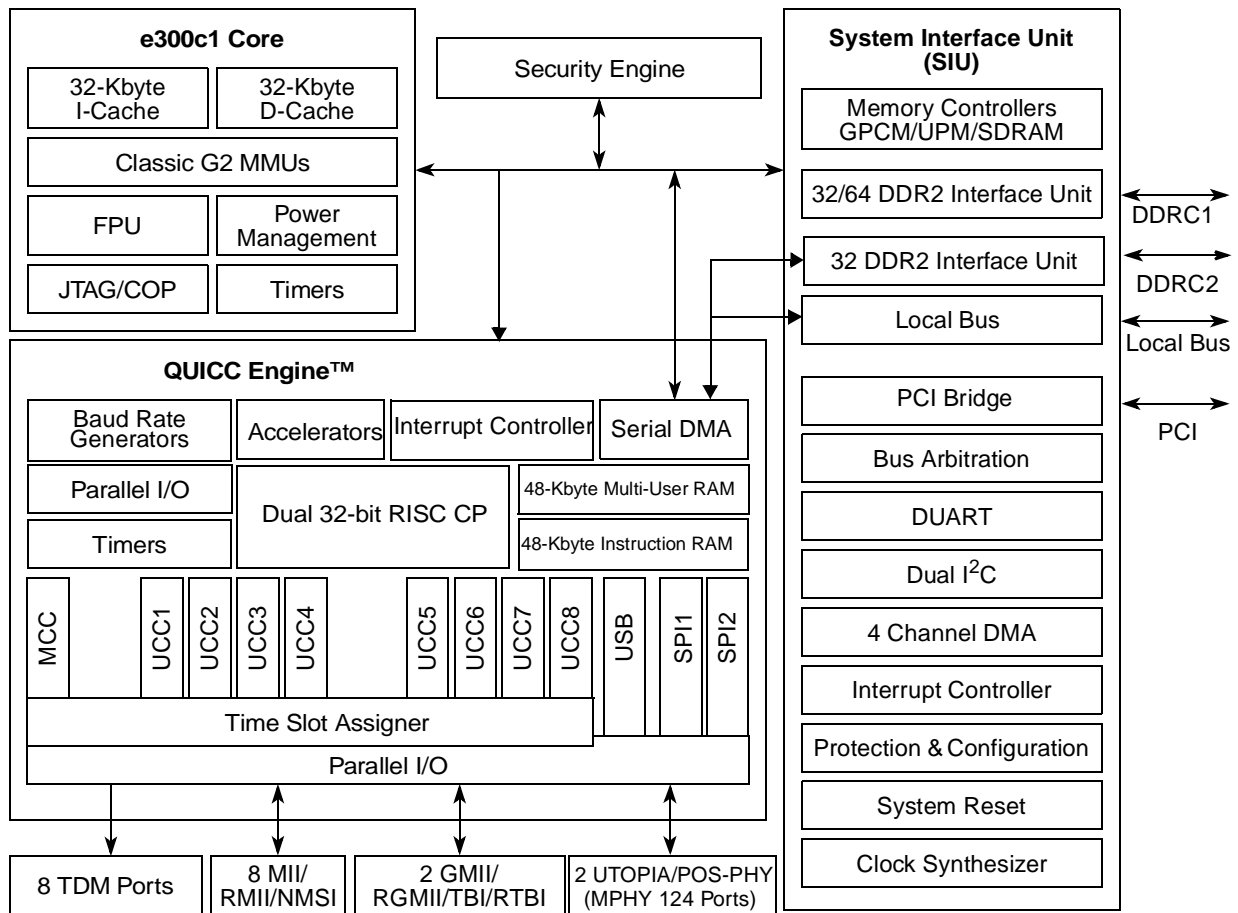


Figure 1-1. MPC8360E Block Diagram

The major features of this device are as follows:

- e300c1 Power Architecture™ processor core
 - Enhanced version of the MPC603e core
 - High-performance, superscalar processor core with a four-stage pipeline and low interrupt latency times
 - Floating-point, integer, load/store, system register, and branch processing units
 - 32-Kbyte instruction cache and 32-Kbyte data cache with lockable capabilities
 - Dynamic power management
 - Enhanced hardware program debug features
 - Software-compatible with Freescale processor families implementing Power Architecture technology
 - Separate PLL that is clocked by the system bus clock

- QUICC Engine 2.0 block
 - Two 32-bit RISC controllers for flexible support of the communications peripherals with the following features:
 - One clock per instruction
 - Separate PLL for operating frequency that is independent of system's bus and core frequency for power and performance optimization
 - 32-bit instruction object code
 - Executes code from internal ROM or RAM
 - 32-bit arithmetic logic unit (ALU) data path
 - Modular architecture allowing for easy functional enhancements
 - Slave bus for CPU access of registers and multiuser RAM space
 - 48 Kbytes of instruction RAM
 - 48 Kbytes of multiuser data RAM
 - QE peripheral request interface (for SEC, PCI, and IEEE Std. 1588™)
 - Two serial DMA channels that are optimized for burst transfers and can perform simultaneous accesses to DDR1/DDR2 memory and to the DDR2 memory/local bus
 - Eight unified communication controllers (UCCs) on the MPC8360E and six UCCs on the MPC8358E supporting the following protocols and interfaces:
 - 10/100 Mbps Ethernet/IEEE® Std. 802.3® through MII and RMII interfaces¹. Note that SMII or SGMII media-independent interface is not supported currently.
 - 1000 Mbps Ethernet/IEEE Std. 802.3 through GMII, RGMII, TBI and RTBI interfaces.
 - 10/100 Mbps Ethernet/IEEE Std. 802.3 L2 switch port through MII and RMII interfaces.
 - IEEE Std. 1588™ support
 - ATM/POS through the UPC
 - Serial ATM through the serial interface
 - HDLC/Transparent (bit rate up to 50 Mbps)
 - HDLC BUS (bit rate up to 10 Mbps)
 - UART
 - BISYNC (bit rate up to 2 Mbps)
 - QUICC multichannel controller (QMC) for 128 TDM channels with 64 channels per UCC
 - PPP, Multi-Link (ML-PPP), Multi-Class (MC-PPP), and PPP multiplexing support
 - One multichannel communication controller (MCC) supporting the following (MPC8360E only):
 - 256 HDLC or transparent channels
 - 128 SS#7 channels
 - Transparent, HDLC or SS#7 per channel
 - Channel multiplexing on up to eight TDM interfaces

1. SMII and SGMII media-independent interface are not supported currently.

- ATM controller
 - Full duplex SAR protocols at OC-12 rate through UTOPIA L2
 - AAL5, AAL2, AAL1, AAL0 protocols. TM 4.1 CBR, VBR, UBR, UBR+ traffic types
 - 64 K external connections
 - Inverse multiplexing ATM capability (IMA)
 - 2 UTOPIA/POS-PHY L2 bus controllers (UPC) for MPC8360E and one UPC for MPC8358E supporting 124 ports (optional 128 ports with extended address)
- Universal serial bus (USB) controller
 - USB 1.1 full/low rate compatible
 - USB 2.0 full/low rate compatible (not high speed)
 - USB host mode
 - USB slave mode
- Two serial peripheral interfaces (SPIs). SPI2 is dedicated to Ethernet PHY management.
- Time slot assigner and 8 TDM serial interfaces on the MPC8360E and 4 TDM serial interfaces on the MPC8358E
 - Support T1, CEPT, T1/E1, T3/E3, pulse code modulation highway, ISDN primary rate, Freescale inter chip digital link (IDL), and user-defined TDM serial interfaces. Frame synchronization.
 - Independent Rx and Tx routing RAM with 512 routing entries each.
 - Time slot assigner with bit or byte resolution.
 - TDM interfaces have 1-bit mode for E3/T3 rates in clear channel.
- QUICC Engine interrupt controller supports 4 external and 19 internal discrete interrupt sources and 2 interrupt levels in the system IPIC
- Sixteen independent baud rate generators and 30 input pins to clock the UCCs, SI, UPCs, USB, time-stamps and timer.
- Four independent 16-bit timers that can be interconnected as two 32-bit timers
- Interworking functionality:
 - 8-port L2 10/100-Base T Ethernet switch
 - ATM-to-ATM switching (AAL0, 2, 5)
 - TDM-to-ATM, circuit emulation (CES)
 - Additional interworking functions are supplied as RAM-based microcode packages.
- Security engine optimized to handle all the algorithms associated with IPSec, SSL/TLS, SRTP, IEEE 802.11i® standard, iSCSI, and IKE processing. The security engine contains four crypto-channels, a controller, and a set of crypto execution units (EUs). The execution units are:
 - Public key execution unit (PKEU) supporting the following:
 - RSA and Diffie-Hellman algorithms
 - Programmable field size up to 2048 bits
 - Elliptic curve cryptography
 - F2m and F(p) modes

- Programmable field size up to 511 bits
- Data encryption standard execution unit (DEU)
 - DES and 3DES algorithms
 - Two key (K1, K2) or three key (K1, K2, K3) for 3DES
 - ECB and CBC modes for both DES and 3DES
- Advanced encryption standard unit (AESU)
 - Implements the Rijndael symmetric-key cipher
 - Key lengths of 128-, 192-, and 256-bits
 - ECB, CBC, CCM, and counter (CTR) modes
 - XOR parity generation accelerator for RAID applications
- ARC four execution unit (AFEU)
 - Implements a stream cipher compatible with the RC4 algorithm
 - 40- to 128-bit programmable key
- Message digest execution unit (MDEU)
 - SHA with 160-, 224-, or 256-bit message digest
 - MD5 with 128-bit message digest
 - HMAC with either algorithm
- Random number generator (RNG)
- Four crypto-channels, each supporting multi-command descriptor chains
 - Static and/or dynamic assignment of crypto-execution units through an integrated controller
 - Buffer size of 256 bytes for each execution unit, with flow control for large data sizes
- Dual DDR SDRAM memory controllers
 - Programmable timing supporting both DDR1 and DDR2 SDRAM
 - Configurable as two 32-bit buses (MPC8360E only) or one 32-/64-bit bus
 - Up to 333-MHz data rate
 - 64-Mbit to 2-Gbit devices with x8/x16/x32 data ports (no direct x4 support)
 - Up to four physical banks (chip selects), each bank up to 1 Gbyte independently addressable
 - Full ECC support (when configured as 2x32-bit DDR memory controllers, both support ECC)
 - On-die termination (ODT) support when using DDR2
 - Driver impedance calibration support (using MDIC signals) when using DDR2
 - Support for up to 16 simultaneous open pages for DDR1 (up to 32 pages for DDR2)
 - Read-modify-write support
 - Sleep-mode support for SDRAM self refresh
 - Supports auto refresh
 - On-the-fly power management using CKE
 - Registered DIMM support
 - 2.5-V SSTL2 compatible I/O for DDR1, 1.8-V SSTL_18 compatible I/O for DDR2

- PCI interface
 - Designed to comply with *PCI Local Bus Specification, Revision 2.3*
 - 32-bit PCI interface operating at up to 66 MHz
 - PCI 3.3-V compatible
 - Not 5-V compatible
 - Support for host and agent modes. When in host mode, the PCI controllers support external signal isolation, thus enabling power to shut off external devices.
 - Support for PCI-to-memory and memory-to-PCI streaming
 - Memory prefetching of PCI read accesses and support for delayed read transactions
 - Support for posting of processor-to-PCI and PCI-to-memory writes
 - On-chip arbitration, supporting three masters on PCI
 - Selectable hardware-enforced coherency
- Local bus controller (LBC)
 - Multiplexed 32-bit address and data operating at up to 133 MHz
 - Eight chip selects supporting eight external slaves
 - Up to eight-beat burst transfers
 - 32-, 16-, and 8-bit ports are controlled by an on-chip memory controller
 - Three protocol engines available on a per chip select basis:
 - General-purpose chip select machine (GPCM)
 - Three user programmable machines (UPMs)
 - Dedicated single data rate SDRAM controller
 - Parity support
 - Default boot ROM chip select with configurable bus width (8, 16, or 32 bits)
- Integrated programmable interrupt controller (IPIC)
 - Functional and programming compatibility with the MPC8260 interrupt controller
 - System interrupt controller supports 8 external, 25 internal discrete interrupt sources and 2 QUICC Engine interrupt levels
 - Support for one external (optional) and seven internal machine check interrupt sources
 - Programmable highest priority request
 - Four groups of interrupts with programmable priority
 - External and internal interrupts directed to communication processor
 - Redirects interrupts to external $\overline{\text{PCI_INTA}}$ signal when in core disable mode
 - Unique vector number for each interrupt source
- Dual I²C interfaces
 - Two-wire interface
 - Multiple-master support
 - Master or slave I²C mode support
 - On-chip digital filtering rejects spikes on the bus

- Boot sequencer
- DMA (Direct memory access) controller
 - Four independent fully programmable DMA channels
 - Handshaking (external control) signals supported for all channels: $\overline{\text{DMA_DREQ}}[0:3]$, $\overline{\text{DMA_DACK}}[0:3]$, $\overline{\text{DMA_DDONE}}[0:3]$
 - Misaligned transfer capability for source/destination address
 - Data chaining and direct mode
 - Interrupt on completed segment and chain
- DUART
 - Two 4-wire interfaces (RxD, TxD, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$)
 - Programming model compatible with the original 16450 UART and the PC16550D
- Parallel I/O
 - General-purpose I/O (GPIO)
 - Open drain capability
 - Interrupt capability
- System timers
 - Periodic interrupt timer
 - Real-time clock
 - Software watchdog timer
 - Eight general-purpose timers
- IEEE Std. 1149.1™ compliant JTAG boundary scan
- Integrated PCI bus and SDRAM clock generation

1.2 MPC8360E Architecture Overview

The following sections describe the major functional units of this device.

1.2.1 Power Architecture Core

The device contains the e300c1 Power Architecture processor core, which is an enhanced version of the MPC603e core (used in previous generations of PowerQUICC II processors). Enhancements include twice as much L1 cache (32-Kbyte data cache and 32-Kbyte instruction cache) with integrated parity checking and other performance-enhancing features. The e300 core is upward software-compatible with existing MPC603e core-based products.

For detailed information regarding the processor core refer to the following:

- The *e300 Power Architecture™ Core Family Reference Manual* (chapters describing the programming model, cache model, memory management model, exception model, and instruction timing) (Document No. E300CORERM)
- The *Programming Environments Manual for 32-Bit Implementations of the PowerPC™ Architecture* (Document No. MPCFPE32B)

The e300 core is a low-power implementation of the family of microprocessors that implements Power Architecture technology. The core implements the 32-bit portion of the architecture, which provides 32-bit effective addresses, integer data types of 8, 16, and 32 bits, and floating-point data types of 32 and 64 bits.

The core is a superscalar processor that can issue three instructions (two plus a branch) and completes and retires as many as two instructions per clock cycle. Instructions can execute out of order for increased performance; however, the core makes completion appear sequential.

The e300c1 core integrates five execution units—an integer unit (IU) with full multiply and divides, a floating-point unit (FPU), a branch processing unit (BPU) with static branch prediction, a load/store unit (LSU) for data transfers, and a system register unit (SRU). The ability to execute five instructions in parallel and the use of simple instructions with rapid execution times yield high efficiency and throughput. Most integer instructions execute in one clock cycle. The FPU is pipelined so a single-precision multiply-add instruction can be issued and completed every clock cycle.

The e300c1 core provides independent on-chip, 32-Kbyte, eight-way set-associative, physically-addressed instruction and data caches with parity and integrated way lock capabilities. The processor also features independent on-chip instruction and data memory management units (MMUs). The MMUs contain 64-entry, two-way set-associative, data and instruction translation lookaside buffers (DTLB and ITLB) that provide support for demand-paged virtual memory address translation. The caches use a pseudo least recently used (PLRU) replacement algorithm; the TLBs use a least recently used (LRU) replacement algorithm. The processor also supports block address translation through the use of two independent instruction and data block address translation (IBAT and DBAT) arrays of eight entries each. Effective addresses are compared simultaneously with all eight entries in the BAT array during block translation. In accordance with the architecture, if an effective address hits in both the TLB and BAT array, the BAT translation takes priority.

As an added feature to the e300 core, the device can lock the contents of one to all ways in the instruction and data cache (or an entire cache). For example, this allows embedded applications to lock interrupt routines or other important (time-sensitive) instruction sequences into the instruction cache. It allows data to be locked into the data cache, which may be important to code that must have deterministic execution.

The e300 core has high-performance 64-bit data bus and 32-bit address bus interfaces to the rest of the device. The e300 core supports single-beat and burst data transfers for memory accesses, and memory-mapped I/O operations.

Figure 1-2 provides a block diagram of the e300 core that shows how the execution units (IU, FPU, BPU, LSU, and SRU) operate independently and in parallel. Note that this is a conceptual diagram and does not attempt to show how these features are physically implemented on the chip.

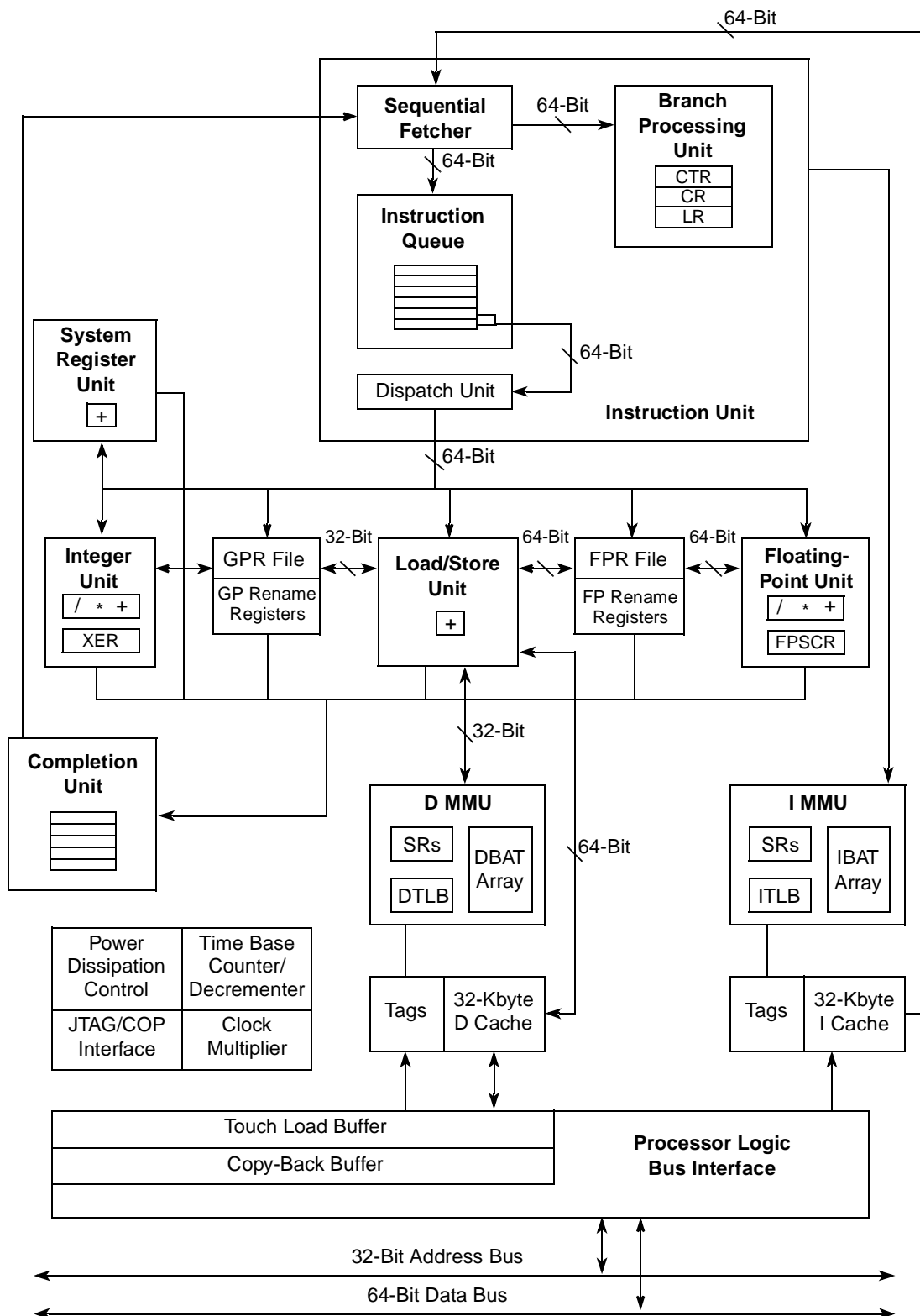


Figure 1-2. MPC8360E Integrated e300c1 Core Block Diagram

1.2.2 QUICC Engine 2.0 Block

The QUICC Engine block is a versatile communications complex that integrates several communications peripheral controllers. It provides an on-chip system design that can be used as a building block for chip integration in a variety of applications, particularly in communications and networking systems.

The QUICC Engine block is the next generation of the Power QUICC II CPM and maintains a high level of compatibility with it. It contains the following communication peripherals:

- Eight unified communication controllers (UCCs) on the MPC8360E and six UCCs on the MPC8358E with the following features:
 - Ethernet, ATM, HDLC/HDLC bus and transparent protocols (also known as fast protocols).
 - UART, BiSync, Async HDLC, Serial ATM and QMC that are user compatible with the SCC of the CPM (also known as slow protocols)
 - Ethernet for the First Mile (IEEE 802.3ah 2BASE-TL and 10PASS-TS)
 - The HDLC and transparent protocols are user compatible with the FCC of the CPM.
- Two UTOPIA-packet over SONET (POS) PHY L2 controllers (UPC) for 124/128 ports (MPC8358E has only one POS PHY L2 controller)
- Two serial peripheral controllers (SPI1 and SPI2¹)
- Multi channel controller (MCC) for 256 channels (MPC8360E only).
- Time slot assigner and serial interface (SI) for 8 TDMs and full duplex routing RAM of 512 entries (MPC8358E has four TDMs).
- One universal serial bus controller (USB 1.1/2.0)

The UCCs are similar to the PowerQUICC II peripherals: SCC (BISYNC, UART, and HDLC bus), and FCC (fast Ethernet, HDLC, transparent, and ATM). In addition, 2×124 UTOPIA PHYs are supported in ATM mode. The QUICC Engine block presents enhanced flexibility by allowing the user to configure the UCCs to support a Layer-2 Ethernet switch.

Figure 1-3 shows the internal architecture and the interfaces provided by the QUICC Engine block. The QUICC Engine block contains two identical groups of four UCCs. Each group is controlled by a RISC engine. A common multiuser RAM is used to store parameters for RISC engines. Each RISC has a ROM associated with it, which contains the code image. The instruction RAM is used to run RISC code from the RAM.

1. SPI2 can be used only for Ethernet management

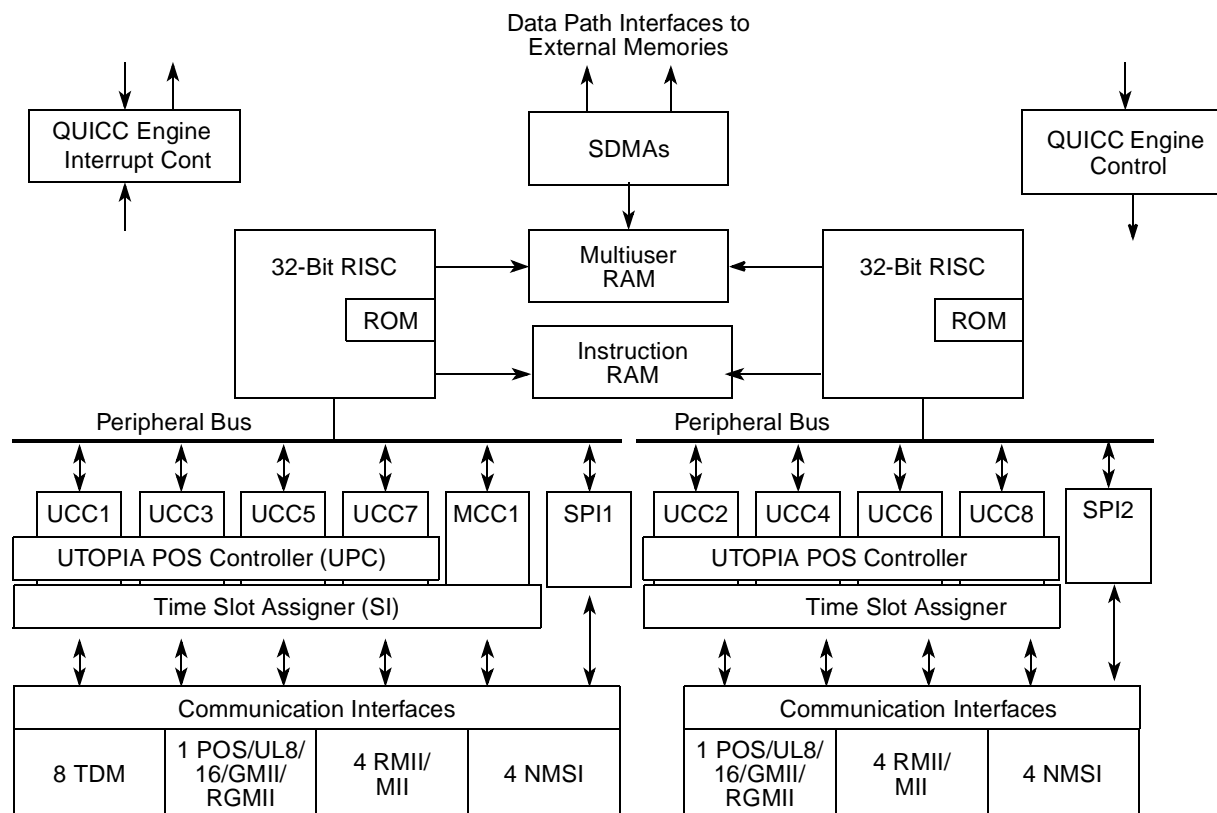


Figure 1-3. QUICC Engine Block Architectural Block Diagram

1.2.2.1 Examples of Chip-Level Pin Multiplexing

The MPC8360E with the QUICC Engine block can be used to implement a DSLAM with up to 96 ports, a BTS, an RNC control PCU, or an RNC BTS in an ATM framework. [Table 1-1](#) shows some pin multiplexing options for the QUICC Engine block.

The first example (DSLAM 48–96 ports¹) shows that UCC1 and UCC2 are configured for gigabit Ethernet through the RGMII interface. UCC3 and UCC8 are configured for Fast Ethernet through the MII interface. UCC5 and UCC7 are both configured for ATM, provided through the UTOPIA POS controller (UPC1) as a UTOPIA interface with 124 PHYs, and UCC4/UCC6 are likewise configured for ATM through the UTOPIA interface controlled by UPC2. One UART is available through the SIU, SPI1 is used for SPI, and SPI2 is used for Ethernet management; three timers, and two pins remain available for I²C's. E1/T1 and PCI are not available in this configuration.

1. This example is given for the sake of pin multiplexing; performance is not considered.

Table 1-1. Chip-Level Pin Multiplexing Examples

	DSLAM 48-96 Ports	DSLAM 12-48 Ports	BTS/NodeB	RNC Control PCU	RNC BTS AAL2 SU + DS PU2	ROBO Router	DSP Aggregator
For each example, the specific UCC mentioned below is configured to the indicated interface							
UCC1	RGMII	RGMII	RGMII	RGMII	—	RGMII	RGMII
UCC2	RGMII	RGMII	—	—	—	RGMII	RGMII
UCC3	MII	MII	MII	MII	MII	UPC1-ATM (SPHY)	RMII
UCC4	UPC2-ATM 124	—	—	—	—	RMII	RMII
UCC5	UPC1-ATM 124	UPC1-POS 124	UPC1-POS	UPC1-ATM (SPHY)	—	—	RMII
UCC6	UPC2-ATM 124	—	—	—	—	—	RMII
UCC7	UPC1-ATM 124	UPC1-POS 124	—	—	—	—	RMII
UCC8	MII	MII	MII	MII	MII	RMII	RMII
The number of peripheral devices used in each example is indicated in the following rows:							
UART	1	1	1	1	1	1	1
E1/T1	0	0	8	8	8	8	0
SPI	1	1	1	1	1	1	1
Timers	3	2	1	1	1	1	1
I²C	2	2	2	2	2	2	2
SMI (Ethernet Mgt)	1	1	1	1	1	1	1
PCI	0	1	1	1	0	1	1

The remaining examples in this chapter could be similarly charted.

1.2.2.2 Differences Between the QUICC Engine Block and the MPC82xx/85xx CPM

The following MPC82xx/MPC85xx features have been modified for the QUICC Engine block:

- SCC DPLL is not provided
- SCC 10 Base T (7-wire Ethernet) is no longer provided
- HDLC bus protocol programming model is FCC- instead of SCC-compatible
- IDMA I²C functions are provided as part of the SIU.
- Support for MSP and policer features as standard on the QUICC Engine block

- Enhanced Ethernet controller which provides support for frame filtering based on the VLAN tag or any Ethernet type field and parsing of frame headers to perform table lookups
- 4-port L2 Ethernet switch
- ATM available bit rate (ABR) scheduling mode is not supported. Other scheduling modes are supported.
- UTOPIA external rate is supported, but the QUICC Engine block does not transmit idle cells when no data is available.
- GCI circuits through the serial interface are not supported.
- The instruction RAM is indirectly accessed.

1.2.2.3 Enhanced Features of the QUICC Engine Block Compared with the CPM

The following list highlights some significant improvements in the QUICC Engine block:

- Dual RISC controller architecture.
- ATM enhancements:
 - Two UTOPIA L2 interfaces which can connect to up to 124/128 ports each.
 - Full duplex SAR protocols at OC-12 rate
 - Support for MSP and policer features as standard on the QUICC Engine block
 - E2AAL2 is provided as standard on the QUICC Engine block, including a WFQ mode, support for external TxQDs and statistic gathering.
 - Address look-up enhancements:
 - Internal mini-CAM
 - Lookup based on user's defined cell extra header
 - Transmit scheduler enhancements:
 - Small memory foot-print scheduler for low bit-rate connections (scalable APC)
 - Hierarchal frame based scheduling
 - APC flux compensation
 - Simultaneous processing of multiple cells (multi-threading)
- Enhanced Ethernet features that provide for:
 - Gigabit Ethernet through GMII/RGMII and TBI/RTBI
 - Frame filtering based on the MAC destination and source address, VLAN tag field and parsing of frame headers to perform table lookups.
 - IP support for IPv4 and IPv6 packets including TOS and header checksum processing.
 - UEC controller for 10/100/1000 Mbps with support of VLAN
 - L2 switch controller for 10/100 Mbps using MAC address or IEEE Std. 802.1P/Q VLAN tags
- QUICC Engine peripheral request interface (for SEC, PCI, and IEEE Std. 1588™)
- PPP, ML-PPP, MC-PPP and PPP mux are provided as standard on the QUICC Engine block.
- USB protocol: automatic transmission of SOF tokens.
- Flexibility to form an Ethernet layer 2 switch with up to 8 ports.

Overview

- SPI controller implement Ethernet MII serial management interface for up to 32 PHYs.
- Multi-channel controller (MCC) on MPC8360E (MPC8358E does not have an MCC)
 - Multiplexes up to 256 HDLC, transparent, or SS#7¹ channels on one to eight TDM interfaces.
 - Superchannels time-slot of 5,6,7,8, or 16 bits.
 - Optional channel underrun mode: Underrun event disables an individual channel instead of globally disabling the MCC. Per channel soft recovery from underrun is possible.

1. 128 SS#7 channels

- Serial interface
 - Support for multi-frame on a single TDM link (TDME)
 - Nibble-parallel data interface on all TDMs.
- TC layer for serial ATM supported as a microcode package.
- IEEE Std. 1588 support
- User can modify the peripheral's (UCC, MCC, SPI) parameter RAM base address in the multi-user RAM
- User programmable FIFO size for UCC fast protocols.
- The QUICC Engine operating frequency is independent of the SIU bus frequency, providing greater performance/power flexibility.
- Two independent time-stamp registers triggered by an external or internal clock

1.2.2.4 Software Migration from the MPC82xx/MPC85xx Family Devices

The QUICC Engine block was designed to minimize the changes required in run time code developed for the PowerQUICC II in order to ease the code porting from previous PowerQUICC II and CPM enabled devices (MPC82xx family, MPC85xx CPM enabled derivatives) to this device. Special attention was given to maintain compatibility with interrupts, events, status, interrupt event queues and data descriptors. However, significant changes have been made in the Ethernet and ATM controllers yielding the significant increase in performance when using those protocols.

Although some registers are new, many registers in the QUICC Engine block retain the previous mode, status, and event bits. The buffer descriptor method of transferring data from the QUICC Engine block to the CPU is maintained. The initialization code differs from that of the MPC82xx.

The UCC is a unification between the SCC and the FCC in the MPC85xx/MPC82xx families of devices. In UART and BISYNC modes, the UCC programming model is compatible with the SCC; in HDLC, transparent, Ethernet, and ATM modes, the UCC programming model is compatible with the FCC.

The UCC Ethernet controller (UEC) has commonalities with the triple speed Ethernet controller (TSEC) present in the MPC85xx, as well as the FCC in the MPC82xx. Some of the mode bits in the FCC programming model have been moved to other registers present in the MPC8560 TSEC controller. The data structures (buffer descriptors) are the same as in the FCC (and the TSEC).

The ATM controller initialization structures reflect the increase in bit rate that is provided in this device, and differ from those in the MPC82xx.

For the multi-channel HDLC/transparent controller, the QUICC Engine block provides the following two options:

- Full compatibility with the QUICC multi-channel controller (QMC) running on top of the UCC providing 64 channels compatible with the MPC82xx QMC
- Multi-channel controller providing 256 channels compatible with the MCC in the CPM.

For all other protocols (UART, HDLC, transparent, BiSync, Async HDLC, HDLC Bus, channelized HDLC/transparent), the QUICC Engine initialization is almost identical to the MPC82xx/MPC85xx.

A new programming model is provided for protocols that are not supported in the MPC82xx family of devices, such as POS. Extensions to the programming model are provided for protocols that run at a high bit rate, such as gigabit Ethernet and OC-12/STM-4 ATM. The SPI and USB peripherals are software compatible with those of the CPM. Also, the QUICC Engine timers are compatible with the CPM's. Finally, serial ATM is offered as a standard microcode protocol for the UCC instead of the FCC2 UTOPIA and hardware enabled TC layer of the CPM.

1.2.2.5 Serial Protocol Table

Table 1-2 summarizes the available protocols for each serial port.

Table 1-2. QUICC Engine 2.0 Protocols

Protocol	Controller					
	UCC	MCC	SPI	USB	UPC	SI
ATM, IMA (UTOPIA)	√	—	—	—	√	—
Serial ATM, IMA	√ (Opt)	√	—	—	—	√
CES	√	√	—	—	√	√
POS	√	—	—	—	√	—
Ethernet, L2 switch	√	—	—	—	—	—
HDLC	√	—	—	—	—	—
HDLC_BUS	√	—	—	—	—	—
Async HDLC	√	—	—	—	—	—
BiSync	√	—	—	—	—	—
Transparent	√	—	—	—	—	—
UART	√	—	—	—	—	—
Multi channel HDLC/Transparent TDM	√	√	—	—	—	√
Multi channel SS#7 TDM	—	√	—	—	—	√
ISDN (IDL)	√	√ (Opt)	—	—	—	√
PPP	—	√	—	—	—	√
SPI	—	—	√	—	—	—
Ethernet management (SMI)	√ (Opt)	—	√	—	—	—
USB	—	—	—	√	—	—

1.2.2.6 QUICC Engine Configurations

The QUICC Engine block offers configuration flexibility for specific applications. The previously-mentioned functions are all available, but not all of them can be used at the same time. The two physical factors that limit the functionality in any given system are performance and pinout. A pin multiplexing tool is provided to simplify the programming of the various options.

Please contact a Freescale FAE for more information on serial performance.

1.2.3 Security Engine

A hardware encryption block is also integrated in the device. It supports many encryption algorithms allowing for high performance data encryption and authentication as required in today's SoHo/RoBo routers. The encryption block is compatible with the corresponding block in the MPC8280.

The security engine supports DES, 3DES, MD-5, SHA-1, AES, PKEU, RNG, and RC-4 encryption algorithms in hardware. It also includes XOR parity generation acceleration for RAID applications.

A block diagram of the security engine's internal architecture is shown in Figure 1-4. The bus interface module is designed to transfer 64-bit words between the internal bus and any register inside the security engine.

An operation begins with a write of a pointer to a crypto-channel fetch register that points to a data packet descriptor. The channel requests the descriptor and decodes the operation to be performed. The channel then requests the controller to assign crypto execution units and fetch the keys, IVs, and data needed to perform the given operation. The controller satisfies the requests by assigning execution units to the channel and by making requests to the master interface. As data is processed, it is written to the individual execution unit's output buffer and then back to system memory through the bus interface module.

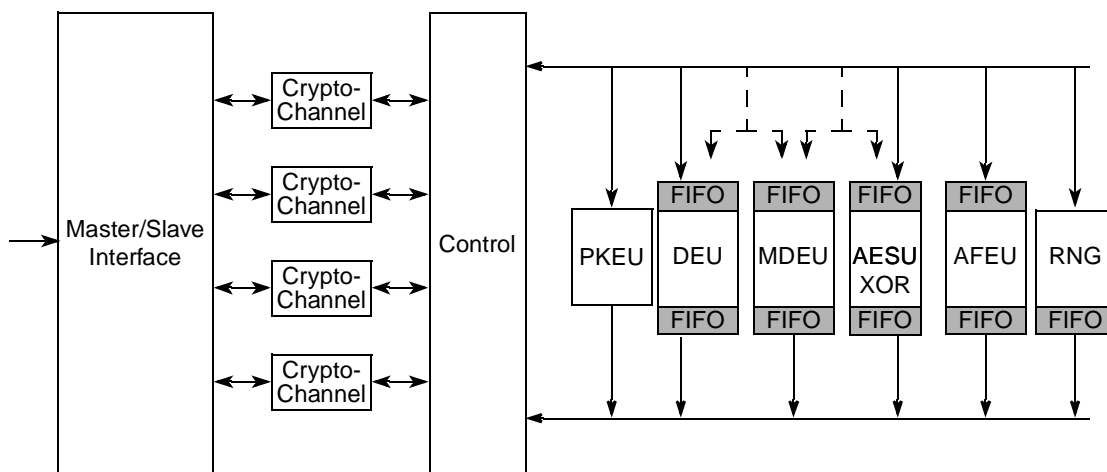


Figure 1-4. Integrated Security Engine Functional Blocks

1.2.4 Dual DDR Memory Controllers

These fully programmable dual DDR SDRAM controllers support most JEDEC standard x8 or x16 DDR1 or DDR2 memories available today, including buffered and unbuffered DIMMs. The MPC8360E dual DDR buses can be configured as two 32-bit buses or one 64-bit bus, and the MPC8358E dual DDR buses can be configured as one 32-bit or one 64-bit bus. However, mixing nonregistered and registered DIMMs in the same system is not supported. The built-in error checking and correction (ECC) ensures very low bit-error rates for reliable high-frequency operation. The DDR memory controller supports ECC error injection for rapid system debug. Dynamic power management and auto-precharge modes simplify memory system design.

The DDR memory controllers include the following features:

- Support for DDR1 and DDR2 SDRAM
- 32-/40-bit and 64-/72-bit SDRAM data bus
- Programmable settings for meeting all SDRAM timing parameters
- Many different SDRAM configurations supported
 - Support for as many as four physical banks (chip selects), each bank independently addressable
 - Support for 64-Mbit to 1-Gbit devices with x8 or x16 data ports (no direct x4 support).
 - Support for unbuffered and registered DIMMs
- Support for data mask signals and read-modify-write operations for sub-double word writes
- Support for double-bit error detection and single-bit error correction ECC (8-bit check word across 64- or 32-bit data when in 32-bit mode)
- Two-entry input request queue
- Open page management (dedicated entry for each sub-bank)
- Memory controller clock frequency of two times the SDRAM clock with support for sleep power management mode
- Support for error injection on ECC

1.2.5 PCI Controller

The 32-bit PCI controller is compatible with the *PCI Local Bus Specification, Rev. 2.3*. The PCI interface can function as a host bridge interface. The PCI interface can optionally function as an agent device. The PCI controller supports 32-bit addressing and 32-bit data buses.

As a host, the device supports read and write operations to the PCI memory space, the PCI I/O space, and the PCI configuration space. Also, the device can generate PCI special-cycle and interrupt acknowledge commands. As an agent, the device supports read and write operations to system memory, as well as PCI configuration space and the on-chip memory mapped configuration space.

The device PCI controller includes the following distinctive features:

- Address stepping on configuration transactions
- Fast back-to-back transactions
- Data streaming
- When in host mode, the PCI controller supports external signal isolation, thus enabling power shut off to external devices

1.2.5.1 PCI Bus Arbitration Unit

The PCI controller contains a PCI bus arbitration unit, which eliminates the need for an external unit, thus lowering system complexity and cost. It has the following features:

- Supports three $\overline{\text{REQ}}/\overline{\text{GNT}}$ signal pairs, thus supporting three external masters. The device PCI controller is the fourth member of the arbitration pool.
- The bus arbitration unit allows fairness as well as a priority mechanism.

- A two-level round-robin scheme is used in which each device can be programmed within a pool of a high- or low-priority arbitration. One member of the low-priority pool is promoted to the high-priority pool. As soon as it is granted the bus, it returns to the low-priority pool.
- The unit can be disabled to allow a remote arbitration unit to be used.
- The unit can be isolated to allow power shut off of external devices.

1.2.6 Local Bus Controller (LBC)

The main component of the local bus controller (LBC) is its memory controller, which provides a seamless interface to many types of memory devices and peripherals. The memory controller is responsible for controlling eight memory banks shared by a high performance SDRAM machine, a general-purpose chip-select machine (GPCM), and up to three user-programmable machines (UPMs). As such, it supports a minimal glue logic interface to synchronous DRAM (SDRAM), SRAM, EPROM, Flash EPROM, burstable RAM, and other peripherals. The LBC external address latch enable (LALE) signal allows multiplexing of addresses with data signals to reduce the device pin count.

The local bus controller also includes a number of data checking and protection features such as data parity generation and checking, write protection, and a bus monitor to ensure that each bus cycle is terminated within a user-specified period.

The main features of the local bus controller (LBC) are as follows:

- Memory controller with eight memory banks (chip selects)
 - 32-bit address decoding with mask
 - Variable memory block sizes (32 Kbytes to 2 Gbytes in FCM mode)
 - Selection of control signal generation on a per-bank basis
 - Data buffer controls activated on a per-bank basis
 - Odd/even parity checking including read-modify-write (RMW) parity for single accesses
 - Parity byte-select
 - Atomic operation
- Synchronous DRAM (SDRAM) machine
 - Provides the control functions and signals for glueless connection to JEDEC-compliant SDRAM devices
 - Supports up to four concurrent open pages
 - Supports SDRAM port size of 32-, 16-, and 8-bit
 - Supports external address and/or command line buffering
- General-purpose chip-select machine (GPCM)
 - Compatible with SRAM, EPROM, FEPROM, and peripherals
 - Global (boot) chip-select available at system reset
 - Boot chip-select support for 8-, 16-, 32-bit devices
 - Minimum three-clock access to external devices
 - Four byte-write-enable signals ($\overline{\text{LWE}}[0:3]$)

- Three user-programmable machines (UPMs)
 - Programmable-array-based machine controls external signal timing with a granularity of up to one quarter of an external bus clock period
 - User-specified control-signal patterns run when an internal master requests a single-beat or burst read or write access
 - User-specified control-signal patterns can be initiated by software
 - Support for 8-, 16-, 32-bit devices
 - Page mode support for successive transfers within a burst
- Delay locked loop (DLL) with software-configurable bypass for generating low-frequency bus clocks

1.2.7 Integrated Programmable Interrupt Controller (IPIC)

The IPIC implements the necessary functions to provide a flexible solution for general-purpose interrupt control. The IPIC includes the following features:

- Functional and programming models are compatible with the MPC8260 interrupt controller
- Support for external and internal discrete interrupt sources, and 2 QUICC Engine interrupt levels
- Support for one external (optional) and seven internal machine checkstop interrupt sources
- Programmable highest priority request
- Two programmable priority mixed groups of four on-chip and four external interrupt signals with two priority schemes for each group: grouped and spread
- Two programmable priority internal groups of eight on-chip interrupt signals with two priority schemes for each group: grouped and spread
- Priority interrupts can be programmed to support a critical (\overline{cint}) or system management (\overline{smi}) interrupt type
- External and internal interrupts directed to a communication processor
- Unique vector number for each interrupt source
- Ability to redirect interrupts to external $\overline{PCI_INTA}$ pin when in core disable mode

1.2.8 Dual I²C Interfaces

The inter-IC (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-master bus of the I²C allows the connection of additional devices to the bus for expansion and system development.

The I²C controller is a true multi-master bus which includes collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously. This feature allows for complex applications with multiprocessor control. The I²C controller consists of a transmitter/receiver unit, clocking unit, and control unit. The I²C unit supports general broadcast mode and on-chip filtering rejects spikes on the bus.

The I²C interfaces include the following features:

- Two-wire interface
- Multi-master operational
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Bus busy detection
- Software-programmable clock frequency
- Software-selectable acknowledge bit
- On-chip filtering for spikes on the bus
- Address broadcasting supported
- System initialization data is optionally loaded from I²C EPROM by boot sequencer embedded hardware

1.2.9 DMA Controller

The DMA engine is capable of transferring blocks of data from any legal address range to any other legal address range. Therefore, it can perform a DMA transfer between any of its I/O or memory ports, or even between two devices or locations on the same port.

The DMA controller offers the following features:

- Four high-speed/high-bandwidth channels accessible by local and remote masters
- Handshaking (external control) signals supported for all channels: $\overline{\text{DREQ}}[0:3]$, $\overline{\text{DACK}}[0:3]$, $\overline{\text{DDONE}}[0:3]$
- Basic DMA operation modes (direct, simple chaining)
- Support for misaligned transfers
- Programmable bandwidth control between channels
- Interrupt on error and completed segment or chain

1.2.10 Dual Universal Asynchronous Receiver/Transmitter (DUART)

The device includes a DUART intended for use in maintenance, bring up, and debug systems. The device provides a standard four-wire handshake (TXD, RXD, RTS, CTS) for each port. The DUART is a slave interface. An interrupt is provided to the interrupt controller or optionally steered externally to allow device handshakes. Interrupts are generated for transmit, receive, and line status.

The DUART supports full-duplex operation. It is compatible with the PC16450 and PC16550 programming models. The transmitter and receiver both support 16-byte FIFOs.

Software programmable baud rate generators divide the system clock to generate a 16x clock. Serial interface data formats (data length, parity, 1/1.5/2 STOP bit, baud rate) are also software selectable.

The DUART includes the following features:

- Full-duplex operation

- Programming model compatible with the original PC16450 UART and the PC16550D (an improved version of the PC16450 that also operates in FIFO mode)
- PC16450 register reset values
- FIFO mode for both transmitter and receiver, providing 16-byte FIFOs
- Serial data encapsulation and decapsulation with standard asynchronous communication bits (START, STOP, and parity)
- Maskable transmit, receive, and line status interrupts
- Software-programmable baud rate generators that divide the system clock by 1 to $(2^{16} - 1)$ and generate a 16x clock for the transmitter and receiver engines
- Clear to send ($\overline{\text{CTS}}$) and ready to send ($\overline{\text{RTS}}$) MODEM control functions
- Software-selectable serial-interface data format (data length, parity, 1/1.5/2 STOP bit, baud rate)
- Line status registers
- Line-break detection and generation
- Internal diagnostic support, local loopback, and break functions
- Prioritized interrupt reporting
- Overrun, parity, and framing error detection

1.2.11 System Timers

The system includes the following timers:

- Periodic interrupt timer
- Real time clock
- Software watchdog timer
- Two general-purpose timer blocks, each supporting four 16-bit programmable timers, two cascaded 32-bit timers, or one cascaded 64-bit counter

1.3 Application Examples

As technology standards for telecommunication equipment including; wireless infrastructure, DSLAMs, routers/switches, line cards, multi-channel modems, network storage, and IADs continue to change and new access technologies are introduced, a programmable communication processing platform that can evolve to accommodate such changes provides equipment vendors with a distinct competitive advantage. These system requirements allow the MPC8360E PowerQUICC II Pro processor to be used in a number of applications as described in the following sections.

1.3.1 SME Router

Figure 1-5 shows a SME router connecting a local area network (LAN) to an ADSL port. The LAN is comprised of an L2 Ethernet switch with VLAN and IGMP snooping support, which is implemented with a microcode package running on four UCCs. The ADSL port is implemented with the fifth UCC by enabling the ATM protocols on this peripheral. Together with the Power Architecture core and the system

interface unit to a DDR DRAM and a local bus with a memory controller, the MPC8360E with the QUICC Engine block provides a full system solution for this class of product.

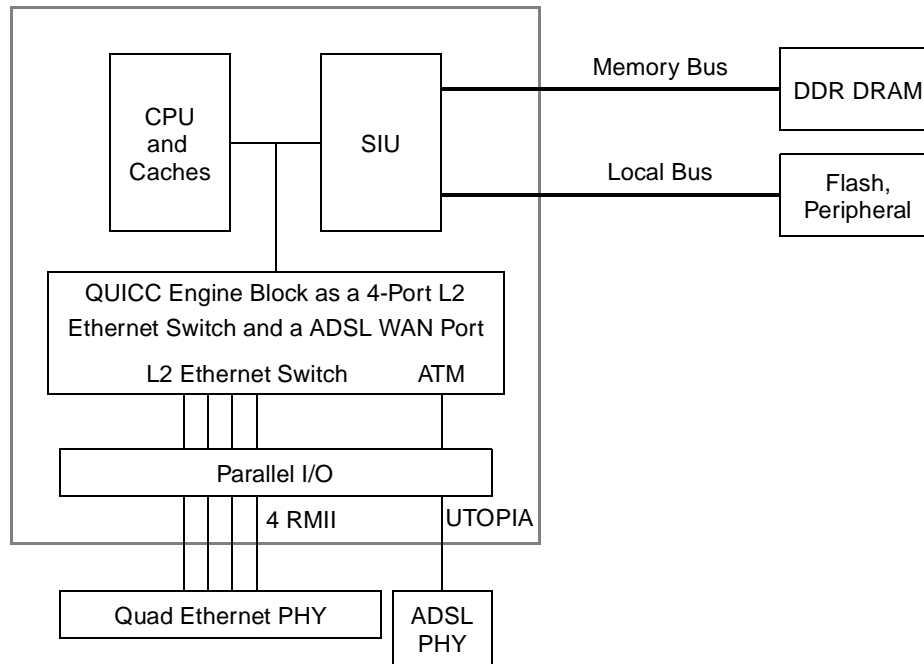


Figure 1-5. QUICC Engine Block in the MPC8360E Used as a SME Router

1.3.2 DSLAM Line Card

In the application shown in [Figure 1-6](#), the MPC8360E with the QUICC Engine block is used as the data path of a DSLAM line card. The line card supports ATM multi-PHY subscriber lines and an Ethernet uplink. The Ethernet uplink may be either MII for a 10-/100-Mbps rates or GMII for gigabit rates, depending on the overall throughput requirements for the card. The device includes a CPU, a system interface unit, and a memory controller, thus providing an integrated solution for an DSLAM line card.

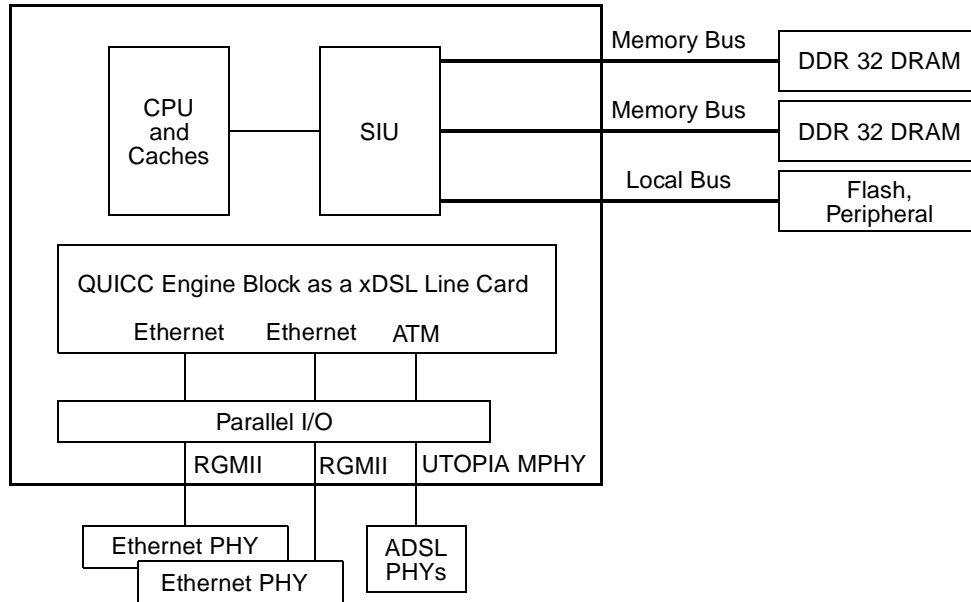


Figure 1-6. QUICC Engine Block in the MPC8360E Used in an DSLAM Line Card

1.3.3 NodeB/BTS—Network Interface Card

In the application shown in Figure 1-7, the QUICC Engine block is used as the data aggregator on a wireless infrastructure network interface card (NIC). This card connects the NodeB/BTS switch to the external network via E1/T1 lines carrying either IMA ATM or multilink multiclass PPP or circuit emulation into the system internal bus which can be ATM based or IP based (GMII/MII/RMII). Other network connections may be from POS or from an internal Ethernet.

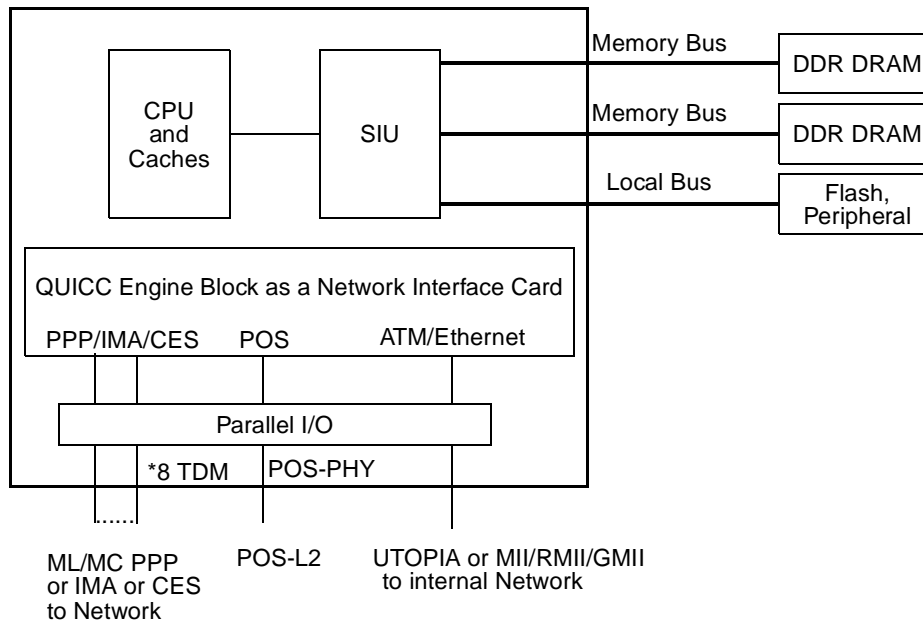


Figure 1-7. Wireless Infrastructure—NodeB/BTS Network Interface Card

Chapter 2

Memory Map

This chapter describes the MPC8360E memory map. The internal memory mapped registers are described, including a complete listing of all memory mapped registers with cross references to the sections detailing descriptions of each.

2.1 Internal Memory Mapped Registers

All of the memory mapped registers in the device are contained within a 2-Mbyte address region. To allow for flexibility, the base address of the memory mapped registers is relocatable in the local address space. The local address map location of this register block is controlled by the internal memory mapped registers base address register (IMMRBAR), see [Section 5.2.4.1, “Internal Memory Map Registers Base Address Register \(IMMRBAR\),”](#) for more information. The default value for IMMRBAR is 0xFF40_0000.

2.2 Accessing IMMR Memory From the Local Processor

When the local e300 processor is used to configure IMMR space, the IMMR memory space should typically be marked as cache-inhibited and guarded.

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

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

2.3 Complete IMMR Map

Reading from address locations which appear as reserved in the memory map table is not guaranteed to return predictable data. Writing to address locations which appear as reserved in the memory map table is not allowed and could lead to unpredictable behavior of the device. Reserved bits in non-reserved registers will be read as zero unless the reset value of those bits is different due to internal logic considerations.

When writing to registers with reserved bits, those reserved bits should be cleared. By doing so, existing software would be able to run on a future modified device in which some reserved bits were allocated for enhanced modes. This would allow for maintaining the legacy functionality when set to zero.

In certain specific cases, reserved bits should not be cleared but should keep their reset value. Thus, the software should perform a ‘read-modify-write’ and make sure that it does not change the reset value of those bits. The description of the specific bits will indicate when this is needed.

Unless stated otherwise in a particular block, all accesses to and from the memory mapped registers must be made with 32-bit accesses. There is no support for accesses of sizes other than 32 bits.

Table 2-1 lists the memory-mapped register regions (windows).

Table 2-1. IMMR Memory Map

Address	Use	Actual Size	Window	Cross Reference
0x00_0000–0x00_01FF	System configuration	512 bytes	512 bytes	Table 2-2
0x00_0200–0x00_02FF	Watchdog timer	16 bytes	256 bytes	Table 2-2
0x00_0300–0x00_03FF	Real time clock	32 bytes	256 bytes	Table 2-2
0x00_0400–0x00_04FF	Periodic interval timer	32 bytes	256 bytes	Table 2-2
0x00_0500–0x00_05FF	Global timers module 1	64 bytes	256 bytes	Table 2-2
0x00_0600–0x00_06FF	Global timers module 2	64 bytes	256 bytes	Table 2-2
0x00_0700–0x00_07FF	Integrated programmable interrupt controller (IPIC)	128 bytes	256 bytes	Table 2-2
0x00_0800–0x00_08FF	System arbiter	30 bytes	256 bytes	Table 2-2
0x00_0900–0x00_09FF	Reset module		256 bytes	Table 2-2
0x00_0A00–0x00_0AFF	Clock module	44 bytes	256 bytes	Table 2-2
0x00_0B00–0x00_0BFF	Power management control module		256 bytes	Table 2-2
0x00_0C00–0x00_0CFF	QUICC Engine™ ports interrupts	24 bytes	256 bytes	Table 2-2 ,
0x00_0D00–0x00_0DFF	Reserved	—	256 bytes	
0x00_0E00–0x00_0EFF	Reserved, should be cleared	—	256 bytes	
0x00_0F00–0x00_0FFF	Clock control secondary DDR	20 bytes	256 bytes	Table 2-2
0x00_1000–0x00_10FF	Clock control primary DDR	20 bytes	256 bytes	Table 2-2
0x00_1100–0x00_11FF	Clock and DLL Control, LBC	20 bytes	256 bytes	Table 2-2
0x00_1200–0x00_12FF	Reserved, should be cleared	—	256 bytes	
0x00_1300–0x00_13FF	Reserved	—	256 bytes	
0x00_1400–0x00_17FF	QUICC Engine parallel I/O ports	168 bytes	1 Kbyte	Table 2-2
0x00_1800–0x00_1BFF	QUICC Engine secondary bus access windows	24 bytes	1 Kbytes	Table 2-2
0x00_1C00–0x00_1FFF	Reserved	—	1 Kbyte	
0x00_2000–0x00_2FFF	DDR MEMC	3.8 Kbytes	4 Kbytes	Table 2-2
0x00_3000–0x00_30FF	I ² C1 controller	24 bytes	256 bytes	Table 2-2
0x00_3100–0x00_31FF	I ² C2 controller	24 bytes	256 bytes	Table 2-2
0x00_3200–0x00_03FFF	Reserved, should be cleared	—	3.5 Kbytes	
0x00_4000–0x00_44FF	Reserved, should be cleared	—	—	
0x00_4500–0x00_46FF	DUART	18 bytes x 2	4 Kbytes	Table 2-2
0x00_4700–0x00_4FFF	Reserved, should be cleared	—	—	
0x00_5000–0x00_5FFF	LBC	224 bytes	4 Kbytes	Table 2-2
0x00_6000–0x00_7FFF	Reserved	—	—	
0x00_8000–0x00_82FF	DMA	768 bytes	768 bytes	Table 2-2
0x00_8300–0x00_837F	PCI configuration	16 bytes	128 bytes	Table 2-2

Table 2-1. IMMR Memory Map (continued)

Address	Use	Actual Size	Window	Cross Reference
0x0_8380–0x0_83FF	Reserved	16 bytes	128 bytes	
0x00_8400–0x00_84FF	IOS	256 bytes	256 bytes	Table 2-2
0x00_8500–0x00_85FF	PCI controller	128 bytes	256 bytes	Table 2-2
0x00_8600–0x00_CFFF	Reserved	—	—	
0x00_D000–0x00_DFFF	Secondary DDR MEMC	3.8 Kbytes	4 Kbytes	
0x00_E000–0x02_5FFF	Reserved	—	—	
0x02_6000–0x02_FFFF	Reserved, should be cleared	—	—	
0x03_0000–0x03_FFFF	Security engine	52 Kbytes	64 Kbytes	Table 2-2
0x04_0000–0x0F_FFFF	Reserved, should be cleared	—	—	
0x10_0000–0x1F_FFFF	QUICC Engine 2.0 block	256 Kbytes	1 Mbyte	Table 2-3

[Table 2-2](#) lists the memory-mapped registers.

Table 2-2. Memory Map

Offset	Register	Access	Reset	Section/Page
System Configuration Registers				
0x0_0000	IMMRBAR—Internal memory map base address register	R/W	0xFF40_0000	5.2.4.1/5-6
0x0_0004	Reserved, should be cleared	—	—	—
0x0_0008	ALTCBAR—Alternate configuration base address register	R/W	All zeros	5.2.4.2/5-7
0x0_000C–0x0_001C	Reserved, should be cleared	—	—	—
0x0_0020	LBLAWBAR0—LBC local access window 0 base address register	R/W	All zeros ¹	5.2.4.3/5-8
0x0_0024	LBLAWAR0—LBC local access window 0 attribute register	R/W	All zeros ²	5.2.4.4/5-9
0x0_0028	LBLAWBAR1—LBC local access window 1 base address register	R/W	All zeros	5.2.4.3/5-8
0x0_002C	LBLAWAR1—LBC local access window 1 attribute register	R/W	All zeros	5.2.4.4/5-9
0x0_0030	LBLAWBAR2—LBC local access window 2 base address register	R/W	All zeros	5.2.4.3/5-8
0x0_0034	LBLAWAR2—LBC local access window 2 attribute register	R/W	All zeros	5.2.4.4/5-9
0x0_0038	LBLAWBAR3—LBC local access window 3 base address register	R/W	All zeros	5.2.4.3/5-8
0x0_003C	LBLAWAR3—LBC local access window 3 attribute register	R/W	All zeros	5.2.4.4/5-9
0x0_0040–0x0_005C	Reserved, should be cleared	—	—	—
0x0_0060	PCILAWBAR0—PCI local access window0 base address register	R/W	All zeros ³	5.2.4.5/5-10
0x0_0064	PCILAWAR0—PCI local access window0 attribute register	R/W	All zeros ⁴	5.2.4.6/5-11
0x0_0068	PCILAWBAR1—PCI local access window1 base address register	R/W	All zeros	5.2.4.5/5-10
0x0_006C	PCILAWAR1—PCI local access window1 attribute register	R/W	All zeros	5.2.4.6/5-11
0x0_0070–0x0_009C	Reserved, should be cleared	—	—	—

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_00A0	DDRLAWBAR0—DDR local access window0 base address register	R/W	All zeros ⁵	5.2.4.7/5-12
0x0_00A4	DDRLAWAR0—DDR local access window0 attribute register	R/W	All zeros ⁶	5.2.4.8/5-13
0x0_00A8	DDRLAWBAR1—DDR local access window1 base address register	R/W	All zeros	5.2.4.7/5-12
0x0_00AC	DDRLAWAR1—DDR local access window1 attribute register	R/W	All zeros	5.2.4.8/5-13
0x0_00B0– 0x0_00FC	Reserved, should be cleared	—	—	—
0x0_0100	System general purpose register low (SGPRL)	R/W	All zeros	5.4.3.1/5-25
0x0_0104	System general purpose register high (SGPRH)	R/W	All zeros	5.4.3.2/5-25
0x0_0108	System part and revision ID register (SPRIDR)	R	All zeros	5.4.3.3/5-26
0x0_010C	Reserved, should be cleared	—	—	—
0x0_0110	System priority configuration register (SPCR)	R/W	All zeros	5.4.3.4/5-27
0x0_0114	System I/O configuration register low (SICRL)	R/W	All zeros	5.4.3.5/5-28
0x0_0118	System I/O configuration register high (SICRH)	R/W	All zeros ⁷	5.4.3.6/5-30
0x0_011C– 0x0_0124	Reserved, should be cleared	—	—	—
0x0_0128	DDR control driver register (DDRCDR)	R/W	0x0004_0000	5.4.3.8/5-33
0x0_012C	DDR debug status register (DDRDSR)	R	0x3300_0000	5.4.3.9/5-35
0x0_0130– 0x0_01FC	Reserved	—	—	—
Watchdog Timer (WDT) Registers				
0x0_0200– 0x0_0203	Reserved, should be cleared	—	—	—
0x0_0204	SWCRR—System watchdog control register	R/W	0xFFFF_0003 or 0xFFFF_0007 ⁸	5.5.4.1/5-38
0x0_0208	SWCNR—System watchdog count register	R	0x0000_FFFF	5.5.4.2/5-39
0x0_020C– 0x0_020D	Reserved, should be cleared	—	—	—
0x0_020E	SWSRR—System watchdog service register	R/W	All zeros	5.5.4.3/5-39
Real Time Clock Module Registers (RTC)				
0x0_0300	RTCNR—Real time counter control register	R/W	All zeros	5.6.5.1/5-44
0x0_0304	RTLDR—Real time counter load register	R/W	All zeros	5.6.5.2/5-45
0x0_0308	RTPSR—Real time counter prescale register	R/W	All zeros	5.6.5.3/5-46
0x0_030C	RTCTR—Real time counter register	R	All zeros	5.6.5.4/5-46
0x0_0310	RTEVR—Real time counter event register	R/W	All zeros	5.6.5.5/5-47
0x0_0314	RTALR—Real time counter alarm register	R/W	All zeros	5.6.5.6/5-47

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0318– 0x0_031F	Reserved, should be cleared	—	—	—
Periodic Interval Timer (PIT) Registers				
0x0_0400	PTCNR—Periodic interval timer control register	R/W	All zeros	5.7.5.1/5-51
0x0_0404	PTLDR—Periodic interval timer load register	R/W	All zeros	5.7.5.2/5-52
0x0_0408	PTPSR—Periodic interval timer prescale register	R/W	All zeros	5.7.5.3/5-53
0x0_040C	PTCTR—Periodic interval timer counter register	R	All zeros	5.7.5.4/5-53
0x0_0410	PTEVR—Periodic interval timer event register	R/W	All zeros	5.7.5.5/5-54
0x0_0410– 0x0_041F	Reserved, should be cleared	—	—	—
Global Timers Module 1				
0x0_0500	GTCFR1—Timer 1 and 2 global timers configuration register	R/W	All zeros	5.8.5.1/5-61
0x0_0501– 0x0_0503	Reserved, should be cleared	—	—	—
0x0_0504	GTCFR2—Timer 3 and 4 global timers configuration register	R/W	All zeros	5.8.5.1/5-61
0x0_0505– 0x0_050F	Reserved, should be cleared	—	—	—
0x0_0510	GTMDR1—Timer 1 global timers mode register	R/W	All zeros	5.8.5.2/5-65
0x0_0512	GTMDR2—Timer 2 global timers mode register			
0x0_0514	GTRFR1—Timer 1 global timers reference register	R/W	All zeros	5.8.5.3/5-66
0x0_0516	GTRFR2—Timer 2 global timers reference register			
0x0_0518	GTCPR1—Timer 1 global timers capture register	R/W	All zeros	5.8.5.4/5-66
0x0_051A	GTCPR2—Timer 2 global timers capture register			
0x0_051C	GTCNR1—Timer 1 global timers counter register	R/W	All zeros	5.8.5.5/5-67
0x0_051E	GTCNR2—Timer 2 global timers counter register			
0x0_0520	GTMDR3—Timer 3 global timers mode register	R/W	All zeros	5.8.5.2/5-65
0x0_0522	GTMDR4—Timer 4 global timers mode register			
0x0_0524	GTRFR3—Timer 3 global timers reference register	R/W	All zeros	5.8.5.3/5-66
0x0_0526	GTRFR4—Timer 4 global timers reference register			
0x0_0528	GTCPR3—Timer 3 global timers capture register	R	All zeros	5.8.5.4/5-66
0x0_052A	GTCPR4—Timer 4 global timers capture register			
0x0_052C	GTCNR3—Timer 3 global timers counter register	R/W	All zeros	5.8.5.5/5-67
0x0_052E	GTCNR4—Timer 4 global timers counter register			
0x0_0530	GTEVR1—Timer 1 global timers event register	Special	All zeros	5.8.5.6/5-67
0x0_0532	GTEVR2—Timer 2 global timers event register			
0x0_0534	GTEVR3—Timer 3 global timers event register			
0x0_0536	GTEVR4—Timer 4 global timers event register			

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0538	GTPSR1—Timer 1 global timers prescale register	R/W	0x0003	5.8.5.7/5-68
0x0_053A	GTPSR2—Timer 2 global timers prescale register			
0x0_053C	GTPSR3—Timer 3 global timers prescale register			
0x0_053E	GTPSR4—Timer 4 global timers prescale register			
Global Timers Module 2				
0x0_0600	GTCFR1—Timer 1 and 2 global timers configuration register	R/W	All zeros	5.8.5.1/5-61
0x0_0601– 0x0_0603	Reserved, should be cleared	—	—	—
0x0_0604	GTCFR2—Timer 3 and 4 global timers configuration register	R/W	All zeros	5.8.5.1/5-61
0x0_0605– 0x0_060F	Reserved, should be cleared	—	—	—
0x0_0610	GTMDR1—Timer 1 global timers mode register	R/W	All zeros	5.8.5.2/5-65
0x0_0612	GTMDR2—Timer 2 global timers mode register			
0x0_0614	GTRFR1—Timer 1 global timers reference register	R/W	All zeros	5.8.5.3/5-66
0x0_0616	GTRFR2—Timer 2 global timers reference register			
0x0_0618	GTCPR1—Timer 1 global timers capture register	R/W	All zeros	5.8.5.4/5-66
0x0_061A	GTCPR2—Timer 2 global timers capture register			
0x0_061C	GTCNR1—Timer 1 global timers counter register	R/W	All zeros	5.8.5.5/5-67
0x0_061E	GTCNR2—Timer 2 global timers counter register			
0x0_0620	GTMDR3—Timer 3 global timers mode register	R/W	All zeros	5.8.5.2/5-65
0x0_0622	GTMDR4—Timer 4 global timers mode register			
0x0_0624	GTRFR3—Timer 3 global timers reference register	R/W	All zeros	5.8.5.3/5-66
0x0_0626	GTRFR4—Timer 4 global timers reference register			
0x0_0628	GTCPR3—Timer 3 global timers capture register	R	All zeros	5.8.5.4/5-66
0x0_062A	GTCPR4—Timer 4 global timers capture register			
0x0_062C	GTCNR3—Timer 3 global timers counter register	R/W	All zeros	5.8.5.5/5-67
0x0_062E	GTCNR4—Timer 4 global timers counter register			
0x0_0630	GTEVR1—Timer 1 global timers event register	Special	All zeros	5.8.5.6/5-67
0x0_0632	GTEVR2—Timer 2 global timers event register			
0x0_0634	GTEVR3—Timer 3 global timers event register			
0x0_0636	GTEVR4—Timer 4 global timers event register			
0x0_0638	GTPSR1—Timer 1 global timers prescale register	R/W	0x0003	5.8.5.7/5-68
0x0_063A	GTPSR2—Timer 2 global timers prescale register			
0x0_063C	GTPSR3—Timer 3 global timers prescale register			
0x0_063E	GTPSR4—Timer 4 global timers prescale register			
Integrated Programmable Interrupt Controller (IPIC)				

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0700	SICFR—System global interrupt configuration register	R/W	All zeros	8.5.1/8-8
0x0_0704	SIVCR—System global interrupt vector register	R	All zeros	8.5.2/8-9
0x0_0708	SIPNR_H—System internal interrupt pending register	R	All zeros	8.5.3/8-11
0x0_070C	SIPNR_L—System internal interrupt pending register	R	All zeros	8.5.3/8-11
0x0_0710	SIPRR_A—System internal interrupt group A priority register	R/W	0x0530_9770	8.5.4/8-13
0x0_0714	Reserved, should be cleared	—	—	—
0x0_0718	Reserved, should be cleared	—	—	—
0x0_071C	SIPRR_D—System internal interrupt group D priority register	R/W	0x0530_9770	8.5.5/8-14
0x0_0720	SIMSR_H—System internal interrupt mask register	R/W	All zeros	8.5.6/8-15
0x0_0724	SIMSR_L—System internal interrupt mask register	R/W	All zeros	8.5.6/8-15
0x0_0728	SICNR—System internal interrupt control register	R/W	All zeros	8.5.7/8-16
0x0_072C	SEPNR—System external interrupt pending register	R/W	Special	8.5.8/8-18
0x0_0730	SMPRR_A—System mixed interrupt group A priority register	R/W	0x0530_9770	8.5.9/8-18
0x0_0734	SMPRR_B—System mixed interrupt group B priority register	R/W	0x0530_9770	8.5.10/8-19
0x0_0738	SEMSR—System external interrupt mask register	R/W	All zeros	8.5.11/8-20
0x0_073C	SECNR—System external interrupt control register	R/W	All zeros	8.5.12/8-21
0x0_0740	SERSR—System error status register	R/W	All zeros	8.5.13/8-22
0x0_0744	SERMR—System error mask register	R/W	—	8.5.14/8-23
0x0_0748	SERCR—System error control register	R/W	All zeros	8.5.15/8-24
0x0_074C– 0x0_074F	Reserved, should be cleared	—	—	—
0x0_0750	SIFCR_H—System internal interrupt force register	R/W	All zeros	8.5.16/8-25
0x0_0754	SIFCR_L—System internal interrupt force register	R/W	All zeros	8.5.16/8-25
0x0_0758	SEFCR—System external interrupt force register	R/W	All zeros	8.5.17/8-26
0x0_075C	SERFR—System error force register	R/W	All zeros	8.5.18/8-26
0x0_0760	SCVCR—System critical interrupt vector register	R	All zeros	8.5.19/8-27
0x0_0764	SMVCR—System management interrupt vector register	R	All zeros	8.5.20/8-27
0x0_0768– 0x0_07FF	Reserved, should be cleared	—	—	—
System Arbiter Registers				
0x0_0800	ACR—Arbiter configuration register	R/W	All zeros	6.2.1/6-2
0x0_0804	ATR—Arbiter timers register	R/W	0x00FF_00FF	6.2.2/6-4
0x0_0808	Reserved, should be cleared	R	All zeros	—
0x0_080C	AER—Arbiter event register	R/W	All zeros	6.2.3/6-5
0x0_0810	AIDR—Arbiter interrupt definition register	R/W	All zeros	6.2.4/6-6
0x0_0814	AMR—Arbiter mask register	R/W	All zeros	6.2.5/6-7

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0818	AEATR—Arbiter event attributes register	R	All zeros	6.2.6/6-8
0x0_081C	AEADR—Arbiter event address register	R	All zeros	6.2.7/6-9
0x0_0820	AERR—Arbiter event response register	R/W	All zeros	6.2.8/6-10
0x0_0824– 0x0_08FF	Reserved, should be cleared	—	—	—
Reset Module				
0x0_0900	RCWLR—Reset configuration word low register	R	All zeros	4.5.1.1/4-34
0x0_0904	RCWHR—Reset configuration word high register	R	All zeros	4.5.1.2/4-34
0x0_0908– 0x0_090C	Reserved, should be cleared	—	—	—
0x0_0910	RSR—Reset status register	R/W	All zeros	4.5.1.3/4-35
0x0_0914	RMR—Reset mode register	R/W	All zeros	4.5.1.4/4-36
0x0_0918	RPR—Reset protection register	R/W	All zeros	4.5.1.5/4-37
0x0_091C	RCR—Reset control register	R/W	All zeros	4.5.1.6/4-37
0x0_0920	RCER—Reset control enable register	R/W	All zeros	4.5.1.7/4-38
0x0_0924– 0x0_09FC	Reserved, should be cleared	—	—	—
Clock Module				
0x0_0A00	SPMR—System PLL mode register	R	All zeros	4.5.2.1/4-39
0x0_0A04	OCCR—Output clock control register	R/W	All zeros	4.5.2.2/4-40
0x0_0A08	SCCR—System clock control register	R/W	0xFFFF_FFFF	4.5.2.3/4-41
0x0_0A0C– 0x0_0AFC	Reserved, should be cleared	—	—	—
Power Management Control Module				
0x0_0B00	PMCCR—Power management controller configuration register	R/W	All zeros	5.9.2.1/5-73
0x0_0B04	PM CER—Power management controller event register	R/W	All zeros	5.9.2.2/5-74
0x0_0B08	PMCMR—Power management controller mask register	R/W	All zeros	5.9.2.3/5-75
0x0_0B0C– 0x0_0BFC	Reserved, should be cleared	—	—	—
QUICC Engine Ports Interrupt Registers				
0x0_0C00– 0x0_0C0B	Reserved	—	—	—
0x0_0C0C	CEPIER—QUICC Engine ports interrupt event register	R/W	Undefined	8.5.21/8-28
0x0_0C10	CEPIMR—QUICC Engine ports interrupt mask register	R/W	All zeros	8.5.22/8-29
0x0_0C14	CEPICR—QUICC Engine ports interrupt control register	R/W	All zeros	8.5.23/8-30
0x0_0C18– 0x0_0CFF	Reserved	—	—	—

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0D00– 0x0_0EFF	Reserved	—	—	—
Clock Control Secondary DDR				
0x0_0F00– 0x0_0F0F	Reserved, should be cleared	—	—	—
0x0_0F10	Secondary MCK enable register (MCKENR2)	R/W	0xFC00_0000	4.5.3/4-41
0x0_0F14– 0x0_0FFF	Reserved, should be cleared	—	—	—
Clock Control Primary DDR				
0x0_1000– 0x0_100F	Reserved, should be cleared	—	—	—
0x0_1010	Primary MCK enable register (MCKENR1)	R/W	0xFC00_0000	4.5.3/4-41
0x0_1014– 0x0_10FF	Reserved, should be cleared	—	—	—
Clock and DLL Control, LBC				
0x0_1100– 0x0_1104	Reserved, should be cleared.	—	—	—
0x0_1108	DLL override register (DLLOVR)	R/W	All zeros	18.4.1/18-4
0x0_110C	DLL status register (DLLSR)	R	All zeros	18.4.2/18-5
0x0_1110	DLL clock register (DLLCK)	R/W	0xE000_0000	18.4.3/18-5
0x0_1114– 0x0_13FF	Reserved, should be cleared	—	—	—
QUICC Engine Parallel I/O Ports Registers				
0x0_1400	CPODRA—QUICC Engine port A open drain register	R/W	All zeros	3.4.2.1/3-21
0x0_1404	CPDATA—QUICC Engine port A data register	R/W	All zeros	3.4.2.2/3-22
0x0_1408	CPDIR1A—QUICC Engine port A direction register 1	R/W	All zeros	3.4.2.3/3-23
0x0_140C	CPDIR2A—QUICC Engine port A direction register 2	R/W	All zeros	3.4.2.3/3-23
0x0_1410	CPPAR1A—QUICC Engine port A pin assignment register 1	R/W	All zeros	3.4.2.4/3-23
0x0_1414	CPPAR2A—QUICC Engine port A pin assignment register 2	R/W	All zeros	3.4.2.4/3-23
0x0_1418	CPODRB—QUICC Engine port B open drain register	R/W	All zeros	3.4.2.1/3-21
0x0_141C	CPDATB—QUICC Engine port B data register	R/W	All zeros	3.4.2.2/3-22
0x0_1420	CPDIR1B—QUICC Engine port B direction register 1	R/W	All zeros	3.4.2.3/3-23
0x0_1424	CPDIR2B—QUICC Engine port B direction register 2	R/W	All zeros	3.4.2.3/3-23
0x0_1428	CPPAR1B—QUICC Engine port B pin assignment register 1	R/W	All zeros	3.4.2.4/3-23
0x0_142C	CPPAR2B—QUICC Engine port B pin assignment register 2	R/W	All zeros	3.4.2.4/3-23
0x0_1430	CPODRC—QUICC Engine port C open drain register	R/W	All zeros	3.4.2.1/3-21
0x0_1434	CPDATC—QUICC Engine port C data register	R/W	All zeros	3.4.2.2/3-22
0x0_1438	CPDIR1C—QUICC Engine port C direction register 1	R/W	All zeros	3.4.2.3/3-23

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_143C	CPDIR2C—QUICC Engine port C direction register 2	R/W	All zeros	3.4.2.3/3-23
0x0_1440	CPPAR1C—QUICC Engine port C pin assignment register 1	R/W	All zeros	3.4.2.4/3-23
0x0_1444	CPPAR2C—QUICC Engine port C pin assignment register 2	R/W	All zeros	3.4.2.4/3-23
0x0_1448	CPODRD—QUICC Engine port D open drain register	R/W	All zeros	3.4.2.1/3-21
0x0_144C	CPDATD—QUICC Engine port D data register	R/W	All zeros	3.4.2.2/3-22
0x0_1450	CPDIR1D—QUICC Engine port D direction register 1	R/W	All zeros	3.4.2.3/3-23
0x0_1454	CPDIR2D—QUICC Engine port D direction register 2	R/W	All zeros	3.4.2.3/3-23
0x0_1458	CPPAR1D—QUICC Engine port D pin assignment register 1	R/W	All zeros	3.4.2.4/3-23
0x0_145C	CPPAR2D—QUICC Engine port D pin assignment register 2	R/W	All zeros	3.4.2.4/3-23
0x0_1460	CPODRE—QUICC Engine port E open drain register	R/W	All zeros	3.4.2.1/3-21
0x0_1464	CPDATE—QUICC Engine port E data register	R/W	All zeros	3.4.2.2/3-22
0x0_1468	CPDIR1E—QUICC Engine port E direction register 1	R/W	All zeros	3.4.2.3/3-23
0x0_146C	CPDIR2E—QUICC Engine port E direction register 2	R/W	All zeros	3.4.2.3/3-23
0x0_1470	CPPAR1E—QUICC Engine port E pin assignment register 1	R/W	All zeros	3.4.2.4/3-23
0x0_1474	CPPAR2E—QUICC Engine port E pin assignment register 2	R/W	All zeros	3.4.2.4/3-23
0x0_1478	CPODRF—QUICC Engine port F open drain register	R/W	All zeros	3.4.2.1/3-21
0x0_147C	CPDATF—QUICC Engine port F data register	R/W	All zeros	3.4.2.2/3-22
0x0_1480	CPDIR1F—QUICC Engine port F direction register 1	R/W	All zeros	3.4.2.3/3-23
0x0_1484	CPDIR2F—QUICC Engine port F direction register 2	R/W	All zeros	3.4.2.3/3-23
0x0_1488	CPPAR1F—QUICC Engine port F pin assignment register 1	R/W	All zeros	3.4.2.4/3-23
0x0_148C	CPPAR2F—QUICC Engine port F pin assignment register 2	R/W	All zeros	3.4.2.4/3-23
0x0_1490	CPODRG—QUICC Engine port G open drain register	R/W	All zeros	3.4.2.1/3-21
0x0_1494	CPDATG—QUICC Engine port G data register	R/W	All zeros	3.4.2.2/3-22
0x0_1498	CPDIR1G—QUICC Engine port G direction register 1	R/W	All zeros	3.4.2.3/3-23
0x0_149C	CPDIR2G—QUICC Engine port G direction register 2	R/W	All zeros	3.4.2.3/3-23
0x0_14A0	CPPAR1G—QUICC Engine port G pin assignment register 1	R/W	All zeros	3.4.2.4/3-23
0x0_14A4	CPPAR2G—QUICC Engine port G pin assignment register 2	R/W	All zeros	3.4.2.4/3-23
0x0_14A8	CPOH1—QE Port output hold register 1	R/W	0x5555_5555	3.4.2.6/3-25
0x0_14AC	CPOH2—QE Port output hold register 2	R/W	0x5555_5555	3.4.2.6/3-25
0x0_14B0– 0x0_14B7	Reserved	—	—	3.4.2.6/3-25
0x0_14B8	CPCE1R—Communication peripherals to QUICC Engine mux register 1	R/W	All zeros	3.4.2.5/3-24
0x0_14BC	CPCE2R—Communication peripherals to QUICC Engine mux register 2	R/W	All zeros	3.4.2.5/3-24
0x0_14C0	CPCE3R—Communication peripherals to QUICC Engine mux register 3	R/W	All zeros	3.4.2.5/3-24

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_14C4	CPCE4R—Communication peripherals to QUICC Engine mux register 4	R/W	All zeros	3.4.2.5/3-24
0x0_14C5–0x0_17FF	Reserved	—	—	3.4.2.6/3-25
QUICC Engine Secondary Bus Access Windows Registers				
0x0_1800	LBMCSAR—Local bus memory controller start address register	R/W	All zeros	5.3.2.1/5-18
0x0_1804	SDMCSAR—Secondary DDR memory controller start address register	R/W	All zeros	5.3.2.2/5-19
0x0_1808–0x0_183C	Reserved	—	—	—
0x0_1840	LBMCEAR—Local bus memory controller end address register	R/W	All zeros	5.3.2.3/5-19
0x0_1844	SDMCEAR—Secondary DDR memory controller end address register	R/W	All zeros	5.3.2.4/5-20
0x0_1848–0x0_187C	Reserved	—	—	—
0x0_1880	LBMCAR—Local bus memory controller attributes register	R/W	All zeros	5.3.2.5/5-20
0x0_1884	SDMCAR—Secondary DDR memory controller attributes register	R/W	All zeros	5.3.2.6/5-21
0x0_1888–0x0_1BFF	Reserved	—	—	—
0x0_1C00–0x0_1FFF	Reserved	—	—	—
DDR Memory Controller Memory Map				
0x0_2000	CS0_BNDS—Chip select memory bounds	R/W	All zeros	9.4.1.1/9-12
0x0_2008	CS1_BNDS—Chip select 1 memory bounds	R/W	All zeros	
0x0_2010	CS2_BNDS—Chip select 2 memory bounds	R/W	All zeros	
0x0_2018	CS3_BNDS—Chip select 3 memory bounds	R/W	All zeros	
0x0_2020–0x0_207F	Reserved	—	—	—
0x0_2080	CS0_CONFIG—Chip select 0 configuration	R/W	All zeros	9.4.1.2/9-12
0x0_2084	CS1_CONFIG—Chip select 1 configuration	R/W	All zeros	
0x0_2088	CS2_CONFIG—Chip select 2 configuration	R/W	All zeros	
0x0_208C	CS3_CONFIG—Chip select 3 configuration	R/W	All zeros	
0x0_2090–0x0_20FF	Reserved	—	—	—
0x0_2100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	All zeros	9.4.1.3/9-14
0x0_2104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	9.4.1.4/9-15
0x0_2108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	All zeros	9.4.1.5/9-17
0x0_210C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	All zeros	9.4.1.6/9-19
0x0_2110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.7/9-21

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_2114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	All zeros	9.4.1.8/9-23
0x0_2118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	All zeros	9.4.1.9/9-25
0x0_211C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	All zeros	9.4.1.10/9-26
0x0_2120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	All zeros	9.4.1.11/9-26
0x0_2124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	All zeros	9.4.1.12/9-29
0x0_2128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	All zeros	9.4.1.13/9-29
0x0_2130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	9.4.1.14/9-30
0x0_2134– 0x0_2144	Reserved	—	—	—
0x0_2148	DDR_INIT_ADDRESS—DDR training initialization address	R/W	All zeros	9.4.1.15/9-30
0x0_214C– 0x0_2BF4	Reserved	—	—	—
0x0_2BF8	DDR_IP_REV1—DDR IP block revision 1	R	0x0002_0200	9.4.1.16/9-31
0x0_2BFC	DDR_IP_REV2—DDR IP block revision 2	R	All zeros	9.4.1.17/9-31
0x0_2E00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	All zeros	9.4.1.18/9-32
0x0_2E04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	All zeros	9.4.1.19/9-32
0x0_2E08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	All zeros	9.4.1.20/9-33
0x0_2E20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	All zeros	9.4.1.21/9-33
0x0_2E24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	All zeros	9.4.1.22/9-34
0x0_2E28	CAPTURE_ECC—Memory data path read capture ECC	R/W	All zeros	9.4.1.23/9-34
0x0_2E40	ERR_DETECT—Memory error detect	w1c	All zeros	9.4.1.24/9-34
0x0_2E44	ERR_DISABLE—Memory error disable	R/W	All zeros	9.4.1.25/9-35
0x0_2E48	ERR_INT_EN—Memory error interrupt enable	R/W	All zeros	9.4.1.26/9-36
0x0_2E4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	All zeros	9.4.1.27/9-37
0x0_2E50	CAPTURE_ADDRESS—Memory error address capture	R/W	All zeros	9.4.1.28/9-38
0x0_2E58	ERR_SBE—Single-Bit ECC memory error management	R/W	All zeros	9.4.1.29/9-39
0x0_2E5C– 0x0_2FFF	Reserved	—	—	—
I²C1 Controller				
0x0_3000	I2C1ADR—I ² C1 address register	R/W	All zeros	15.3.1.1/15-5
0x0_3004	I2C1FDR—I ² C1 frequency divider register	R/W	All zeros	15.3.1.2/15-6
0x0_3008	I2C1CR—I ² C1 control register	R/W	All zeros	15.3.1.3/15-7
0x0_300C	I2C1SR—I ² C1 status register	R/W	0x81	15.3.1.4/15-8
0x0_3010	I2C1DR—I ² C1 data register	R/W	All zeros	15.3.1.5/15-9
0x0_3014	I2C1DFSRR—I ² C1 digital filter sampling rate register	R/W	0x0001_0000	15.3.1.6/15-10

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_3018–0x0_30FF	Reserved, should be cleared	—	—	—
I²C2 Controller				
0x0_3100	I2C2ADR—I ² C2 address register	R/W	All zeros	15.3.1.1/15-5
0x0_3104	I2C2FDR—I ² C2 frequency divider register	R/W	All zeros	15.3.1.2/15-6
0x0_3108	I2C2CR—I ² C2 control register	R/W	All zeros	15.3.1.3/15-7
0x0_310C	I2C2SR—I ² C2 status register	R/W	0x81	15.3.1.4/15-8
0x0_3110	I2C2DR—I ² C2 data register	R/W	All zeros	15.3.1.5/15-9
0x0_3114	I2C2DFSRR—I ² C2 digital filter sampling rate register	R/W	0001_0000	15.3.1.6/15-10
0x0_311C–0x0_31FF	Reserved, should be cleared	—	—	—
0x0_3200–0x0_3FFF	Reserved, should be cleared	—	—	—
DUART				
0x0_4000–0x0_44FF	Reserved, should be cleared	—	—	—
0x0_4500	URBR—ULCR[DLAB] = 0 UART1 receiver buffer register	R	All zeros	16.3.1.1/16-6
0x0_4500	UTHR—ULCR[DLAB] = 0 UART1 transmitter holding register	W	All zeros	16.3.1.2/16-6
0x0_4500	UDLB—ULCR[DLAB] = 1 UART1 divisor least significant byte register	R/W	All zeros	16.3.1.3/16-7
0x0_4501	UIER—ULCR[DLAB] = 0 UART1 interrupt enable register	R/W	All zeros	16.3.1.4/16-8
0x0_4501	UDMB—ULCR[DLAB] = 1 UART1 divisor most significant byte register	R/W	All zeros	16.3.1.3/16-7
0x0_4502	UIIR—ULCR[DLAB] = 0 UART1 interrupt ID register	R	0x01	16.3.1.5/16-9
0x0_4502	UFCR—ULCR[DLAB] = 0 UART1 FIFO control register	W	All zeros	16.3.1.6/16-10
0x0_4502	UAFR—ULCR[DLAB] = 1 UART1 alternate function register	R/W	All zeros	16.3.1.7/16-11
0x0_4503	ULCR—ULCR[DLAB] = x UART1 line control register	R/W	All zeros	16.3.1.8/16-12
0x0_4504	UMCR—ULCR[DLAB] = x UART1 MODEM control register	R/W	All zeros	16.3.1.9/16-14
0x0_4505	ULSR—ULCR[DLAB] = x UART1 line status register	R	0x60	16.3.1.10/16-15
0x0_4506	UMSR—ULCR[DLAB] = x UART1 MODEM status register	R	All zeros	16.3.1.11/16-16
0x0_4507	USCR—ULCR[DLAB] = x UART1 scratch register	R/W	All zeros	16.3.1.12/16-17
0x0_4510	UDSR—ULCR[DLAB] = x UART1 DMA status register	R	0x01	16.3.1.13/16-17
0x0_4600	URBR—ULCR[DLAB] = 0 UART2 receiver buffer register	R	All zeros	16.3.1.1/16-6
0x0_4600	UTHR—ULCR[DLAB] = 0 UART2 transmitter holding register	W	All zeros	16.3.1.2/16-6
0x0_4600	UDLB—ULCR[DLAB] = 1 UART2 divisor least significant byte register	R/W	All zeros	16.3.1.3/16-7
0x0_4601	UIER—ULCR[DLAB] = 0 UART2 interrupt enable register	R/W	All zeros	16.3.1.4/16-8

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_4601	UDMB_ULCR[DLAB] = 1 UART2 divisor most significant byte register	R/W	All zeros	16.3.1.3/16-7
0x0_4602	UIIR—ULCR[DLAB] = 0 UART2 interrupt ID register	R	0x01	16.3.1.5/16-9
0x0_4602	UFCR—ULCR[DLAB] = 0 UART2 FIFO control register	W	All zeros	16.3.1.6/16-10
0x0_4602	UAFR—ULCR[DLAB] = 1 UART2 alternate function register	R/W	All zeros	16.3.1.7/16-11
0x0_4603	ULCR—ULCR[DLAB] = x UART2 line control register	R/W	All zeros	16.3.1.8/16-12
0x0_4604	UMCR—ULCR[DLAB] = x UART2 MODEM control register	R/W	All zeros	16.3.1.9/16-14
0x0_4605	ULSR—ULCR[DLAB] = x UART2 line status register	R	0x60	16.3.1.10/16-15
0x0_4606	UMSR—ULCR[DLAB] = x UART2 MODEM status register	R	All zeros	16.3.1.11/16-16
0x0_4607	USCR—ULCR[DLAB] = x UART2 scratch register	R/W	All zeros	16.3.1.12/16-17
0x0_4610	UDSR—ULCR[DLAB] = x UART2 DMA status register	R	0x01	16.3.1.13/16-17
0x0_4700– 0x0_4FFF	Reserved, should be cleared	—	—	—
Local Bus Controller (LBC) Registers				
0x0_5000	BR0—Base register 0 Note: Port size for BR0 is configured from the value of RCWH[ROMLOC] which is loaded during reset, hence 'RR' is either 0x08, 0x10, or 0x18.	R/W	0x0000_RR01	10.3.1.1/10-11
0x0_5008	BR1—Base register 1		All zeros	
0x0_5010	BR2—Base register 2			
0x0_5018	BR3—Base register 3			
0x0_5020	BR4—Base register 4			
0x0_5028	BR5—Base register 5			
0x0_5030	BR6—Base register 6			
0x0_5038	BR7—Base register 7			
0x0_5004	OR0—Options register 0	R/W	0x0000_0FF7	10.3.1.2/10-12
0x0_500C	OR1—Options register 1		All zeros	
0x0_5014	OR2—Options register 2			
0x0_501C	OR3—Options register 3			
0x0_5024	OR4—Options register 4			
0x0_502C	OR5—Options register 5			
0x0_5034	OR6—Options register 6			
0x0_503C	OR7—Options register 7			
0x0_5068	MAR—UPM address register	R/W	All zeros	10.3.1.3/10-18
0x0_5070	MAMR—UPMA mode register	R/W	All zeros	10.3.1.4/10-19
0x0_5074	MBMR—UPMB mode register	R/W	All zeros	10.3.1.4/10-19
0x0_5078	MCMR—UPMC mode register	R/W	All zeros	10.3.1.4/10-19

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_5084	MRTPR—Memory refresh timer prescaler register	R/W	All zeros	10.3.1.5/10-21
0x0_5088	MDR—UPM data register	R/W	All zeros	10.3.1.6/10-22
0x0_5094	LSDMR—SDRAM mode register	R/W	All zeros	10.3.1.7/10-22
0x0_50A0	LURT—UPM refresh timer	R/W	All zeros	10.3.1.8/10-24
0x0_50A4	LSRT—SDRAM refresh timer	R/W	All zeros	10.3.1.9/10-25
0x0_50B0	LTESR—Transfer error status register	Read/ bit-reset	All zeros	10.3.1.10/10-26
0x0_50B4	LTEDR—Transfer error check disable register	R/W	All zeros	10.3.1.11/10-27
0x0_50B8	LTEIR—Transfer error interrupt enable register	R/W	All zeros	10.3.1.12/10-28
0x0_50BC	LTEATR—Transfer error attributes register	R/W	All zeros	10.3.1.13/10-29
0x0_50C0	LTEAR—Transfer error address register	R/W	All zeros	10.3.1.14/10-30
0x0_50D0	LBCR—Local bus configuration register	R/W	All zeros	10.3.1.15/10-30
0x0_50D4	LCRR—Clock ratio register	R/W	0x8000_0008	10.3.1.16/10-31
0x0_6000– 0x0_6FFF	Reserved, should be cleared	—	—	—
DMA Registers				
0x0_8030	OMISR—Outbound message interrupt status register	Special	All zeros	12.4.1/12-4
0x0_8034	OMIMR—Outbound message interrupt mask register	R/W	All zeros	12.4.2/12-5
0x0_8050	IMR0—Inbound message register 0	R/W	All zeros	12.4.3/12-6
0x0_8054	IMR1—Inbound message register 1	R/W	All zeros	12.4.3/12-6
0x0_8058	OMR0—Outbound message register 0	R/W	All zeros	12.4.4/12-6
0x0_805C	OMR1—Outbound message register 1	R/W	All zeros	12.4.4/12-6
0x0_8060	ODR—Outbound doorbell register	R/W	All zeros	12.4.5/12-7
0x0_8068	IDR—Inbound doorbell register	R/W	All zeros	12.4.5/12-7
0x0_8080	IMISR—Inbound message interrupt status register	R/W	All zeros	12.4.6/12-8
0x0_8084	IMIMR—Inbound message interrupt mask register	R/W	All zeros	12.4.7/12-9
0x0_8100	DMAMR0—DMA 0 mode register	R/W	All zeros	12.4.8.1/12-10
0x0_8104	DMASR0—DMA 0 status register	R/W	All zeros	12.4.8.2/12-12
0x0_8108	DMACDAR0—DMA 0 current descriptor address register	R/W	All zeros	12.4.8.3/12-13
0x0_8110	DMASAR0—DMA 0 source address register	R/W	All zeros	12.4.8.4/12-14
0x0_8118	DMADAR0—DMA 0 destination address register	R/W	All zeros	12.4.8.5/12-14
0x0_8120	DMABCR0—DMA 0 byte count register	R/W	All zeros	12.4.8.6/12-15
0x0_8124	DMANDAR0—DMA 0 next descriptor address register	R/W	All zeros	12.4.8.7/12-15
0x0_8180	DMAMR1—DMA 1 mode register	R/W	All zeros	12.4.8.1/12-10
0x0_8184	DMASR1—DMA 1 status register	R/W	All zeros	12.4.8.2/12-12
0x0_8188	DMACDAR1—DMA 1 current descriptor address register	R/W	All zeros	12.4.8.3/12-13

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8190	DMASAR1—DMA 1 source address register	R/W	All zeros	12.4.8.4/12-14
0x0_8198	DMADAR1—DMA 1 destination address register	R/W	All zeros	12.4.8.5/12-14
0x0_81A0	DMABCR1—DMA 1 byte count register	R/W	All zeros	12.4.8.6/12-15
0x0_81A4	DMANDAR1—DMA 1 next descriptor address register	R/W	All zeros	12.4.8.7/12-15
0x0_8200	DMAMR2—DMA 2 mode register	R/W	All zeros	12.4.8.1/12-10
0x0_8204	DMASR2—DMA 2 status register	R/W	All zeros	12.4.8.2/12-12
0x0_8208	DMACDAR2—DMA 2 current descriptor address register	R/W	All zeros	12.4.8.3/12-13
0x0_8210	DMASAR2—DMA 2 source address register	R/W	All zeros	12.4.8.4/12-14
0x0_8218	DMADAR2—DMA 2 destination address register	R/W	All zeros	12.4.8.5/12-14
0x0_8220	DMABCR2—DMA 2 byte count register	R/W	All zeros	12.4.8.6/12-15
0x0_8224	DMANDAR2—DMA 2 next descriptor address register	R/W	All zeros	12.4.8.7/12-15
0x0_8280	DMAMR3—DMA 3 mode register	R/W	All zeros	12.4.8.1/12-10
0x0_8284	DMASR3—DMA 3 status register	R/W	All zeros	12.4.8.2/12-12
0x0_8288	DMACDAR3—DMA 3 current descriptor address register	R/W	All zeros	12.4.8.3/12-13
0x0_8290	DMASAR3—DMA 3 source address register	R/W	All zeros	12.4.8.4/12-14
0x0_8298	DMADAR3—DMA 3 destination address register	R/W	All zeros	12.4.8.5/12-14
0x0_82A0	DMABCR3—DMA 3 byte count register	R/W	All zeros	12.4.8.6/12-15
0x0_82A4	DMANDAR3—DMA 3 next descriptor address register	R/W	All zeros	12.4.8.7/12-15
0x0_82A8	DMAGSR—DMA general status register	R	All zeros	12.4.8.8/12-16
0x0_82B0– 0x0_82FF	Reserved, should be cleared	—	—	—
PCI Software Configuration Registers				
0x0_8300	PCI CONFIG_ADDRESS	W	—	13.3.1.1/13-13
0x0_8304	PCI CONFIG_DATA	R/W	—	13.3.1.2/13-14
0x0_8308	PCI INT_ACK	R	—	13.3.1.3/13-15
0x0_830C– 0x0_837F	Reserved, should be cleared	—	—	—
0x0_8380– 0x0_83FF	Reserved	—	—	—
Sequencer (IOS)				
0x0_8400	POTAR0—PCI outbound translation address register 0	R/W	All zeros	11.4.1/11-3
0x0_8408	POBAR0—PCI outbound base address register 0	R/W	All zeros	11.4.2/11-3
0x0_8410	POCMR0—PCI outbound comparison mask register 0	R/W	All zeros	11.4.3/11-4
0x0_8418	POTAR1—PCI outbound translation address register 1	R/W	All zeros	11.4.1/11-3
0x0_8420	POBAR1—PCI outbound base address register 1	R/W	All zeros	11.4.2/11-3
0x0_8428	POCMR1—PCI outbound comparison mask register 1	R/W	All zeros	11.4.3/11-4
0x0_8430	POTAR2—PCI outbound translation address register 2	R/W	All zeros	11.4.1/11-3

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8438	POBAR2—PCI outbound base address register 2	R/W	All zeros	11.4.2/11-3
0x0_8440	POCMR2—PCI outbound comparison mask register 2	R/W	All zeros	11.4.3/11-4
0x0_8448	POTAR3—PCI outbound translation address register 3	R/W	All zeros	11.4.1/11-3
0x0_8450	POBAR3—PCI outbound base address register 3	R/W	All zeros	11.4.2/11-3
0x0_8458	POCMR3—PCI outbound comparison mask register 3	R/W	All zeros	11.4.3/11-4
0x0_8460	POTAR4—PCI outbound translation address register 4	R/W	All zeros	11.4.1/11-3
0x0_8468	POBAR4—PCI outbound base address register 4	R/W	All zeros	11.4.2/11-3
0x0_8470	POCMR4—PCI outbound comparison mask register 4	R/W	All zeros	11.4.3/11-4
0x0_8478	POTAR5—PCI outbound translation address register 5	R/W	All zeros	11.4.1/11-3
0x0_8480	POBAR5—PCI outbound base address register 5	R/W	All zeros	11.4.2/11-3
0x0_8488	POCMR5—PCI outbound comparison mask register 5	R/W	All zeros	11.4.3/11-4
0x0_84F0	PMCR—Power management control register	R/W	All zeros	11.4.4/11-5
0x0_84F8	DTCR—Discard timer control register	R/W	All zeros	11.4.5/11-6
PCI Error Management Registers				
0x0_8500	PCI_ESR—PCI error status register	R / w1c	All zeros	13.3.2.1/13-15
0x0_8504	PCI_ECDR—PCI error capture disable register	R/W	All zeros	13.3.2.2/13-16
0x0_8508	PCI_EER—PCI error enable register	R/W	All zeros	13.3.2.3/13-17
0x0_850C	PCI_EATCR—PCI error attributes capture register	R/W	All zeros	13.3.2.4/13-18
0x0_8510	PCI_EACR—PCI error address capture register	R	All zeros	13.3.2.5/13-19
0x0_8514	PCI_EEACR—PCI error extended address capture register	R	All zeros	13.3.2.6/13-20
0x0_8518	PCI_EDCR—PCI error data capture register	R	All zeros	13.3.2.7/13-20
0x0_851C	Reserved	—	—	—
PCI Control and Status Registers				
0x0_8520	PCI_GCR—PCI general control register	R/W	All zeros	13.3.2.8/13-20
0x0_8524	PCI_ECR—PCI error control register	R/W	All zeros	13.3.2.9/13-21
0x0_8528	PCI_GSR—PCI general status register	R	All zeros	13.3.2.10/13-22
PCI Inbound ATU Registers				
0x0_8538	PITAR2—PCI inbound translation address register 2	R/W	All zeros	13.3.2.11/13-22
0x0_853C	Reserved, should be cleared	—	—	—
0x0_8540	PIBAR2—PCI inbound base address register 2	R/W	All zeros	13.3.2.12/13-23
0x0_8544	PIEBAR2—PCI inbound extended base address register 2	R/W	All zeros	13.3.2.13/13-24
0x0_8548	PIWAR2—PCI inbound window attributes register 2	R/W	All zeros	13.3.2.14/13-24
0x0_8550	PITAR1—PCI inbound translation address register 1	R/W	All zeros	13.3.2.11/13-22
0x0_8554	Reserved, should be cleared	—	—	—
0x0_8558	PIBAR1—PCI inbound base address register 1	R/W	All zeros	13.3.2.12/13-23
0x0_855C	PIEBAR1—PCI inbound extended base address register 1	R/W	All zeros	13.3.2.13/13-24

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8560	PIWAR1—PCI inbound window attributes register 1	R/W	All zeros	13.3.2.14/13-24
0x0_8568	PITAR0—PCI inbound translation address register 0	R/W	All zeros	13.3.2.11/13-22
0x0_856C	Reserved, should be cleared	—	—	—
0x0_8570	PIBAR0—PCI inbound base address register 0	R/W	All zeros	13.3.2.12/13-23
0x0_8578	PIWAR0—PCI inbound window attributes register 0	R/W	All zeros	13.3.2.13/13-24
0x0_857C– 0x0_CFFF	Reserved	—	—	—
Secondary DDR Memory Controller Memory Map				
0x0_D000	CS0_BNDS—Chip select memory bounds	R/W	All zeros	9.4.1.1/9-12
0x0_D008	CS1_BNDS—Chip select 1 memory bounds	R/W	All zeros	
0x0_D010– 0x0_D028	Reserved	—	—	—
0x0_D080	CS0_CONFIG—Chip select 0 configuration	R/W	All zeros	9.4.1.2/9-12
0x0_D084	CS1_CONFIG—Chip select 1 configuration	R/W	All zeros	
0x0_D088– 0x0_D0FF	Reserved	—	—	—
0x0_D100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	All zeros	9.4.1.3/9-14
0x0_D104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	9.4.1.4/9-15
0x0_D108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	All zeros	9.4.1.5/9-17
0x0_D10C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	All zeros	9.4.1.6/9-19
0x0_D110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.7/9-21
0x0_D114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	All zeros	9.4.1.8/9-23
0x0_D118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	All zeros	9.4.1.9/9-25
0x0_D11C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	All zeros	9.4.1.10/9-26
0x0_D120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	All zeros	9.4.1.11/9-26
0x0_D124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	All zeros	9.4.1.12/9-29
0x0_D128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	All zeros	9.4.1.13/9-29
0x0_D130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	9.4.1.14/9-30
0x0_D140	Reserved	—	—	—
0x0_D148	DDR_INIT_ADDRESS—DDR training initialization address	R/W	All zeros	9.4.1.15/9-30
0x0_DBF8	DDR_IP_REV1—DDR IP block revision 1	R	0x0002_0200	9.4.1.16/9-31
0x0_DBFC	DDR_IP_REV2—DDR IP block revision 2	R	All zeros	9.4.1.17/9-31
0x0_DE00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	All zeros	9.4.1.18/9-32
0x0_DE04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	All zeros	9.4.1.19/9-32
0x0_DE08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	All zeros	9.4.1.20/9-33

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_DE20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	All zeros	9.4.1.21/9-33
0x0_DE24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	All zeros	9.4.1.22/9-34
0x0_DE28	CAPTURE_ECC—Memory data path read capture ECC	R/W	All zeros	9.4.1.23/9-34
0x0_DE40	ERR_DETECT—Memory error detect	w1c	All zeros	9.4.1.24/9-34
0x0_DE44	ERR_DISABLE—Memory error disable	R/W	All zeros	9.4.1.25/9-35
0x0_DE48	ERR_INT_EN—Memory error interrupt enable	R/W	All zeros	9.4.1.26/9-36
0x0_DE4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	All zeros	9.4.1.27/9-37
0x0_DE50	CAPTURE_ADDRESS—Memory error address capture	R/W	All zeros	9.4.1.28/9-38
0x0_DE58	ERR_SBE—Single-Bit ECC memory error management	R/W	All zeros	9.4.1.29/9-39
0x0_DE5C– 0x0_DFFF	Reserved	—	—	—
0x0_E000– 0x2_5FFF	Reserved	—	—	—
0x2_4000– 0x2_400C	Reserved, should be cleared	—	—	—
Security Engine Address Map Registers				
Controller Registers				
0x3_0000– 0x3_0FFF	Reserved, should be cleared	—	—	—
0x3_1008	IMR—Interrupt mask register	R/W	All zeros	14.7.2.1/14-94
0x3_1010	ISR—Interrupt status register	R	All zeros	14.7.2.2/14-96
0x3_1018	ICR—Interrupt clear register	W	All zeros	14.7.2.3/14-96
0x3_1020	ID—Identification register	R	All zeros _0000_0040	14.7.2.4/14-98
0x3_1028	EUASR—EU assignment status register	R	0xF0F0_F0F0 _00FF_F0F0	14.7.2/14-93
0x3_1030	MCR—Master control register	R/W	All zeros	14.7.2.6/14-99
Channel 1				
0x3_1108	CCCR1—Crypto-channel 1 configuration register	R/W	All zeros	14.6.1.1/14-82
0x3_1110	CCPSR1—Crypto-channel 1 pointer status register	R	All zeros	14.6.1.2/14-85
0x3_1140	CDPR1—Crypto-channel 1 current descriptor pointer register	R	All zeros	14.6.1.3/14-90
0x3_1148	FF1—Crypto-channel 1 fetch FIFO address register	W	All zeros	14.6.1.4/14-90
0x3_1180– 0x3_11BF	DB _n —Crypto-channel 1 descriptor buffers [0–7]	R	All zeros	14.6.1.5/14-91
Channel 2				
0x3_1208	CCCR2—Crypto-channel 2 configuration register	R/W	All zeros	14.6.1.1/14-82
0x3_1210	CCPSR2—Crypto-channel 2 pointer status register	R	All zeros	14.6.1.2/14-85

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_1240	CDPR2—Crypto-channel 2 current descriptor pointer register	R	All zeros	14.6.1.3/14-90
0x3_1248	FF2—Crypto-channel 2 fetch FIFO address register	W	All zeros	14.6.1.4/14-90
0x3_1280– 0x3_12BF	DBn—Crypto-channel 2 descriptor buffers[0–7]	R	All zeros	14.6.1.5/14-91
Channel 3				
0x3_1308	CCCR3—Crypto-channel 3 configuration register	R/W	All zeros	14.6.1.1/14-82
0x3_1310	CCPSR3—Crypto-channel 3 pointer status register	R	All zeros	14.6.1.2/14-85
0x3_1340	CDPR3—Crypto-channel 3 current descriptor pointer register	R	All zeros	14.6.1.3/14-90
0x3_1348	FF3—Crypto-channel 3 fetch FIFO address register	W	All zeros	14.6.1.4/14-90
0x3_1380– 0x3_13BF	DBn—Crypto-channel 3 descriptor buffers[0–7]	R	All zeros	14.6.1.5/14-91
Channel 4				
0x3_1408	CCCR4—Crypto-channel 4 configuration register	R/W	All zeros	14.6.1.1/14-82
0x3_1410	CCPSR4—Crypto-channel 4 pointer status register	R	All zeros	14.6.1.2/14-85
0x3_1440	CDPR4—Crypto-channel 4 current descriptor pointer register	R	All zeros	14.6.1.3/14-90
0x3_1448	FF4—Crypto-channel 4 fetch FIFO address register	W	All zeros	14.6.1.4/14-90
0x3_1480– 0x3_14BF	DBn—Crypto-channel 4 descriptor buffers[0–7]	R	All zeros	14.6.1.5/14-91
Data Encryption Standard Execution Unit (DEU)				
0x3_2000	DEUMR—DEU mode register	R/W	All zeros	14.5.2.1/14-35
0x3_2008	DEUKSR—DEU key size register	R/W	All zeros	14.5.2.2/14-36
0x3_2010	DEUDSR—DEU data size register	R/W	All zeros	14.5.2.3/14-36
0x3_2018	DEURCR—DEU reset control register	R/W	All zeros	14.5.2.4/14-37
0x3_2028	DEUSR—DEU status register	R	All zeros	14.5.2.5/14-37
0x3_2030	DEUISR—DEU interrupt status register	R	All zeros	14.5.2.6/14-38
0x3_2038	DEUICR—DEU interrupt control register	R/W	All zeros	14.5.2.7/14-40
0x3_2050	DEUEUG—DEU EU-Go register	W	All zeros	14.5.2.8/14-41
0x3_2100	DEUIV—DEU initialization vector register	R/W	All zeros	14.5.2.9/14-42
0x3_2400	DEUK1—DEU key 1 register	W	—	14.5.2.10/14-42
0x3_2408	DEUK2—DEU key 2 register	W	—	14.5.2.10/14-42
0x3_2410	DEUK3—DEU key 3 register	W	—	14.5.2.10/14-42
0x3_2800– 0x3_2FFF	DEU FIFO	R/W	All zeros	14.5.2.11/14-42
Advanced Encryption Standard Execution Unit (AESU)				
0x3_4000	AESUMR—AESU mode register	R/W	All zeros	14.5.6.1/14-68
0x3_4008	AESUKSR—AESU key size register	R/W	All zeros	14.5.6.2/14-71

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_4010	AESUDSR—AESU data size register	R/W	All zeros	14.5.6.3/14-71
0x3_4018	AESURCR—AESU reset control register	R/W	All zeros	14.5.6.4/14-72
0x3_4028	AESUSR—AESU status register	R	All zeros	14.5.6.5/14-73
0x3_4030	AESUISR—AESU interrupt status register	R	All zeros	14.5.6.6/14-74
0x3_4038	AESUICR—AESU interrupt control register	R/W	All zeros	14.5.6.7/14-75
0x3_4050	AESUEMR—AESU end-of-message register	W	All zeros	14.5.6.8/14-76
0x3_4100	AESU context memory registers	R/W	All zeros	14.5.6.9/14-77
0x3_4400– 0x3_4408	AESU key memory	R/W	All zeros	14.5.6.9.5/14-81
0x3_4800– 0x3_4FFF	AESU FIFO	R/W	All zeros	14.5.6.9.6/14-81
Message Digest Execution Unit (MDEU)				
0x3_6000	MDEUMR—MDEU mode register	R/W	All zeros	14.5.4.1/14-51
0x3_6008	MDEUKSR—MDEU key size register	R/W	All zeros	14.5.4.3/14-55
0x3_6010	MDEUDSR—MDEU data size register	R/W	All zeros	14.5.4.4/14-56
0x3_6018	MDEURCR—MDEU reset control register	R/W	All zeros	14.5.4.5/14-56
0x3_6028	MDEUSR—MDEU status register	R	All zeros	14.5.4.6/14-57
0x3_6030	MDEUISR—MDEU interrupt status register	R	All zeros	14.5.4.7/14-58
0x3_6038	MDEUICR—MDEU interrupt control register	R/W	All zeros	14.5.4.8/14-59
0x3_6050	MDEUEUG—MDEU EU-Go register	W	All zeros	14.5.4.10/14-61
0x3_6100– 0x3_6120	MDEU context memory registers	R/W	All zeros	14.5.4.11/14-61
0x3_6400– 0x3_647F	MDEU key memory	W	All zeros	14.5.4.12/14-62
0x3_6800– 0x3_6FFF	MDEU FIFO	W	All zeros	14.5.4.13/14-63
ARC Four Execution Unit (AFEU)				
0x3_8000	AFEUMR—AFEU mode register	R/W	All zeros	14.5.3.1/14-43
0x3_808	AFEUKSR—AFEU key size register	R/W	All zeros	14.5.3.3/14-44
0x3_8010	AFEUDSR—AFEU data size register	R/W	All zeros	14.5.3.4/14-45
0x3_8018	AFEURCR—AFEU reset control register	R/W	All zeros	14.5.3.5/14-46
0x3_8028	AFEUSR—AFEU status register	R	All zeros	14.5.3.6/14-46
0x3_8030	AFEUISR—AFEU interrupt status register	R	All zeros	14.5.3.7/14-47
0x3_8038	AFEUICR—AFEU interrupt control register	R/W	All zeros	14.5.3.8/14-49
0x3_8050	AFEUEMR—AFEU end of message register	W	All zeros	14.5.3.9/14-50
0x3_8100– 0x3_81FF	AFEU context memory registers	R/W	All zeros	14.5.3.10.1/14-50

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_8200	AFEU context memory pointers	R/W	All zeros	14.5.3.10.2/14-51
0x3_8400	AFEUK0—AFEU key register 0	W	—	14.5.3.11/14-51
0x3_848	AFEUK1—AFEU key register 1	W	—	14.5.3.11/14-51
0x3_8800– 0x3_8FFF	AFEU FIFO	R/W	All zeros	14.5.3.11.1/14-51
Random Number Generator (RNG)				
0x3_A000	RNGMR—RNG mode register	R/W	All zeros	14.5.5.1/14-63
0x3_A010	RNGDSR—RNG data size register	R/W	All zeros	14.5.5.2/14-64
0x3_A018	RNGRCR—RNG reset control register	R/W	All zeros	14.5.5.3/14-65
0x3_A028	RNGSR—RNG status register	R	All zeros	14.5.5.4/14-65
0x3_A030	RNGISR—RNG interrupt status register	R	All zeros	14.5.5.5/14-66
0x3_A038	RNGICR—RNG interrupt control register	R/W	All zeros	14.5.5.6/14-67
0x3_A050	RNGEUG—RNG EU-Go register	W	All zeros	14.5.5.7/14-68
0x3_A800– 0x3_AFFF	RNG FIFO	R	All zeros	14.5.5.8/14-68
Public Key Execution Unit (PKEU)				
0x3_C000	PKEUMR—PKEU mode register	R/W	All zeros	14.5.1.1/14-26
0x3_C008	PKEUKSR—PKEU key size register	R/W	All zeros	14.5.1.2/14-28
0x3_C010	PKEUDSR—PKEU data size register	R/W	All zeros	14.5.1.3/14-28
0x3_C018	PKEURCR—PKEU reset control register	R/W	All zeros	14.5.1.5/14-29
0x3_C028	PKEUSR—PKEU status register	R	All zeros	14.5.1.6/14-30
0x3_C030	PKEUISR—PKEU interrupt status register	R	All zeros	14.5.1.7/14-31
0x3_C038	PKEUICR—PKEU interrupt control register	R/W	All zeros	14.5.1.8/14-32
0x3_C040	PKEUABS—PKEU AB size register	R/W	All zeros	14.5.1.3/14-28
0x3_C050	PKEUEUG—PKEU EU-Go	W	All zeros	14.5.1.9/14-33

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_C200– 0x3_C23F	PKEU parameter memory A0	R/W	All zeros	14.5.1.10/14-34
0x3_C240– 0x3_C27F	PKEU parameter memory A1	R/W	All zeros	
0x3_C280– 0x3_C2BF	PKEU parameter memory A2	R/W	All zeros	
0x3_C2C0– 0x3_C2FF	PKEU parameter memory A3	R/W	All zeros	
0x3_C300– 0x3_C33F	PKEU parameter memory B0	R/W	All zeros	
0x3_C340– 0x3_C37F	PKEU parameter memory B1	R/W	All zeros	
0x3_C380– 0x3_C3BF	PKEU parameter memory B2	R/W	All zeros	
0x3_C3C0– 0x3_C3FF	PKEU parameter memory B3	R/W	All zeros	
0x3_C400– 0x3_C4FF	PKEU parameter memory E	W	All zeros	
0x3_C800– 0x3_C8FF	PKEU parameter memory N	R/W	All zeros	
QUICC Engine 2.0 Block				
0x10_0000 – 0x1F_FFFF	See Table 2-3 for more information.			

- ¹ Depends on the reset configuration word high values. See [Section 5.2.4.3.1, “LBLAWBAR0\[BASE_ADDR\] Reset Value,”](#) for details.
- ² Depends on the reset configuration word high values. See [Section 5.2.4.4.1, “LBLAWAR0\[EN\] and LBLAWAR0\[SIZE\] Reset Value,”](#) for details.
- ³ Depends on the reset configuration word high values. See [Section 5.2.4.5.1, “PCILAWBAR0\[BASE_ADDR\] Reset Value,”](#) for details.
- ⁴ Depends on the reset configuration word high values. See [Section 5.2.4.6.1, “PCILAWAR0\[EN\] and PCILAWAR0\[SIZE\] Reset Value,”](#) for details.
- ⁵ Depends on the reset configuration word high values. See [Section 5.2.4.7.1, “DDRLAWBAR0\[BASE_ADDR\] Reset Value,”](#) for details.
- ⁶ Depends on the reset configuration word high values. See [Section 5.2.4.8.1, “DDRLAWAR0\[EN\] and DDRLAWAR0\[SIZE\] Reset Value,”](#) for details.
- ⁷ Depends on the reset configuration word high values.
- ⁸ SWCRR[SWEN] reset value directly depends on RCWHR[SWEN] (reset configuration word high).

2.4 QUICC Engine Internal Memory Map

The QUICC Engine block’s internal memory resources are mapped within a contiguous block of memory. The size of the internal space is 1 Mbyte. The location of the QUICC Engine internal memory space within the global system memory space can be mapped (aligned to a 1-Mbyte boundary) through an

Memory Map

implementation-specific special register, see [Section 5.2.4.1, “Internal Memory Map Registers Base Address Register \(IMMRBAR\),”](#) for more information. Note that the last 768 Kbytes of the QUICC Engine internal memory space are reserved for future expansions. Any access to reserved regions will result in undefined behavior.

Figure 2-1 is a high level representation of the QUICC Engine memory map.

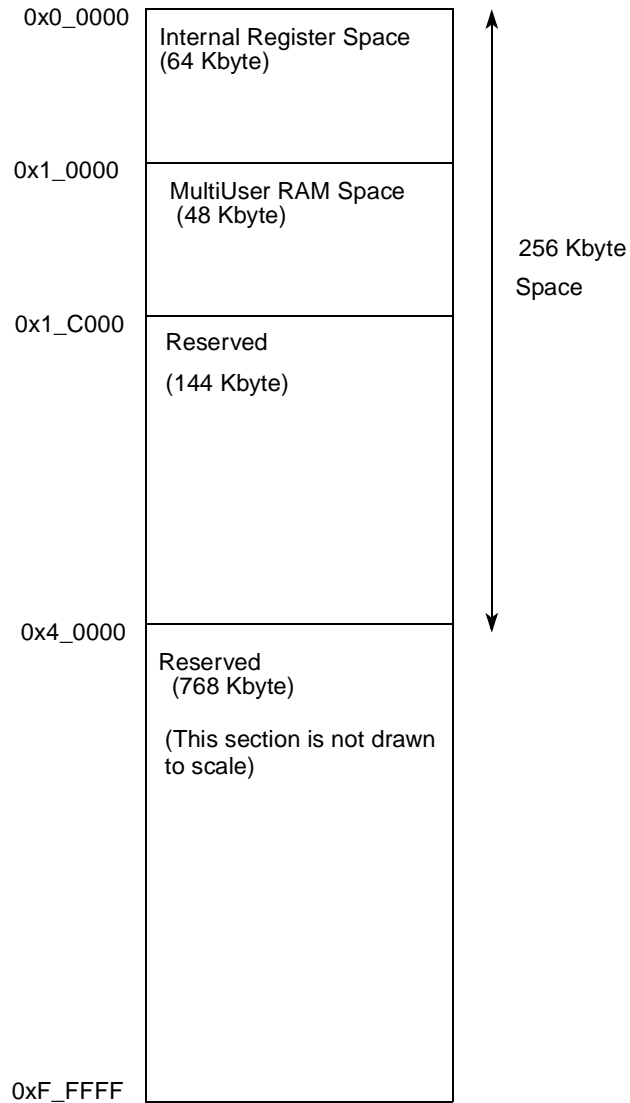


Figure 2-1. QUICC Engine 2.0 High-Level Memory Map

Table 2-3 defines the internal memory map of the QUICC Engine block.

Table 2-3. QUICC Engine High-Level Memory Map

Internal Address	Abbreviation	Name	Size	Comments
0x0_0000–0x0_FFFF	Register Space			
0x0_0000–0x0_3FFF	General Space			
0x0_0000–0x0_003F	I-RAM	Instruction RAM registers	64 bytes	—
0x0_0040–0x0_007F	Reserved	—	64 bytes	—
0x0_0080–0x0_00FF	IRQ	Interrupt controller	128 bytes	—
0x0_0100–0x0_01FF	RISC Config	RISC configuration register	256 bytes	—
0x0_0200–0x0_03FF	Reserved	—	512 bytes	—
0x0_0400–0x0_043F	QUICC Engine Mux	QUICC Engine clock multiplexer registers	64 bytes	—
0x0_0440–0x0_047F	Timers	QUICC Engine timers	64 bytes	—
0x0_0480–0x0_04BF	Reserved	—	64 bytes	—
0x0_04C0–0x0_04FF	SPI1	SPI1 registers	64 bytes	—
0x0_0500–0x0_053F	SPI2	SPI2 registers	64 bytes	—
0x0_0540–0x0_057F	MCC	MCC registers	64 bytes	—
0x0_0580–0x0_063F	Reserved	—	192 bytes	—
0x0_0640–0x0_067F	BRG	Baud rate generator registers	64 bytes	—
0x0_0680–0x0_06FF	Reserved	—	128 bytes	—
0x0_06C0–0x0_06FF	USB1.0	USB 1.0 registers	64 bytes	—
0x0_0700–0x0_077F	SI1	SI1 registers	128 bytes	—
0x0_0780–0x0_07FF	Reserved	—	128 bytes	—
0x0_0800–0x0_0FFF	Reserved	—	2K bytes	—
0x0_1000–0x0_17FF	SI1RT	SI1 routing table	2K bytes	—
0x0_1800–0x0_1FFF	Reserved	—	2K bytes	—
0x0_2000–0x0_21FF	UCC1	UCC1 registers	512 bytes	—
0x0_2200–0x0_23FF	UCC3	UCC3 registers	512 bytes	—
0x0_2400–0x0_25FF	UCC5	UCC5 registers	512 bytes	—
0x0_2600–0x0_27FF	UCC7	UCC7 registers	512 bytes	—
0x0_2800–0x0_2DFF	Reserved	—	1536 bytes	—
0x0_2E00–0x0_2FFF	UPC1	Utopia POS controller 1	512 bytes	—
0x0_3000–0x0_31FF	UCC2	UCC2 registers	512 bytes	—
0x0_3200–0x0_33FF	UCC4	UCC4 registers	512 bytes	—
0x0_3400–0x0_35FF	UCC6	UCC6 registers	512 bytes	—

Table 2-3. QUICC Engine High-Level Memory Map (continued)

Internal Address	Abbreviation	Name	Size	Comments
0x0_3600–0x0_37FF	UCC8	UCC8 registers	512 bytes	—
0x0_3800–0x0_3DFF	Reserved	—	1536 bytes	
0x0_3E00–0x0_3FFF	UPC2	Utopia POS controller 2	512 bytes	—
0x0_4000–0x0_407F	SDMA	Serial DMA	128 bytes	—
0x0_4080–0x0_7FFF	Debug Space			
0x0_4080–0x0_42FF	Reserved	—	256 bytes	—
0x0_4300–0x0_43FF	Reserved	—	256 bytes	
0x0_4400–0x0_45FF	Reserved	—	256 bytes	
0x0_4600–0x0_467F	Reserved	—	128 bytes	—
0x0_4680–0x0_46FF	Reserved	—	128 bytes	—
0x0_4700–0x0_477F	Reserved	—	128 bytes	
0x0_4780–0x0_47FF	Reserved	—	128 bytes	
0x0_4800–0x0_4FFF	IEEE 1588	IEEE 1588 Registers	2 Kbytes	—
0x0_5000–0x0_7FFF	Reserved	—	12 Kbytes	—
0x0_8000–0x3_FFFF	RAM Space			
0x0_8000–0x0_FFFF	Reserved	—	32 Kbytes	—
0x1_0000–0x1_BFFF	MURAM	Multi-user RAM	48 Kbytes	—
0x1_C000–0x3_FFFF	Reserved	—	144 Kbytes	
0x4_0000–0xF_FFFF	Reserved	—	768 Kbytes	

Chapter 3

Signal Descriptions

This chapter describes the MPC8360E external signals. It is organized into the following sections:

- Overview of signals and cross references for signals that serve multiple functions, including two lists: one ordered by functional block and one alphabetical.
- List of reset configuration signals
- List of output signal states at reset
- Parallel I/O port signals

NOTE

A bar over a signal name indicates that the signal is active low, such as $\overline{\text{IRQ_OUT}}$ (interrupt out). Active-low signals are referred to as asserted (active) when they are low and negated when they are high. Signals that are not active low, such as IRQ (interrupt input), are referred to as asserted when they are high and negated when they are low.

Internal signals throughout this document are shown as lower case and in italics. For example, *sys_logic_clk* is an internal signal. These are referenced only as necessary for understanding of the external functionality of the device.

3.1 Signals Overview

The MPC8360E signals are grouped as follows:

- DDR memory interface signals
- PCI interface signals
- QUICC Engine interface signals
- DUART interface signals
- I²C interface signals
- Local bus interface signals
- PIC interface signals
- PM interface signal
- JTAG, PMC, system control signals
- Clock signals
- PCI mode signal

Figure 3-1 illustrates the external signals of the MPC8360E, showing how the signals are grouped. Refer to the *MPC8360E Integrated Processor Hardware Specifications* for a pinout diagram showing pin numbers and a listing of all the electrical and mechanical specifications.

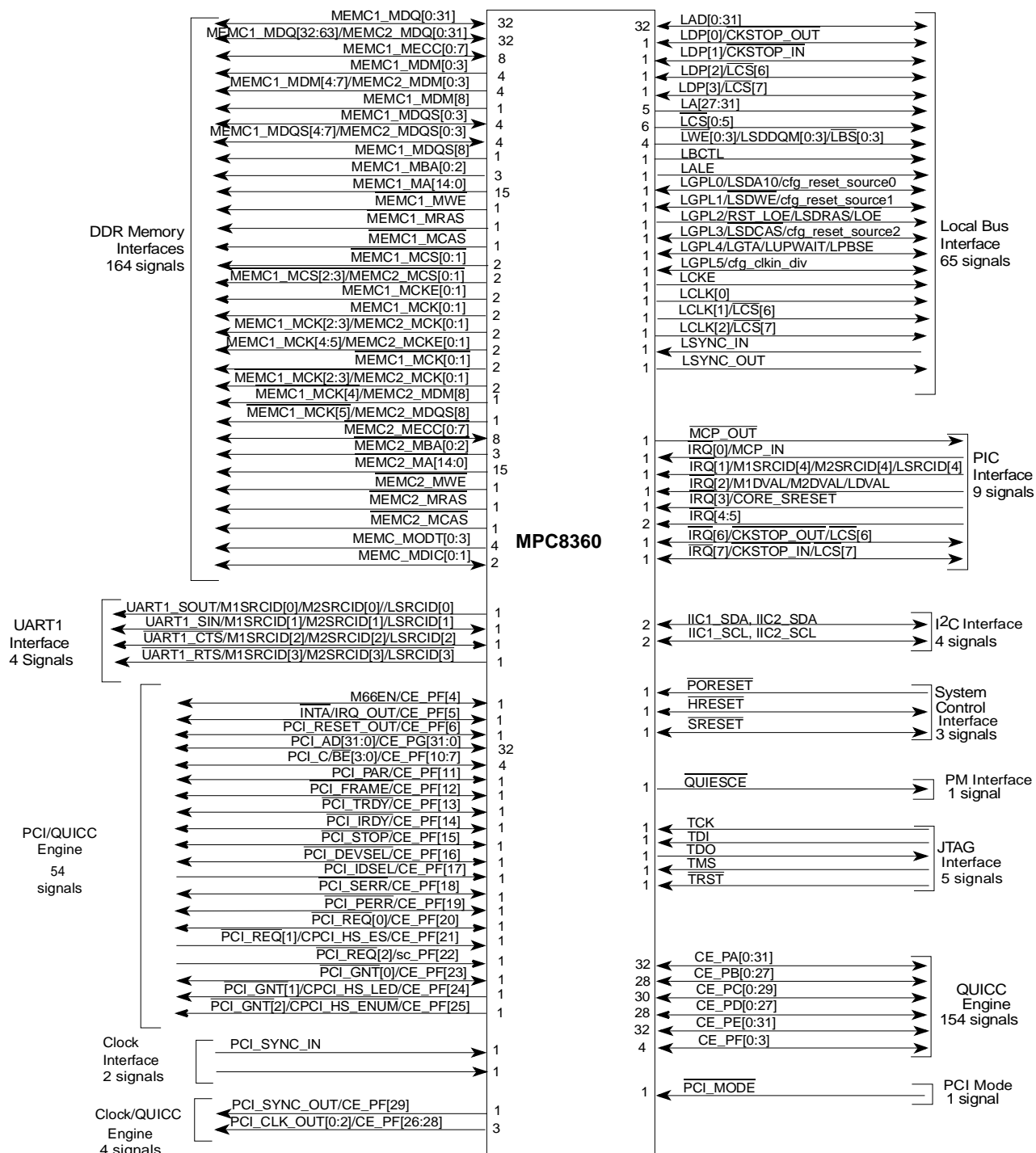


Figure 3-1. MPC8360E Signal Groupings

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

The following tables provide summaries of signal functions. [Table 3-1](#) provides a summary of the signals grouped by function, and [Table 3-2](#) provides a summary of the signals grouped alphabetically. These tables detail the signal name, interface, alternate functions, number of signals, and whether the signal is an input, an output, or bidirectional. They also provide a pointer to the table where the signal function is described.

Table 3-1. MPC8360E Signal Reference by Functional Block

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
Primary DDR SDRAM Memory Controller Interface						
MEMC1_MDQ[0:31]	—	Primary DDR data	DDR	32	I/O	9-3/9-6
MEMC1_MDQ[32:63]/ MEMC2_MDQ[0:31]	MEMC1_MDQ[32:63]	Primary DDR data	DDR	32	I/O	9-3/9-6
	MEMC2_MDQ[0:31]	Secondary DDR data	DDR	32	I/O	9-3/9-6
MEMC1_MECC[0:4]	MEMC1_MSRCID[0:4]	Primary DDR error correcting code	DDR	5	I/O	9-3/9-6
MEMC1_MECC[5]	MEMC1_MDVAL	Primary DDR error correcting code	DDR	1	I/O	9-3/9-6
MEMC1_MECC[6:7]	—	Primary DDR error correcting code	DDR	3	I/O	9-3/9-6
MEMC1_MDM[0:3]	—	Primary DDR data mask 0–3	DDR	4	O	9-3/9-6
MEMC1_MDM[4:7]/ MEMC2_MDM[0:3]	MEMC1_MDM[4:7]	Primary DDR data mask 4–7	DDR	4	O	9-3/9-6
	MEMC2_MDM[0:3]	Secondary DDR data mask 0–3			O	9-3/9-6
MEMC1_MDM[8]	—	Primary DDR ECC data mask	DDR	1	O	9-3/9-6
MEMC1_MDQS[0:3]	—	Primary DDR data strobe	DDR	4	I/O	9-3/9-6
MEMC1_MDQS[4:7]/ MEMC2_MDQS[0:3]	MEMC1_MDQS[4:7]	Primary DDR data strobe	DDR	4	I/O	9-3/9-6
	MEMC2_MDQS[0:3]	Secondary DDR data strobe			I/O	9-3/9-6
MEMC1_MDQS[8]	—	Primary DDR data strobe	DDR	1	I/O	9-3/9-6
MEMC1_MBA[0:2]	—	Primary DDR bank select	DDR	3	O	9-3/9-6
MEMC1_MA[14:0]	—	Primary DDR address	DDR	15	O	9-3/9-6
MEMC1_MWE	—	Primary DDR write enable	DDR	1	O	9-3/9-6
MEMC1_MRAS	—	Primary DDR row address strobe	DDR	1	O	9-3/9-6
MEMC1_MCAS	—	Primary DDR column address strobe	DDR	1	O	9-3/9-6
MEMC1_MCS[0:1]	—	Primary DDR chip select (2/DIMM)	DDR	2	O	9-3/9-6
MEMC1_MCS[2:3]/ MEMC2_MCS[0:1]	MEMC1_MCS[2:3]	Primary DDR chip select (2/DIMM)	DDR	2	O	9-3/9-6
	MEMC2_MCS[0:1]	Secondary DDR chip select (2/DIMM)			O	9-3/9-6
MEMC1_MCKE[0:1]	—	Primary DDR clock enable	DDR	2	O	9-4/9-9

Table 3-1. MPC8360E Signal Reference by Functional Block (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
MEMC1_MCK[0:1]	—	Primary DDR differential clocks (positive)	DDR	2	O	9-4/9-9
MEMC1_MCK[2:3]/ MEMC2_MCK[0:1]	MEMC1_MCK[2:3]	Primary DDR differential clocks (positive)	DDR	2	O	9-4/9-9
	MEMC2_MCK[0:1]	Secondary DDR differential clocks (positive)	DDR		O	9-4/9-9
MEMC1_MCK[4:5]/ MEMC2_MCKE[0:1]	MEMC1_MCK[4:5]	Primary DDR differential clocks (positive)	DDR	2	O	9-4/9-9
	MEMC2_MCKE[0:1]	Secondary DDR clock enable			O	9-4/9-9
$\overline{\text{MEMC1_MCK}}[0:1]$	—	Primary DDR differential clocks (negative)	DDR	2	O	9-4/9-9
$\overline{\text{MEMC1_MCK}}[2:3]/$ $\overline{\text{MEMC2_MCK}}[0:1]$	$\overline{\text{MEMC1_MCK}}[2:3]$	Primary DDR differential clocks (negative)	DDR	2	O	9-4/9-9
	$\overline{\text{MEMC2_MCK}}[0:1]$	Secondary DDR differential clocks (negative)	DDR		O	9-4/9-9
$\overline{\text{MEMC1_MCK}}[4]/$ $\overline{\text{MEMC2_MDM}}[8]$	$\overline{\text{MEMC1_MCK}}[4]$	Primary DDR differential clocks (negative)	DDR	1	O	9-4/9-9
	MEMC2_MDM[8]	Secondary DDR ECC data mask	DDR		O	9-4/9-9
$\overline{\text{MEMC1_MCK}}[5]/$ $\overline{\text{MEMC2_MDQS}}[8]$	$\overline{\text{MEMC1_MCK}}[5]$	Primary DDR differential clocks (negative)	DDR	1	O	9-4/9-9
	MEMC2_MDQS[8]	Secondary DDR data strobe	DDR		I/O	9-4/9-9
MEMC1_MODT[0:3]	—	On-die termination for DRAM chip	DDR	4	O	9-3/9-6
MEMC1_MDIC[0:1]	—	DRAM driver impedance calibration	DDR	2	I/O	9-3/9-6
Secondary DDR SDRAM Memory Controller Interface						
MEMC2_MECC[0:4]	MEMC2_MSRCID[0:4]	Secondary DDR error correcting code	DDR	5	I/O	9-3/9-6
MEMC2_MECC[5]	MEMC2_MDVAL	Secondary DDR error correcting code	DDR	1	I/O	9-3/9-6
MEMC2_MECC[6:7]	—	Secondary DDR error correcting code	DDR	3	I/O	9-3/9-6
MEMC2_MBA[0:2]	—	Secondary DDR bank select	DDR	3	O	9-3/9-6
MEMC2_MA[14:0]	—	Secondary DDR address	DDR	15	O	9-3/9-6
$\overline{\text{MEMC2_MWE}}$	—	Secondary DDR write enable	DDR	1	O	9-3/9-6
$\overline{\text{MEMC2_MRAS}}$	—	Secondary DDR row address strobe	DDR	1	O	9-3/9-6
$\overline{\text{MEMC2_MCAS}}$	—	Secondary DDR column address strobe	DDR	1	O	9-3/9-6

Table 3-1. MPC8360E Signal Reference by Functional Block (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
PCI						
$\overline{\text{PCI_INTA}}$ / IRQ_OUT/ CE_PF5	$\overline{\text{PCI_INTA}}$	PCI interrupt output	PCI	1	O	13-3/13-5
	IRQ_OUT	Interrupt output	IPIC		O	13-3/13-5
	CE_PF5	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
$\overline{\text{PCI_RESET_OUT}}$ / CE_PF6	$\overline{\text{PCI_RESET_OUT}}$	PCI reset	PCI	1	O	13-3/13-5
	CE_PF6	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_AD[31:0]/ CE_PG[31:0]	PCI_AD[31:0]	PCI address/data	PCI	32	I/O	13-3/13-5
	CE_PG[31:0]	QUICC Engine parallel I/O	PIO		I/O	3-22/3-77
PCI_C/ $\overline{\text{BE}}$ [3:0]/ CE_PF[10:7]	PCI_C	PCI command	PCI	4	I/O	13-3/13-5
	$\overline{\text{BE}}$ [3:0]	Byte enable	PCI		I/O	13-3/13-5
	CE_PF[10:7]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_PAR/ CE_PF11	PCI_PAR	PCI parity	PCI	1	I/O	13-3/13-5
	CE_PF11	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
$\overline{\text{PCI_FRAME}}$ / CE_PF12	$\overline{\text{PCI_FRAME}}$	PCI frame	PCI	1	I/O	13-3/13-5
	CE_PF12	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
$\overline{\text{PCI_TRDY}}$ / CE_PF13	$\overline{\text{PCI_TRDY}}$	PCI target ready	PCI	1	I/O	13-3/13-5
	CE_PF13	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
$\overline{\text{PCI_IRDY}}$ / CE_PF14	$\overline{\text{PCI_IRDY}}$	PCI initiator ready	PCI	1	I/O	13-3/13-5
	CE_PF14	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
$\overline{\text{PCI_STOP}}$ / CE_PF15	$\overline{\text{PCI_STOP}}$	PCI stop	PCI	1	I/O	13-3/13-5
	CE_PF15	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
$\overline{\text{PCI_DEVSEL}}$ / CE_PF16	$\overline{\text{PCI_DEVSEL}}$	PCI device select	PCI	1	I/O	13-3/13-5
	CE_PF16	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_IDSEL/ CE_PF17	PCI_IDSEL	PCI initial device select	PCI	1	I	13-3/13-5
	CE_PF17	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
$\overline{\text{PCI_SERR}}$ / CE_PF18	$\overline{\text{PCI_SERR}}$	PCI system error	PCI	1	I/O	13-3/13-5
	CE_PF18	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
$\overline{\text{PCI_PERR}}$ / CE_PF19	$\overline{\text{PCI_PERR}}$	PCI parity error	PCI	1	I/O	13-3/13-5
	CE_PF19	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
$\overline{\text{PCI_REQ0}}$ / CE_PF20	$\overline{\text{PCI_REQ0}}$	PCI request 0	PCI	1	I/O	13-3/13-5
	CE_PF20	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69

Table 3-1. MPC8360E Signal Reference by Functional Block (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
PCI_REQ1/ CPCI_HS_ES/ CE_PF21	$\overline{\text{PCI_REQ1}}$	PCI request 1	PCI	1	I	13-3/13-5
	CPCI_HS_ES	Compact PCI hot swap ejector switch	CPCI		I	13-3/13-5
	CE_PF21	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_REQ2/ CE_PF22	$\overline{\text{PCI_REQ2}}$	PCI request 2	PCI	1	I	13-3/13-5
	CE_PF22	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_GNT0/ CE_PF23	$\overline{\text{PCI_GNT0}}$	PCI grant 0	PCI	1	I/O	13-3/13-5
	CE_PF23	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_GNT1/ CPCI_HS_LED/ CE_PF24	$\overline{\text{PCI_GNT1}}$	PCI grant 1	PCI	1	O	13-3/13-5
	CPCI_HS_LED	Compact PCI hot swap LED	CPCI		O	13-3/13-5
	CE_PF24	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_GNT2/ CPCI_HS_ENUM/ CE_PF25	$\overline{\text{PCI_GNT2}}$	PCI grant 2	PCI	1	O	13-3/13-5
	$\overline{\text{CPCI_HS_ENUM}}$	Compact PCI hot swap enumerator	CPCI		O	13-3/13-5
	CE_PF25	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
$\overline{\text{PCI_MODE}}$	—	PCI mode select	PCI	1	I	5-32/5-24
M66EN/CE_PF4	M66EN	PCI 66-MHz timing on/off.	PCI	1	I	13-3/13-5
	CE_PF4	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
Local Bus Controller Interface						
LAD[0:31]	—	LBC address/data	LBC	32	I/O	10-2/10-5
LDP[0]/ $\overline{\text{CKSTOP_OUT}}$	LDP[0]	LBC data parity 0	LBC	1	I/O	10-2/10-5
	$\overline{\text{CKSTOP_OUT}}$	Checkstop out	CPU		O	4-3/4-4
LDP[1]/ $\overline{\text{CKSTOP_IN}}$	LDP[1]	LBC data parity 1	LBC	1	I/O	10-2/10-5
	$\overline{\text{CKSTOP_IN}}$	Checkstop in	CPU		I	4-3/4-4
LDP[2]/ $\overline{\text{LCS}}$ [6]	LDP[2]	LBC data parity 2	LBC	1	I/O	10-2/10-5
	$\overline{\text{LCS}}$ [6]	LBC chip select 6			O	10-2/10-5
LDP[3]/ $\overline{\text{LCS}}$ [7]	LDP[3]	LBC data parity 3	LBC	1	I/O	10-2/10-5
	$\overline{\text{LCS}}$ [7]	LBC chip select 7			O	10-2/10-5
LA[27:31]	—	LBC port address 27–31	LBC	5	O	10-2/10-5
$\overline{\text{LCS}}$ [0:5]	—	LBC chip select 0–5	LBC	6	O	10-2/10-5
$\overline{\text{LWE}}$ [0:3]/ LSDDQM[0:3]/ $\overline{\text{LBS}}$ [0:3]	$\overline{\text{LWE}}$ [0:3]	LBC write enable 0–3	LBC	4	O	10-2/10-5
	LSDDQM[0:3]	Byte lane data mask 0–3			O	3-4/3-19
	$\overline{\text{LBS}}$ [0:3]	Byte select 0–3			O	10-2/10-5

Table 3-1. MPC8360E Signal Reference by Functional Block (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
LBCTL	—	LBC data buffer control	LBC	1	O	10-2/10-5
LALE	—	LBC address latch enable	LBC	1	O	10-2/10-5
LGPL0/LSDA10/ cfg_reset_source0	LGPL0	LBC UPM general purpose line 0	LBC	1	O	10-2/10-5
	LSDA10	SDRAM address bit 10	LBC		O	10-2/10-5
	cfg_reset_source0	Configuration reset source 0	Reset & clock		I	4-1/4-1
LGPL1/ $\overline{\text{LSDWE}}$ / cfg_reset_source1	LGPL1	LBC UPM general purpose line 1	LBC	1	O	10-2/10-5
	$\overline{\text{LSDWE}}$	SDRAM write enable	LBC		O	10-2/10-5
	cfg_reset_source1	Configuration reset source 1	Reset & clock		I	4-1/4-1
LGPL2/ $\overline{\text{LSDRAS}}$ / $\overline{\text{LOE}}$	LGPL2	LBC UPM general purpose line 2	LBC	1	O	10-2/10-5
	$\overline{\text{LSDRAS}}$	SDRAM RAS			O	10-2/10-5
	$\overline{\text{LOE}}$	LBC output enable			O	10-2/10-5
LGPL3/ $\overline{\text{LSDCAS}}$ /cfg_r reset_source2	LGPL3	LBC UPM general purpose line 3	LBC		O	10-2/10-5
	$\overline{\text{LSDCAS}}$	SDRAM CAS	LBC		O	10-2/10-5
	cfg_reset_source2	Configuration reset source 2	Reset & clock		I	4-1/4-1
LGPL4/ $\overline{\text{LGTA}}$ / LUPWAIT/LPBSE	LGPL4	LBC UPM general purpose line 4	LBC	1	I/O	10-2/10-5
	$\overline{\text{LGTA}}$	GPCM terminate access				
	LUPWAIT	UPM wait				
	LPBSE	LBC parity byte select				
LGPL5/ cfg_clk_in_div	LGPL5	LBC UPM general purpose line 5	LBC	1	O	10-2/10-5
	cfg_clk_in_div	Configuration clock in div	Reset & clock		I	4-1/4-1
LCKE	—	LBC clock enable	LBC	1	O	10-2/10-5
LCLK[0]	—	LBC clocks 0	LBC	1	O	10-2/10-5
LCLK[1]/ $\overline{\text{LCS}}[6]$	LCLK[1]	LBC clocks 1	LBC	1	O	10-2/10-5
	$\overline{\text{LCS}}[6]$	LBC chip select 6				
LCLK[2]/ $\overline{\text{LCS}}[7]$	LCLK[2]	LBC clocks 2	LBC	1	O	10-2/10-5
	$\overline{\text{LCS}}[7]$	LBC chip select 7				
LSYNC_OUT	—	LBC DLL synchronization output	LBC	1	O	10-2/10-5
LSYNC_IN	—	LBC DLL synchronization input	LBC	1	I	10-2/10-5

Table 3-1. MPC8360E Signal Reference by Functional Block (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
Programmable Interrupt Controller						
MCP_OUT	—	Machine check interrupt output	IPIC	1	O	8-2/8-5
$\overline{\text{IRQ}}[0]/$ MCP_IN	$\overline{\text{IRQ}}[0]$	External interrupt 0	IPIC	1	I	8-2/8-5
	$\overline{\text{MCP_IN}}$	Machine check interrupt input	IPIC		I	8-2/8-5
$\overline{\text{IRQ}}[1]$	—	External interrupt 1	IPIC	1	I	8-2/8-5
$\overline{\text{IRQ}}[1]/$ M1SRCID[4]/ M2SRCID[4]/ LSRCID[4]	$\overline{\text{IRQ}}[1]$	External interrupt 1	IPIC	1	I	8-2/8-5
	M1SRCID[4]	DDR memory debug source port ID 4	DDR		O	9-7/9-13
	M2SRCID[4]	Secondary DDR memory debug source port ID 4	DDR		O	9-7/9-13
	LSRCID[4]	LBC debug source port ID 4	LBC		O	10-2/10-5
$\overline{\text{IRQ}}[2]/$ M1DVAL/ M2DVAL/ LDVAL	$\overline{\text{IRQ}}[2]$	External interrupt 2	IPIC	1	I	8-2/8-5
	M1DVAL	DDR memory debug data valid	DDR		O	9-7/9-13
	M2DVAL	Secondary DDR memory debug data valid	DDR		O	9-7/9-13
	LDVAL	LBC debug data valid	LBC		O	10-2/10-5
$\overline{\text{IRQ}}[3]/$ CORE_SRESET	$\overline{\text{IRQ}}[3]$	External interrupt 3	IPIC	1	I	8-2/8-5
	$\overline{\text{CORE_SRESET}}$	Core soft reset	Core	1	I	
$\overline{\text{IRQ}}[4:5]$	—	External interrupt 4–5	IPIC	1	I	8-2/8-5
$\overline{\text{IRQ}}[6]/$ LCS[6] CKSTOP_OUT	$\overline{\text{IRQ}}[6]$	External interrupt 6	IPIC	1	I	8-2/8-5
	$\overline{\text{LCS}}[6]$	LBC chip select 6	LBC		O	10-2/10-5
	$\overline{\text{CKSTOP_OUT}}$	Checkstop output	CPU		O	4-3/4-4
$\overline{\text{IRQ}}[7]/$ LCS[7]/ CKSTOP_IN	$\overline{\text{IRQ}}[7]$	External interrupt 7	IPIC	1	I	8-2/8-5
	$\overline{\text{LCS}}[7]$	LBC chip select 7	LBC		O	10-2/10-5
	$\overline{\text{CKSTOP_IN}}$	Checkstop input	CPU		I	4-3/4-4
DUART						
UART1_SOUT/ M1SRCID[0]/ M2SRCID[0] LSRCID[0]	UART1_SOUT	UART1 serial data out	UART	1	O	16-2/16-3
	M1SRCID[0]	DDR memory debug source port ID 0	DDR		O	9-7/9-13
	M2SRCID[0]	Secondary DDR memory debug source port ID 0	DDR		O	9-7/9-13
	LSRCID[0]	LBC debug source port ID 0	LBC		O	10-2/10-5

Table 3-1. MPC8360E Signal Reference by Functional Block (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
UART1_SIN/ M1SRCID[1]/ M2SRCID[1] LSRCID[1]	UART1_SIN	UART1 serial data in	UART	1	I	16-2/16-3
	M1SRCID[1]	DDR memory debug source port ID 1	DDR		O	9-7/9-13
	M2SRCID[1]	Secondary DDR memory debug source port ID 1	DDR		O	9-7/9-13
	LSRCID[1]	LBC debug source port ID 1	LBC		O	10-2/10-5
UART1_CTS/ M1SRCID[2]/ M2SRCID[2] LSRCID[2]	$\overline{\text{UART1_CTS}}$	UART1 clear to send	UART	1	I	16-2/16-3
	M1SRCID2	DDR memory debug source port ID 2	DDR		O	9-7/9-13
	M2SRCID2	Secondary DDR memory debug source port ID 2	DDR		O	9-7/9-13
	LSRCID2	LBC debug source port ID 2	LBC		O	10-2/10-5
UART1_RTS M1SRCID[3]/ M2SRCID[3] LSRCID[3]	$\overline{\text{UART1_RTS}}$	UART1 ready to send	UART	1	O	16-2/16-3
	M1SRCID3	DDR memory debug source port ID 3	DDR		O	9-7/9-13
	M2SRCID3	Secondary DDR memory debug source port ID 3	DDR		O	9-7/9-13
	LSRCID3	LBC debug source port ID 3	LBC		O	10-2/10-5
UART2 signals are multiplexed with QUICC Engine functions; see Table 3-20						
I²C Interface						
IIC1_SDA	—	I2C serial data	I2C1	1	I/O	15-2/15-4
IIC1_SCL	—	I2C serial clock	I2C1	1	I/O	15-2/15-4
IIC2_SDA	—	I2C serial data	I2C2	1	I/O	15-2/15-4
IIC2_SCL/	—	I2C serial clock	I2C2	1	I/O	15-2/15-4
QUICC Engine Block						
CE_PA[0:31]	—	QUICC Engine parallel port A	QUICC Engine	32	I/O	3-16/3-31
CE_PB[0:27]	—	QUICC Engine parallel port B	QUICC Engine	28	I/O	3-17/3-39
CE_PC[0:29]	—	QUICC Engine parallel port C	QUICC Engine	30	I/O	3-18/3-47
CE_PD[0:27]	—	QUICC Engine parallel port D	QUICC Engine	28	I/O	3-19/3-53
CE_PE[0:31]	—	QUICC Engine parallel port E	QUICC Engine	32	I/O	3-20/3-60

Table 3-1. MPC8360E Signal Reference by Functional Block (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
CE_PF[0:3]	—	QUICC Engine parallel port F	QUICC Engine	4	I/O	3-21/3-69
Clocks						
PCI_CLK_OUT[0:2] CE_PF[26:28]	PCI_CLK_OUT[0:2]	PCI clock out 0–2	Clocks	3	O	4-2/4-3
	CE_PF[26:28]	QUICC Engine port F	QUICC Engine		I/O	3-21/3-69
CLK_IN	—	Clock input	Clocks	1	I	4-2/4-3
PCI_SYNC_IN/ PCI_CLOCK	PCI_SYNC_IN	PCI clock sync input	Clocks	1	I	4-2/4-3
	PCI_CLOCK	PCI clock	Clocks			
PCI_SYNC_OUT/ CE_PF[29]	PCI_SYNC_OUT	PCI clock sync output	Clocks	1	O	4-2/4-3
	CE_PF[29]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
JTAG						
TCK	—	Test clock	JTAG	1	I	17-2/17-2
TDI	—	Test data in	JTAG	1	I	17-2/17-2
TDO	—	Test data out	JTAG	1	O	17-2/17-2
TMS	—	Test mode select	JTAG	1	I	17-2/17-2
TRST	—	Test reset	JTAG	1	I	17-2/17-2
Test						
TEST_SEL (MPC8360) $\overline{\text{TEST_SEL}}$ (MPC8358)	—	Test mode	TEST	1	I	—
TEST	—	Test mode	TEST	1	I	—
PMC						
$\overline{\text{QUIESCE}}$	—	Quiesce state	PMC	1	O	5-77/5-73
System Control						
$\overline{\text{PORESET}}$	—	Power-on reset	System control	1	I	4-1/4-1
$\overline{\text{HRESET}}$	—	Hard reset	System control	1	I/O	4-1/4-1
$\overline{\text{SRESET}}$	—	Soft reset	System control	1	I/O	4-1/4-1

Table 3-2 lists the signals in alphabetical order.

Table 3-2. MPC8360E Alphabetical Signal Reference

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
CE_PA[0:31]	—	QUICC Engine parallel port A	QUICC Engine	32	I/O	3-16/3-31
CE_PB[0:27]	—	QUICC Engine parallel port B	QUICC Engine	28	I/O	3-17/3-39
CE_PC[0:29]	—	QUICC Engine parallel port C	QUICC Engine	3032	I/O	3-18/3-47
CE_PD[0:27]	—	QUICC Engine parallel port D	QUICC Engine	28	I/O	3-19/3-53
CE_PE[0:31]	—	QUICC Engine parallel port E	QUICC Engine	32	I/O	3-20/3-60
CE_PF[0:3]	—	QUICC Engine parallel port F	QUICC Engine	4	I/O	3-21/3-69
CLK_IN	—	Clock input	Clocks	1	I	4-2/4-3
HRESET	—	Hard reset	System control	1	I/O	4-4/4-5
IIC1_SCL	—	I2C serial clock	I2C1	1	I/O	15-2/15-4
IIC1_SDA	—	I2C serial data	I2C1	1	I/O	15-2/15-4
IIC2_SCL	—	I2C serial clock	I2C2	1	I/O	15-2/15-4
IIC2_SDA	—	I2C serial data	I2C2	1	I/O	15-2/15-4
$\overline{\text{IRQ}}[0]/$ MCP_IN	$\overline{\text{IRQ}}[0]$	External interrupt 0	IPIC	1	I	8-2/8-5
	$\overline{\text{MCP_IN}}$	Machine check interrupt input	IPIC		I	8-2/8-5
$\overline{\text{IRQ}}[1]/$ M1SRCID[4]/ M2SRCID[4]/ LSRCID[4]	$\overline{\text{IRQ}}[1]$	External interrupt 1	IPIC	1	I	8-2/8-5
	M1SRCID[4]	DDR memory debug source port ID 4	DDR		O	9-7/9-13
	M2SRCID[4]	Secondary DDR memory debug source port ID 4	DDR		O	9-7/9-13
	LSRCID[4]	LBC debug source port ID 4	LBC		O	10-2/10-5
$\overline{\text{IRQ}}[2]/$ M1DVAL/ M2DVAL/ LDVAL	$\overline{\text{IRQ}}[2]$	External interrupt 2	IPIC	1	I	8-2/8-5
	M1DVAL	DDR memory debug data valid	DDR		O	9-7/9-13
	M2DVAL	Secondary DDR memory debug data valid	DDR		O	9-7/9-13
	LDVAL	LBC debug data valid	LBC		O	10-2/10-5
$\overline{\text{IRQ}}[3]/$ CORE_SRESET	$\overline{\text{IRQ}}[3]$	External interrupt 3	IPIC	1	I	8-2/8-5
	$\overline{\text{CORE_SRESET}}$	Core soft reset	Core	1	I	
$\overline{\text{IRQ}}[4:5]$	—	External interrupt 4–5	IPIC	1	I	8-2/8-5

Table 3-2. MPC8360E Alphabetical Signal Reference (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
$\overline{\text{IRQ}}[6]/$ $\overline{\text{LCS}}[6]/$ $\overline{\text{CKSTOP_OUT}}$	$\overline{\text{IRQ}}[6]$	External interrupt 6	IPIC	1	I	8-2/8-5
	$\overline{\text{LCS}}[6]$	LBC chip select 6	LBC		O	10-2/10-5
	$\overline{\text{CKSTOP_OUT}}$	Checkstop output	CPU		O	4-5/4-10
$\overline{\text{IRQ}}[7]/$ $\overline{\text{LCS}}[7]/$ $\overline{\text{CKSTOP_IN}}$	$\overline{\text{IRQ}}[7]$	External interrupt 7	IPIC	1	I	8-2/8-5
	$\overline{\text{LCS}}[7]$	LBC chip select 7	LBC		O	10-2/10-5
	$\overline{\text{CKSTOP_IN}}$	Checkstop input	CPU		I	4-5/4-10
LA[27:31]	—	LBC port address 27–31	LBC	5	O	10-2/10-5
LAD[0:31]	—	LBC address/data	LBC	32	I/O	10-2/10-5
LALE	—	LBC address latch enable	LBC	1	O	10-2/10-5
LBCTL	—	LBC data buffer control	LBC	1	O	10-2/10-5
LCKE	—	LBC clock enable	LBC	1	O	10-2/10-5
LCLK[0]	—	LBC clocks 0	LBC	1	O	10-2/10-5
LCLK[1]/ $\overline{\text{LCS}}[6]$	LCLK[1]	LBC clocks 1	LBC	1	O	10-2/10-5
	$\overline{\text{LCS}}[6]$	LBC chip select 6				
LCLK[2]/ $\overline{\text{LCS}}[7]$	LCLK[2]	LBC clocks 2	LBC	1	O	10-2/10-5
	$\overline{\text{LCS}}[7]$	LBC chip select 7				
$\overline{\text{LCS}}[0:5]$	—	LBC chip select 0–5	LBC	6	O	10-2/10-5
LDP[0]/ $\overline{\text{CKSTOP_OUT}}$	LDP[0]	LBC data parity 0	LBC	1	I/O	10-2/10-5
	$\overline{\text{CKSTOP_OUT}}$	Checkstop out	CPU		O	4-3/4-4
LDP[1]/ $\overline{\text{CKSTOP_IN}}$	LDP[1]	LBC data parity 1	LBC	1	I/O	10-2/10-5
	$\overline{\text{CKSTOP_IN}}$	Checkstop in	CPU		I	4-3/4-4
LDP[2]/ $\overline{\text{LCS}}[6]$	LDP[2]	LBC data parity 2	LBC	1	I/O	10-2/10-5
	$\overline{\text{LCS}}[6]$	LBC chip select 6			O	10-2/10-5
LDP[3]/ $\overline{\text{LCS}}[7]$	LDP[3]	LBC data parity 3	LBC	1	I/O	10-2/10-5
	$\overline{\text{LCS}}[7]$	LBC chip select 7			O	10-2/10-5
LGPL0/LSDA10/ cfg_reset_source0	LGPL0	LBC UPM general purpose line 0	LBC	1	O	10-2/10-5
	LSDA10	SDRAM address bit 10	LBC		O	10-2/10-5
	cfg_reset_source0	Configuration reset source 0	Reset & clock		I	4-5/4-10
LGPL1/ $\overline{\text{LSDWE}}$ / cfg_reset_source1	LGPL1	LBC UPM general purpose line 1	LBC	1	O	10-2/10-5
	$\overline{\text{LSDWE}}$	SDRAM write enable	LBC		O	10-2/10-5
	cfg_reset_source1	Configuration reset source 1	Reset & clock		I	4-7/4-11

Table 3-2. MPC8360E Alphabetical Signal Reference (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
LGPL2/ $\overline{\text{LSDRAS}}$ / $\overline{\text{LOE}}$	LGPL2	LBC UPM general purpose line 2	LBC	1	O	10-2/10-5
	$\overline{\text{LSDRAS}}$	SDRAM RAS			O	10-2/10-5
	$\overline{\text{LOE}}$	LBC output enable			O	10-2/10-5
LGPL3/ $\overline{\text{LSDCAS}}$ /cfg_r reset_source2	LGPL3	LBC UPM general purpose line 3	LBC		O	10-2/10-5
	$\overline{\text{LSDCAS}}$	SDRAM CAS	LBC		O	10-2/10-5
	cfg_reset_source2	Configuration reset source 2	Reset & clock		I	4-5/4-10
LGPL4/ $\overline{\text{LGTA}}$ / LUPWAIT/LPBSE	LGPL4	LBC UPM general purpose line 4	LBC	1	I/O	10-2/10-5
	$\overline{\text{LGTA}}$	GPCM terminate access				
	LUPWAIT	UPM wait				
	LPBSE	LBC parity byte select				
LGPL5/ cfg_clkdiv	LGPL5	LBC UPM general purpose line 5	LBC	1	O	10-2/10-5
	cfg_clkdiv	Configuration clock in div	Reset & clock		I	4-7/4-11
LSYNC_IN	—	LBC DLL synchronization input	LBC	1	I	10-2/10-5
LSYNC_OUT	—	LBC DLL synchronization output	LBC	1	O	10-2/10-5
$\overline{\text{LWE}}[0:3]$ / LSDDQM[0:3]/ $\overline{\text{LBS}}[0:3]$	$\overline{\text{LWE}}[0:3]$	LBC write enable 0–3	LBC	4	O	10-2/10-5
	LSDDQM[0:3]	Byte lane data mask 0–3			O	3-4/3-19
	$\overline{\text{LBS}}[0:3]$	Byte select 0–3			O	10-2/10-5
M66EN/CE_PF[4]	M66EN	PCI 66-MHz timing on/off.	PCI	1	I	13-3/13-5
	CE_PF[4]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
$\overline{\text{MCP_OUT}}$	—	Machine check interrupt output	IPIC	1	O	8-2/8-5
MEMC1_MA[14:0]	—	Primary DDR address	DDR	15	O	9-3/9-6
MEMC1_MBA[0:2]	—	Primary DDR bank select	DDR	3	O	9-3/9-6
$\overline{\text{MEMC1_MCAS}}$	—	Primary DDR column address strobe	DDR	1	O	9-3/9-6
MEMC1_MCK[0:1]	—	Primary DDR differential clocks (positive)	DDR	2	O	9-4/9-9
$\overline{\text{MEMC1_MCK}}[0:1]$	—	Primary DDR differential clocks (negative)	DDR	2	O	9-4/9-9
MEMC1_MCK[2:3]/ MEMC2_MCK[0:1]	MEMC1_MCK[2:3]	Primary DDR differential clocks (positive)	DDR	2	O	9-4/9-9
	MEMC2_MCK[0:1]	Secondary DDR differential clocks (positive)	DDR		O	9-4/9-9

Table 3-2. MPC8360E Alphabetical Signal Reference (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
MEMC1_MCK[2:3]/ MEMC2_MCK[0:1]	MEMC1_MCK[2:3]	Primary DDR differential clocks (negative)	DDR	2	O	9-4/9-9
	MEMC2_MCK[0:1]	Secondary DDR differential clocks (negative)	DDR		O	9-4/9-9
MEMC1_MCK[4:5]/ MEMC2_MCKE[0:1]	MEMC1_MCK[4:5]	Primary DDR differential clocks (positive)	DDR	2	O	9-4/9-9
	MEMC2_MCKE[0:1]	Secondary DDR clock enable			O	9-4/9-9
MEMC1_MCK[4]/ MEMC2_MDM[8]	MEMC1_MCK[4]	Primary DDR differential clocks (negative)	DDR	1	O	9-4/9-9
	MEMC2_MDM[8]	Secondary DDR ECC data mask	DDR		O	9-4/9-9
MEMC1_MCK[5]/ MEMC2_MDQS[8]	MEMC1_MCK[5]	Primary DDR differential clocks (negative)	DDR	1	O	9-4/9-9
	MEMC2_MDQS[8]	Secondary DDR data strobe	DDR		I/O	9-4/9-9
MEMC1_MCKE[0:1]	—	Primary DDR clock enable	DDR	2	O	9-4/9-9
MEMC1_MCS[0:1]	—	Primary DDR chip select (2/DIMM)	DDR	2	O	9-3/9-6
MEMC1_MDIC[0:1]	—	DRAM driver impedance calibration	DDR	2	I/O	9-3/9-6
MEMC1_MCS[2:3]/ MEMC2_MCS[0:1]	MEMC1_MCS[2:3]	Primary DDR chip select (2/DIMM)	DDR	2	O	9-3/9-6
	MEMC2_MCS[0:1]	Secondary DDR chip select (2/DIMM)			O	9-3/9-6
MEMC1_MDM[0:3]	—	Primary DDR data mask 0–3	DDR	4	O	9-3/9-6
MEMC1_MDM[4:7]/ MEMC2_MDM[0:3]	MEMC1_MDM[4:7]	Primary DDR data mask 4–7	DDR	4	O	9-3/9-6
	MEMC2_MDM[0:3]	Secondary DDR data mask 0–3			O	9-3/9-6
MEMC1_MDM[8]	—	Primary DDR ECC data mask	DDR	1	O	9-3/9-6
MEMC1_MDQ[0:31]	—	Primary DDR data	DDR	32	I/O	9-3/9-6
MEMC1_MDQ[32:63]/ MEMC2_MDQ[0:31]	MEMC1_MDQ[32:63]	Primary DDR data	DDR	32	I/O	9-3/9-6
	MEMC2_MDQ[0:31]	Secondary DDR data	DDR	32	I/O	9-3/9-6
MEMC1_MDQS[0:3]	—	Primary DDR data strobe	DDR	4	I/O	9-3/9-6
MEMC1_MDQS[4:7]/ MEMC2_MDQS[0:3]	MEMC1_MDQS[4:7]	Primary DDR data strobe	DDR	4	I/O	9-3/9-6
	MEMC2_MDQS[0:3]	Secondary DDR data strobe			I/O	9-3/9-6
MEMC1_MDQS[8]	—	Primary DDR data strobe	DDR	1	I/O	9-3/9-6
MEMC1_MECC[0:4]	MEMC1_MSRCID[0:4]	Primary DDR error correcting code	DDR	5	I/O	9-3/9-6

Table 3-2. MPC8360E Alphabetical Signal Reference (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
MEMC1_MECC[5]	MEMC1_MDVAL	Primary DDR error correcting code	DDR	1	I/O	9-3/9-6
MEMC1_MECC[6:7]	—	Primary DDR error correcting code	DDR	3	I/O	9-3/9-6
MEMC1_MODT[0:3]	—	On-die termination for DRAM chip	DDR	4	O	9-3/9-6
MEMC1_MRAS	—	Primary DDR row address strobe	DDR	1	O	9-3/9-6
MEMC1_MWE	—	Primary DDR write enable	DDR	1	O	9-3/9-6
MEMC2_MA[14:0]	—	Secondary DDR address	DDR	15	O	9-3/9-6
MEMC2_MBA[0:2]	—	Secondary DDR bank select	DDR	3	O	9-3/9-6
MEMC2_MCAS	—	Secondary DDR column address strobe	DDR	1	O	9-3/9-6
MEMC2_MECC[0:4]	MEMC2_MSRCID[0:4]	Secondary DDR error correcting code	DDR	5	I/O	9-3/9-6
MEMC2_MECC[5]	MEMC2_MDVAL	Secondary DDR error correcting code	DDR	1	I/O	9-3/9-6
MEMC2_MECC[6:7]	—	Secondary DDR error correcting code	DDR	3	I/O	9-3/9-6
MEMC2_MRAS	—	Secondary DDR row address strobe	DDR	1	O	9-3/9-6
MEMC1_MWE	—	Primary DDR write enable	DDR	1	O	9-3/9-6
PCI_AD[31:0]/ CE_PG[31:0]	PCI_AD[31:0]	PCI address/data	PCI	32	I/O	13-3/13-5
	CE_PG[31:0]	QUICC Engine parallel I/O	PIO		I/O	3-22/3-77
PCI_C/ $\overline{\text{BE}}$ [3:0]/ CE_PF[10:7]	PCI_C	PCI command	PCI	4	I/O	13-3/13-5
	$\overline{\text{BE}}$ [3:0]	Byte enable	PCI		I/O	13-3/13-5
	CE_PF[10:7]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_CLK_OUT[0:2] CE_PF[26:28]	PCI_CLK_OUT[0:2]	PCI clock out 0-2	Clocks	3	O	4-4/4-5
	CE_PF[26:28]	QUICC Engine port F	QUICC Engine		I/O	3-21/3-69
PCI_CLOCK/ PCI_SYNC_IN	PCI_CLOCK	PCI clock	Clocks	1	I	4-2/4-3
	PCI_SYNC_IN	PCI clock sync input	Clocks			
$\overline{\text{PCI_DEVSEL}}$ / CE_PF[16]	$\overline{\text{PCI_DEVSEL}}$	PCI device select	PCI	1	I/O	13-3/13-5
	CE_PF[16]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
$\overline{\text{PCI_FRAME}}$ / CE_PF[12]	$\overline{\text{PCI_FRAME}}$	PCI frame	PCI	1	I/O	13-3/13-5
	CE_PF[12]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69

Table 3-2. MPC8360E Alphabetical Signal Reference (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
PCI_GNT[0]/ CE_PF[23]	PCI_GNT[0]	PCI grant 0	PCI	1	I/O	13-3/13-5
	CE_PF[23]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_GNT[1]/ CPCI_HS_LED/ CE_PF[24]	PCI_GNT[1]	PCI grant 1	PCI	1	O	13-3/13-5
	CPCI_HS_LED	Compact PCI hot swap LED	CPCI		O	13-3/13-5
	CE_PF[24]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_GNT[2]/ CPCI_HS_ENUM/ CE_PF[25]	PCI_GNT[2]	PCI grant 2	PCI	1	O	13-3/13-5
	CPCI_HS_ENUM	Compact PCI hot swap enumerator	CPCI		O	13-3/13-5
	CE_PF[25]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_IDSEL/ CE_PF[17]	PCI_IDSEL	PCI initial device select	PCI	1	I	13-3/13-5
	CE_PF[17]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_INTA/IRQ_OUT/ CE_PF[5]	PCI_INTA	PCI interrupt output	PCI	1	O	13-3/13-5
	IRQ_OUT	Interrupt output	IPIC		O	13-3/13-5
	CE_PF[5]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_IRDY/ CE_PF[14]	PCI_IRDY	PCI initiator ready	PCI	1	I/O	13-3/13-5
	CE_PF[14]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_MODE	—	PCI mode select	PCI	1	I	5-32/5-24
PCI_PAR/CE_PF[11]	PCI_PAR	PCI parity	PCI	1	I/O	13-3/13-5
	CE_PF[11]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_PERR/ CE_PF[19]	PCI_PERR	PCI parity error	PCI	1	I/O	13-3/13-5
	CE_PF[19]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_REQ[0]/ CE_PF[20]	PCI_REQ[0]	PCI request 0	PCI	1	I/O	13-3/13-5
	CE_PF[20]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_REQ[1]/ CPCI_HS_ES/ CE_PF[21]	PCI_REQ[1]	PCI request 1	PCI	1	I	13-3/13-5
	CPCI_HS_ES	Compact PCI hot swap ejector switch	CPCI		I	13-3/13-5
	CE_PF[21]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_REQ[2]/ CE_PF[22]	PCI_REQ[2]	PCI request 2	PCI	1	I	13-3/13-5
	CE_PF[22]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_RESET_OUT/ CE_PF[6]	PCI_RESET_OUT	PCI reset	PCI	1	O	13-3/13-5
	CE_PF[6]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_SERR/ CE_PF[18]	PCI_SERR	PCI system error	PCI	1	I/O	13-3/13-5
	CE_PF[18]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69

Table 3-2. MPC8360E Alphabetical Signal Reference (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
PCI_STOP/ CE_PF[15]	$\overline{\text{PCI_STOP}}$	PCI stop	PCI	1	I/O	13-3/13-5
	CE_PF[15]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_SYNC_IN/ PCI_CLOCK	PCI_SYNC_IN	PCI clock sync input	Clocks	1	I	4-4/4-5
	PCI_CLOCK	PCI clock	Clocks			
PCI_SYNC_OUT/ CE_PF[29]	PCI_SYNC_OUT	PCI clock sync output	Clocks	1	O	4-4/4-5
	CE_PF[29]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
PCI_TRDY/ CE_PF[13]	$\overline{\text{PCI_TRDY}}$	PCI target ready	PCI	1	I/O	13-3/13-5
	CE_PF[13]	QUICC Engine parallel I/O	PIO		I/O	3-21/3-69
$\overline{\text{PORESET}}$	—	Power-on reset	System control	1	I	4-3/4-4
$\overline{\text{QUIESCE}}$	—	Quiesce state	PMC	1	O	5-77/5-73
$\overline{\text{SRESET}}$	—	Soft reset	System control	1	I/O	4-3/4-4
TCK	—	Test clock	JTAG	1	I	17-2/17-2
TDI	—	Test data in	JTAG	1	I	17-2/17-2
TDO	—	Test data out	JTAG	1	O	17-2/17-2
TEST_SEL (MPC8360) $\overline{\text{TEST_SEL}}$ (MPC8358)	—	Test mode	TEST	1	I	—
TEST	—	Test mode	TEST	1	I	—
TMS	—	Test mode select	JTAG	1	I	17-2/17-2
$\overline{\text{TRST}}$	—	Test reset	JTAG	1	I	17-2/17-2
$\overline{\text{UART1_CTS}}$ / M1SRCID[2]/ M2SRCID[2] LSRCID[2]	$\overline{\text{UART1_CTS}}$	UART1 clear to send	UART	1	I	16-2/16-3
	M1SRCID2	DDR memory debug source port ID 2	DDR		O	9-7/9-13
	M2SRCID2	Secondary DDR memory debug source port ID 2	DDR		O	9-7/9-13
	LSRCID2	LBC debug source port ID 2	LBC		O	10-2/10-5
$\overline{\text{UART1_RTS}}$ M1SRCID[3]/ M2SRCID[3] LSRCID[3]	$\overline{\text{UART1_RTS}}$	UART1 ready to send	UART	1	O	16-2/16-3
	M1SRCID3	DDR memory debug source port ID 3	DDR		O	9-7/9-13
	M2SRCID3	Secondary DDR memory debug source port ID 3	DDR		O	9-7/9-13
	LSRCID3	LBC debug source port ID 3	LBC		O	10-2/10-5

Table 3-2. MPC8360E Alphabetical Signal Reference (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
UART1_SIN/ M1SRCID[1]/ M2SRCID[1]/ LSRCID[1]	UART1_SIN	UART1 serial data in	UART	1	I	16-2/16-3
	M1SRCID[1]	DDR memory debug source port ID 1	DDR		O	9-7/9-13
	M2SRCID[1]	Secondary DDR memory debug source port ID 1	DDR		O	9-7/9-13
	LSRCID[1]	LBC debug source port ID 1	LBC		O	10-2/10-5
UART1_SOUT/ M1SRCID[0]/ M2SRCID[0]/ LSRCID[0]	UART1_SOUT	UART1 serial data out	UART	1	O	16-2/16-3
	M1SRCID[0]	DDR memory debug source port ID 0	DDR		O	9-7/9-13
	M2SRCID[0]	Secondary DDR memory debug source port ID 0	DDR		O	9-7/9-13
	LSRCID[0]	LBC debug source port ID 0	LBC		O	10-2/10-5

3.2 Configuration Signals Sampled at Reset

Signals that serve alternate functions as configuration input signals during system reset are summarized in [Table 3-3](#). A detailed interpretation of their voltage levels during reset is described in [Chapter 4, “Reset, Clocking, and Initialization.”](#)

Table 3-3. MPC8360E Reset Configuration Signals

Functional Interface	Functional Signal Name	Reset Configuration Name
LBC	LSDA10/LGPL0	CFG_RESET_SOURCE0
	$\overline{\text{LSDWE}}/\text{LGPL1}$	CFG_RESET_SOURCE1
	$\overline{\text{LSDCAS}}/\text{LGPL3}$	CFG_RESET_SOURCE2
	LGPL5	CFG_CLKIN_DIV

3.3 Output Signal States During Reset

When a system reset is recognized ($\overline{\text{PORESET}}$ or $\overline{\text{HRESET}}$ are asserted), the MPC8360E aborts all current internal and external transactions and releases all bidirectional I/O signals to a high-impedance state. See [Chapter 4, “Reset, Clocking, and Initialization,”](#) for a complete description of the reset functionality. During reset, the MPC8360E ignores most input signals (except for the reset configuration signals) and drives most of the output-only signals to an inactive state. [Table 3-4](#) shows the states of the output-only signals.

Table 3-4. Output Signal States During System Reset

Interface	Signal	State During Reset
MEMCx_MDM[0:8]	DDR data mask	All '0'
MEMCx_MBA[0:1]	DDR bank select	All 'Z'
MEMCx_MA[14:0]	DDR address	All 'Z'
$\overline{\text{MEMCx_MWE}}$	DDR write enable	'Z'
$\overline{\text{MEMCx_MRAS}}$	DDR row address strobe	'Z'
$\overline{\text{MEMCx_MCAS}}$	DDR column address strobe	'Z'
$\overline{\text{MEMCx_MCS}}$ [0:3]	DDR chip select (2/DIMM)	All 'Z'
MEMCx_MCKE[0:1]	DDR clock enable	All '0'
MEMCx_MCK[0:5]	DDR differential clocks	All '0'
$\overline{\text{MEMCx_MCK}}$ [0:5]	DDR differential clocks	All '0'
MEMC1_MODT[0:3]	DRAM on-die termination	All '0'
$\overline{\text{INTA/}}/$ $\overline{\text{IRQ_OUT}}$	PCI interrupt output	High impedance
$\overline{\text{PCI_RESET_OUT}}$	PCI reset output	'0'
$\overline{\text{PCI_GNT}}$ [0:2]	PCI grant 0–2	High impedance
UART1_SOUT	UART1 serial data out	'1'
$\overline{\text{UART1_RTS}}$	UART1 ready to send	'1'
LA[27:31]	LBC port address	Active – being used to load reset configuration word
$\overline{\text{LCS}}$ [0]	LBC chip select 0	Active – being used to load reset configuration word
$\overline{\text{LCS}}$ [1:7]	LBC chip select 1–7	'1111111'
$\overline{\text{LWE}}$ [0:3]/ LSDQM[0:3]/ $\overline{\text{LBS}}$ [0:3]	LBC write enable/byte lane data mask/byte select	'1111'
LBCTL	LBC data buffer control	Active – being used to load reset configuration word
LALE	LBC address latch enable	Active – being used to load reset configuration word
$\overline{\text{LGPL2/LSDRAS/}}/$ $\overline{\text{LOE}}$	LBC output enable/SDRAM RAS/GP line 2	Active – being used to load reset configuration word
LCKE	LBC clock enable	'1'
LCLK[0:2]	LBC clock	'0'
LSYNC_OUT	LBC clock synchronization output	'0'
$\overline{\text{MCP_OUT}}$	Machine check interrupt output	High impedance
TDO	Test data out	High impedance
$\overline{\text{QUIESCE}}$	Quiesce state	'1'

Table 3-4. Output Signal States During System Reset (continued)

Interface	Signal	State During Reset
PCI_CLK_OUT[0:2]	PCI clock output 0–2	All '0'
PCI_SYNC_OUT	PCI sync output	Active clock

Table 3-5 provides the list of signals and registers that control the the chip’s signal multiplexing.

Table 3-5. Signals for Multiplexing

Signal Group	Muxing is Controlled By	Table/Page
PCI (CE_PF[4:29] & CE_PG[0:31])	Signal $\overline{\text{PCI_MODE}}$	5-32/5-24
PCI clock outputs (CE_PF[26:28])	RCWH[PCICKDRV] (In addition to $\overline{\text{PCI_MODE}}$ as described above)	4-22/4-23
PCI/CPCI	RCWH[PCIARB] (In addition to $\overline{\text{PCI_MODE}}$ as described above)	4-13/4-17
QUICC Engine block	CPPARxA–CPPARxG	3-9/3-24
All others	SICRL/SICRH	5-40/5-29 5-41/5-31

3.4 Parallel I/O Ports

The QUICC Engine block supports 7 general-purpose I/O ports: ports A, B, C, D, E, F, and G. Each pin in the I/O ports can be configured as a general-purpose I/O signal or as a dedicated peripheral interface signal. Each pin can be configured as open-drain (the pin can be configured in a wired-OR configuration on the board). The pin drives a zero voltage but three-states when driving a high voltage.

Note that port pins do not have an internal pull-up resistor. Due to the QUICC Engine block’s significant flexibility, many dedicated peripheral functions are multiplexed onto the ports. The functions are grouped to maximize the pins’ usefulness in the greatest number of MPC8360 applications. The reader may not obtain a full understanding of the pin assignment capability described in this chapter without understanding the QUICC Engine peripherals.

3.4.1 Features

The following is a list of the parallel I/O ports’ important features:

- Ports A, E, and G have 32 pins.
- Ports C and F have 30 pins.
- Ports B and D have 28 pins.
- All ports are bidirectional.
- All ports have alternate on-chip peripheral functions.
- Each pin has four direction options: Input, Output, I/O and Disabled. A disabled pin is not driving any value and can be left floating outside the device (no need for external pull up or pull down).
- All port pins are disabled at hard reset.

- Usually pin values can be read while the pin is connected to an on-chip peripheral (this is not possible for port pins, which are used as PCI pins).
- Open-drain capability on all port pins
- Some of the port pins can be used as interrupt input pins.

3.4.2 Port Registers

Each port has six memory-mapped, read/write, 32-bit control registers that determine its characteristics. A total of 6 bits are used for each pin.¹

The port registers are: CPODR_x, CPDAT_x, CPDIR1_x, CPDIR2_x, CPPAR1_x and CPPAR2_x.

- The CPODR_x registers (one bit per pin—bit number corresponds to pin number) determine the open-drain configuration for each pin.
- The CPDAT_x registers (one bit per pin—bit number corresponds to pin number) are used to read/write data from/to the pins.
- The CPDIR1_x and CPDIR2_x registers (two bits per pin) determine the in/out characteristics of the pin.
- The CPPAR1_x and CPPAR2_x registers (two bits per pin) determine the functionality of each pin, according to the respective pin assignment table.

3.4.2.1 QUICC Engine Port Open-Drain Registers (CPODRA–CPODRG)

The QUICC Engine port open-drain register (CPODR), shown in Figure 3-2 and Table 3-6, indicates a normal or wired-OR configuration of the port pins.

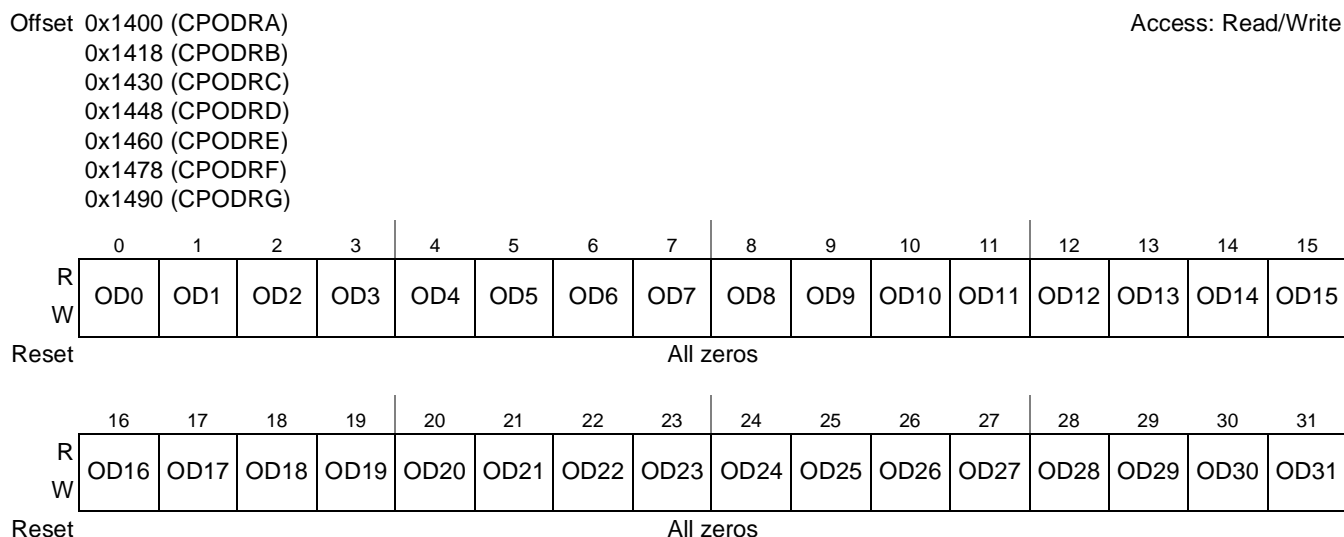


Figure 3-2. QUICC Engine Port Open-Drain Registers (CPODRA–CPODRG)

1. Note that for ports with less than 32 pins, only the relevant bits are used. For example, for port C, which has 30 pins, CPODRC(OD30–OD31), CPDATC(D30–D31), CPDIR2C(DIR30–DIR31= bits 28 to 31), and CPPAR2C(SEL30–SEL31 = bits 28 to 31) are not used.

Table 3-6. CPODRx Field Descriptions

Bits	Name	Description
0–31	ODx	Open-drain configuration. Determines whether the corresponding pin is actively driven as an output or is an open-drain driver. 0 The I/O pin is actively driven as an output. 1 The I/O pin is an open-drain driver. As an output, the pin is driven active-low, otherwise it is three-stated.

3.4.2.2 QUICC Engine Port Data Registers (CPDATA–CPDATG)

A read of a QUICC Engine port data register (CPDAT_x), shown in Figure 3-3 and Table 3-7, returns the data at the pin, independent of whether the pin is defined as an input or output. This allows detection of output conflicts at the pin by comparing the written data with the data on the pin.

A write to the CPDAT_x is latched and if the corresponding CPDIR bit is configured as an output, the value latched for that bit is driven onto its respective pin. CPDAT_x can be read or written at any time and is not initialized.

If a port pin is selected as a general purpose I/O pin, it can be accessed through the QUICC Engine port data register (CPDAT_x). Data written to the CPDAT_x is stored in an output latch. If a port pin is configured as an output, the output latch data is gated onto the port pin. In this case, when CPDAT_x is read, the port pin itself is read. If a port pin is configured as an input, data written to CPDAT_x is still stored in the output latch, but is prevented from reaching the port pin. In this case, when CPDAT_x is read, the state of the port pin is read.

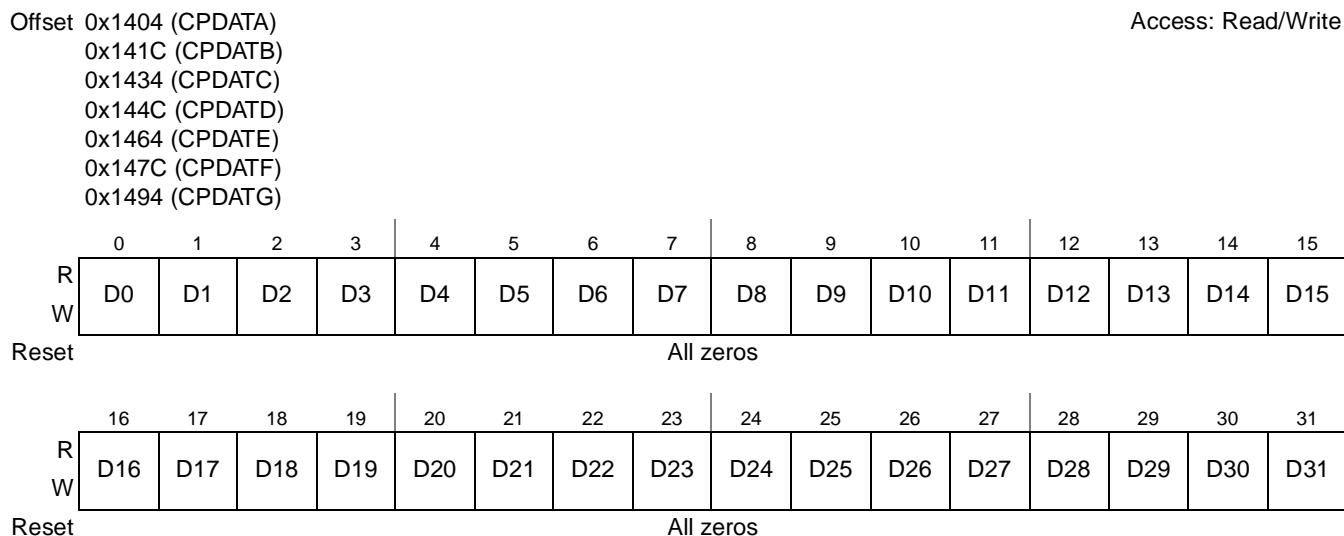


Figure 3-3. QUICC Engine Port Data Registers (CPDATA–CPDATG)

Table 3-7. CPDATx Field Descriptions

Bits	Name	Description
0–31	Dx	Contains data for the respective pin.

3.4.2.3 QUICC Engine Port Direction Registers (CPDIRxA–CPDIRxG)

The QUICC Engine port direction registers determine the direction of each port pin, shown in [Figure 3-4](#), [Figure 3-5](#), and [Table 3-8](#). There are four possible direction options per port pin.

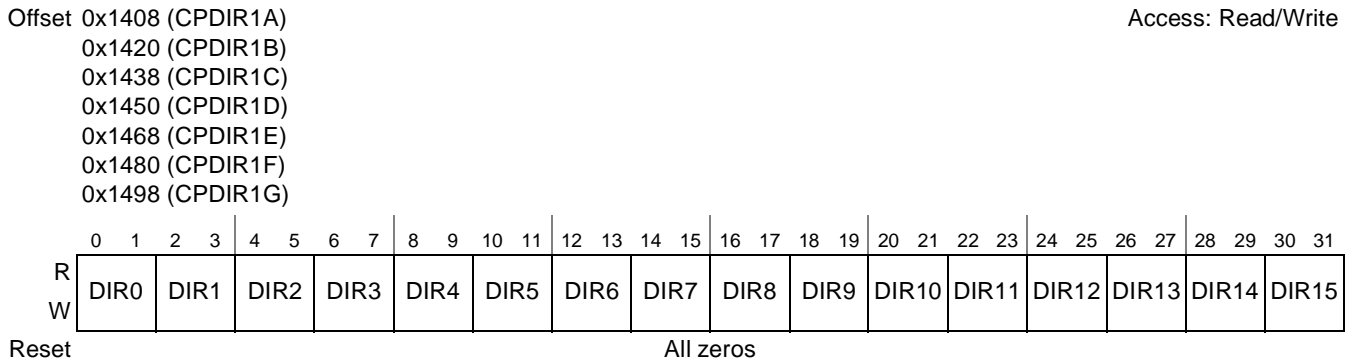


Figure 3-4. QUICC Engine Port Direction 1 Registers (CPDIR1A–CPDIR1G)

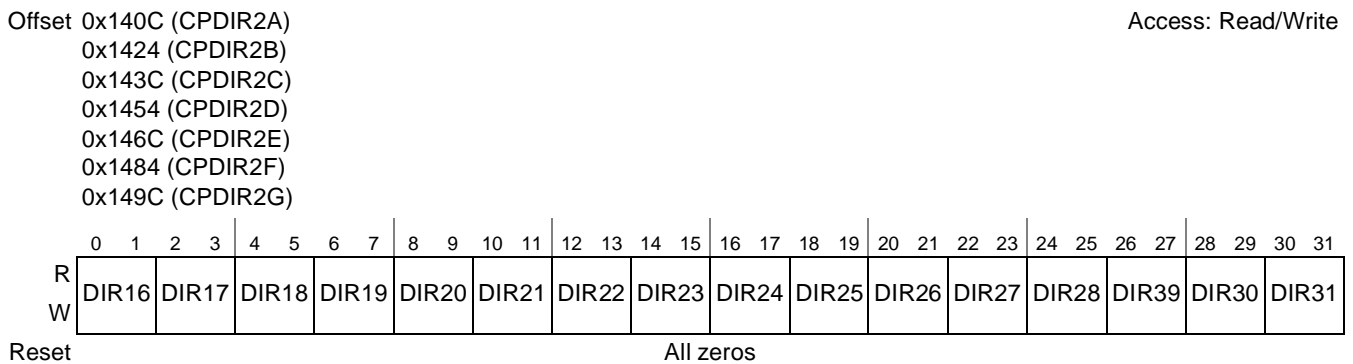


Figure 3-5. QUICC Engine Port Direction 2 Registers (CPDIR2A–CPDIR2G)

Table 3-8. CPDIRxy Field Descriptions

Bits	Name	Description
0–1, 2–3, etc.	DIR _n	Determines the I/O characteristics of pin number n 00 The corresponding pin is disabled (Output driver disabled, Input buffer disabled). 01 The corresponding pin is an output. 10 The corresponding pin is an input. 11 The corresponding pin is I/O.

3.4.2.4 QUICC Engine Port Pin Assignment Registers (CPPARxA–CPPARxG)

The QUICC Engine port pin assignment registers select the function of each port pin, shown in [Figure 3-6](#), [Figure 3-7](#), and [Table 3-9](#).

Signal Descriptions

Offset 0x1410 (CPPAR1A) Access: Read/Write
 0x1428 (CPPAR1B)
 0x1440 (CPPAR1C)
 0x1458 (CPPAR1D)
 0x1470 (CPPAR1E)
 0x1488 (CPPAR1F)
 0x14A0 (CPPAR1G)

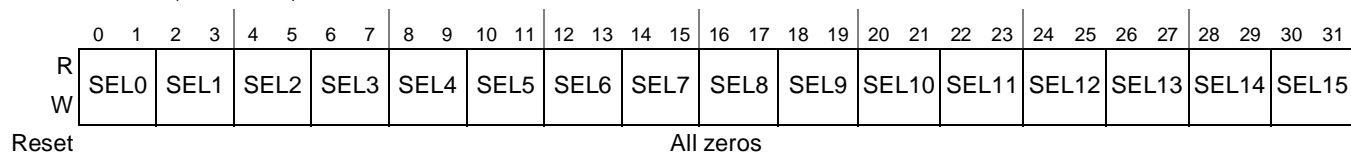


Figure 3-6. QUICC Engine Port Pin Assignment 1 Registers (CPPAR1A–CPPAR1G)

Offset 0x1414 (CPPAR2A) Access: Read/Write
 0x142C (CPPAR2B)
 0x1444 (CPPAR2C)
 0x145C (CPPAR2D)
 0x1474 (CPPAR2E)
 0x148C (CPPAR2F)
 0x14A4 (CPPAR2G)

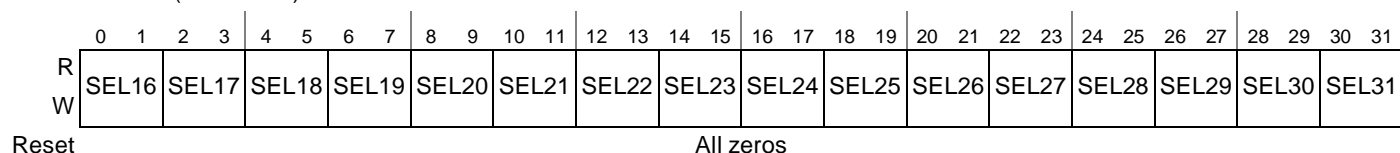


Figure 3-7. QUICC Engine Port Pin Assignment 2 Registers (CPPAR2A–CPPAR2G)

Table 3-9. CPPARxy Field Descriptions

Bits	Name	Description
0–1, 2–3, etc.	SEL n	Determines the function of pin number n , according to Table 3-16 through Table 3-22 below. Functions not shown in the table represent Reserved values of these bits.

3.4.2.5 Communication Peripherals to QUICC Engine Mux Control Registers

The following registers are programmed when the user needs to connect the interrupt line of an on-chip accelerator, or an external pin to the QUICC Engine External requests (EXT[1:4]).

The QUICC Engine port pin assignment registers select the function of each port pin.

Offset 0x014B8 (CPCE1R) Access: Read/Write
 0x014BC (CPCE2R)
 0x014C0 (CPCE3R)
 0x014C4 (CPCE4R)

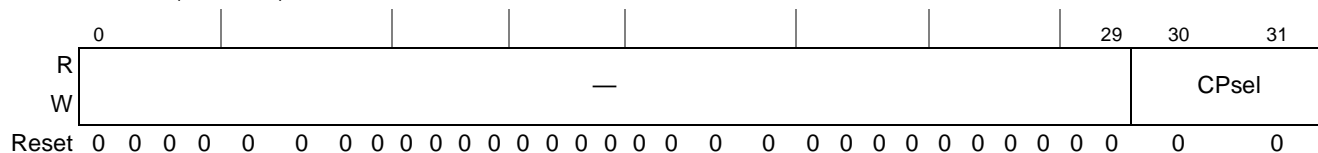


Figure 3-8. Communication Peripherals to QUICC Engine Mux Registers (CPCE1R–CPCE4R)

Table 3-10. CPCEXR Field Descriptions

Bits	Name	Description
0-29	—	Reserved. Must be cleared by the Host
30-31	CPsel	Communication Peripheral Select 00 - External pin CE-EXT_REQ[1:4] is connected to EXTx 01 - Reserved. 10 - Security Accelerator Interrupt is connected to EXTx 11 - Reserved.

3.4.2.6 QUICC Engine Port Output Hold Registers (CPOH1, CPOH2)

The QUICC Engine port output hold registers control the output delay of the QUICC Engine I/O pins. Each field in these registers can be used to adjust the output delay of a single output pin or group of output pins. Each field can be configured to one of four delay options. The output hold registers should remain in the default state including the reserved fields unless required by *Errata to MPC8360E PowerQUICC II Pro Integrated Host Processor Family Reference Manual, Rev. 2* (MPC8360ERMAD). If the default value is modified, the AC timings specified in *MPC8360E/MPC8358E PowerQUICC II Pro Processor Revision 2.x TBGA Silicon Hardware Specifications* may not be valid. The I/O delay registers are optimized for the fast Ethernet protocols, RGMII, GMII and TBI. [Figure 3-9](#) describes CPOH1, and [Figure 3-10](#) describes CPOH2.

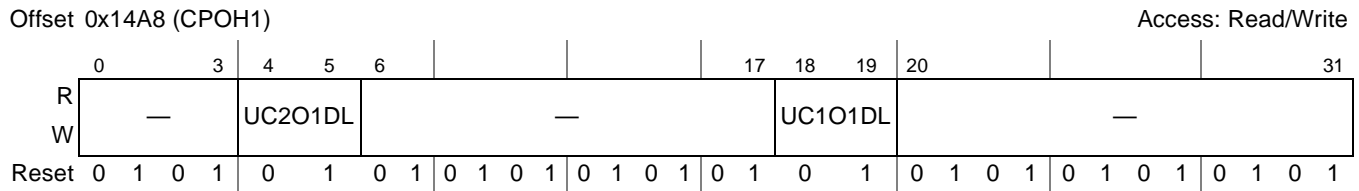


Figure 3-9. QUICC Engine Port Output Hold Register 1 (CPOH1)

[Table 3-11](#) defines the CPOH1 bit fields.

Table 3-11. CPOH1 Bit Setting

Bits	Name	Description	Reset Value
0-3	—	Reserved, default value should not be changed	0101
4-5	UC2O1DL	UCC2 Option 1 Delay configures the delay of the UCC1 output clock pin CE_PC2. The delay options are described in Table 3-13 .	01
6-17	—	Reserved, default value should not be changed	0x555
18-19	UC1O1DL	UCC1 Option 1 Delay configures the delay of the UCC1 output clock pin CE_PC9. The delay options are described in Table 3-13 .	01
20-31	—	Reserved, default value should not be changed	0x555

Signal Descriptions

Offset 0x14AC (CPOH2)

Access: Read/Write

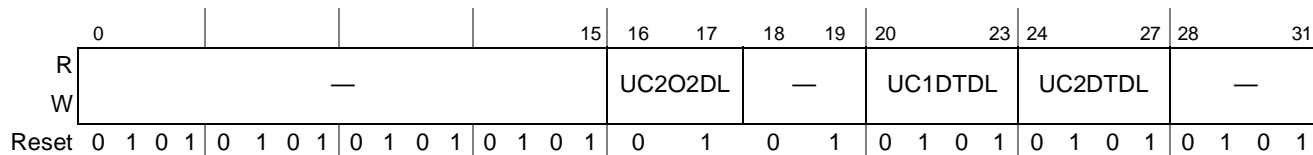


Figure 3-10. QUICC Engine Port Output Hold Register 2 (CPOH2)

Table 3-12 defines the CPOH2 bit fields.

Table 3-12. CPOH2 Bit Setting

Bits	Name	Description	Reset Value
0–15	—	Reserved, default value should not be changed	0x5555
16–17	UC2O2DL	UCC2 Option 2 Delay configures the delay of the UCC2 output clock pin PCI_CLK_OUT[0]/CE_PF26. The delay options are described in Table 3-13.	01
18–19	—	Reserved, default value should not be changed	01
20–23	UC1DTDL ¹	UCC1 Delay configures the delay of the UCC1 output data pins CE_PA3–8, CE_PC22–25. The delay options are described in Table 3-13.	0101
24–27	UC2DTDL	UCC2 Delay configures the delay of the UCC2 output data pins CE_PA17–22, CE_PB2–3, CE_PB5–10. The delay options are described in Table 3-13. Note: This only applies to the UCC2 Option 1 data pins.	0101
20–31	—	Reserved, default value should not be changed	0101

¹ Only valid for revision 2.1 silicon

Table 3-13 defines the output delay adjustments for the CPOH1 and CPOH2 register fields.

Table 3-13. Output Delay Options

Fields	Value	Typical Output Delay
UC1O1DL, UC2O1DL, UC2O2DL	00	–0.5 ns
	01 (default)	0 ns
	10	0.5 ns
	11	1.5 ns
UC1DTDL, UC2DTDL	0000	–0.5 ns
	0101 (default)	0 ns
	1010	0.25 ns
	1111	0.5 ns
	Others	Reserved

3.4.3 Port Block Diagram

Figure 3-11 shows the functional block diagram per one port pin.

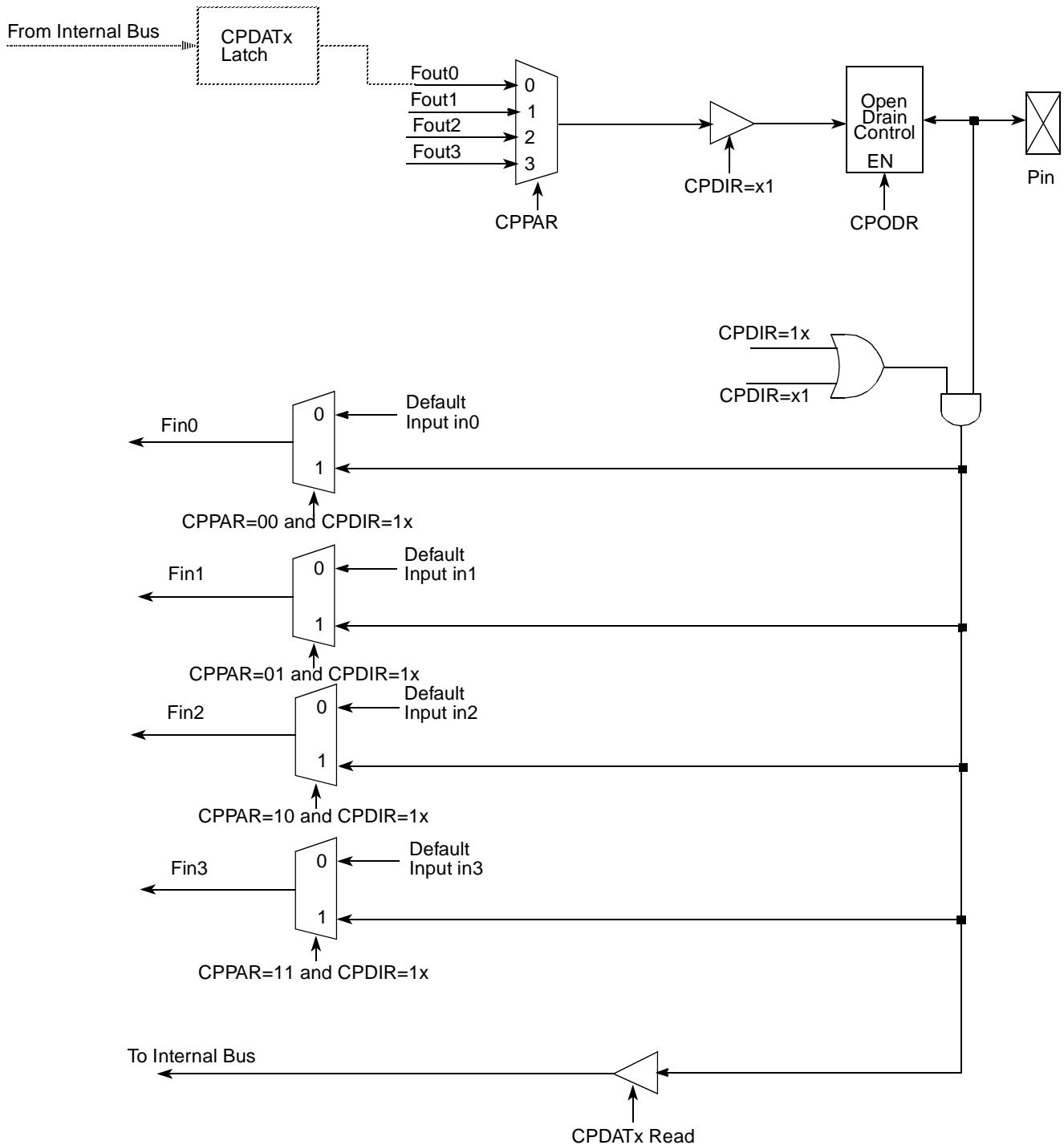


Figure 3-11. Port Functional Block Diagram

3.4.4 Port Pins Functions

Generally each pin can function as a general purpose I/O pin or as a dedicated input or output pin. Note however that some pins have only “General Purpose Input” or “General Purpose Output” function (not both possibilities). Refer to [Table 3-16](#) through [Table 3-22](#) for details. Usually following hard reset all port pins are disabled—both output and input buffers are off.

3.4.4.1 General-Purpose I/O Pins

Usually a value of 00 in CPPAR selects the general-purpose I/O function. The signal direction is determined by the value in the CPDIR. If a port pin is selected as a general-purpose I/O pin, it can be accessed through the port data register (CPDATx). Data written to the CPDATx is stored in an output latch. If a port pin is configured as an output, the output latch data is driven onto the port pin. In this case, when CPDATx is read, the port pin itself is read. If a port pin is configured as an input, data written to CPDATx is still stored in the output latch, but is prevented from reaching the port pin. In this case, when CPDATx is read, the state of the port pin is read.

3.4.4.2 Dedicated Pins

When a port is not configured as a general-purpose I/O pin, it has a dedicated functionality, as described in the following tables. Note that if an input to a peripheral is not supplied from a pin, a default value is supplied to the on-chip peripheral as listed in the “Default Input” column in the tables.

NOTE

Some output functions can be output on more than one pin. The user can freely configure such functions to be output on more than one pin at once. However, there is typically no advantage in doing so unless there is a large fanout where it is advantageous to share the load between several pins.

Many input functions can also come from two or three different pins; see [Section 3.4.8, “Ports Tables.”](#)

3.4.5 PCI Pins

Generally CPPAR and CPDIR bits are set to 0b00 following hard reset. Some of the port pins may function as PCI pins. Bits 0, 2–3 of the Reset Configuration Word High (PCIHOST, PCIARB, PCICKDRV) affect the reset values of CPPAR and CPDIR bits for these port pins. An external pin, PCI_MODE is a global PCI enable pin that enables full PCI functionality of the device immediately out of reset. When this pin is pulled low, the port pins that have PCI functionality are functioning properly out of reset (as inputs or outputs). The PCIARB bit is the PCI arbiter enable bit. When the PCI arbiter is enabled, while PCI_MODE is pulled low, port pins which are related to PCI arbiter (PCI_REQ, PCI_GNT) are functioning properly out of reset. In PCI Agent mode when the PCI arbiter is disabled, PF[22] is not used as a PCI function and is free to be used for other functions described in the Ports Tables, in addition port pins which relate to CompactPCI are functioning properly out of reset. In PCI Host mode, PF[17] is not used as a PCI function and is free to be used for other functions described in the Ports Tables, in addition if the arbiter is disabled all the PCI arbiter related pins are free to be used for other functions described in the Ports Tables. The

PCICKDRV bit affects the function of the PCI_CLK_OUT pins out of reset. Note that it is not possible to read the values on port pins which are functioning as PCI pins through the PDATx registers.

3.4.6 QUICC Engine Ports Interrupts

Twenty eight QUICC Engine port pins may be selected as the source of external interrupts. This is useful in communication interfaces that require interrupt handling.

See [Section 8.6.10, “QUICC Engine Ports Interrupts,”](#) for details on configuring these QUICC Engine ports interrupts.

3.4.7 Gigabit Ethernet Pins

3.4.7.1 RGMII pins

The MPC8360 supports two RGMII ports. For RGMII of UCC1 there is one set of pins. For RGMII of UCC2 there are two sets of pins which can be used. For RGMII of UCC2 the user should select which option to use and then program all the pins to conform to this selection.

Table 3-14. RGMII Pins

SIGNAL	UCC1 RGMII	UCC2 RGMII option 1	UCC2 RGMII option 2
CLK_IN	PC8	PC3	PC7
TxD[0]	PA3	PA17	PF5
TxD[1]	PA4	PA18	PF6
TxD[2]	PA5	PA19	PF21
TxD[3]	PA6	PA20	PG24
Tx_EN	PA7	PA21	PF22
GTX_CLK	PC9	PC2	PF26
RxD[0]	PA9	PA23	PF23
RxD[1]	PA10	PA24	PF24
RxD[2]	PA11	PA25	PG30
RxD[3]	PA12	PA26	PF25
Rx_DV	PA15	PA29	PG31
Rx_CLK	PA0	PA31	PF20

3.4.7.2 GMII and TBI pins

The MPC8360 supports two GMII or TBI ports. For GMII/TBI of UCC1 there is one set of pins. For TxD[4:7] of UCC1 there are two options for each signal. For GMII/TBI of UCC2 there is one set of pins which can be used. The user should select which option to use and then program all the pins to conform to this selection.

Table 3-15. GMII and TBI Pins

SIGNAL	UCC1 GMII/TBI	UCC2 GMII/TBI
CLK_IN	PC[8,9,10,11,14,15]	PC[2,3,6,7,15,16,17]
TxD[0]	PA3	PA17
TxD[1]	PA4	PA18
TxD[2]	PA5	PA19
TxD[3]	PA6	PA20
TxD[4]	PB6 or PC25	PB2
TxD[5]	PB7 or PC24	PB3
TxD[6]	PB9 or PC23	PB5
TxD[7]	PB10 or PC22	PB8
Tx_ERR	PA8	PA22
Tx_EN	PA7	PA21
GTX_CLK	PC9	PC2
RxD[0]	PA9	PA23
RxD[1]	PA10	PA24
RxD[2]	PA11	PA25
RxD[3]	PA12	PA26
RxD[4]	PA13	PA27
RxD[5]	PB1	PB12
RxD[6]	PB0	PB13
RxD[7]	PB4	PB11
Rx_ERR	PA16	PA30
Rx_DV	PA15	PA29
Rx_CLK	PA0	PA31
Rx_CLK1 (for TBI 2 clocks mode)	PC29	PC28

3.4.8 Ports Tables

Table 3-16 through Table 3-22 describe the ports functionality according to the configuration of the port registers (CPPARxy and CPDIRxy).

Some input functions can come from two or three different pins for flexibility. Figure 3-12 shows an example in which three different pins can be selected for one input function. Secondary option programming is relevant only if primary option is programmed to the default value. Third option programming is relevant only if both primary and secondary options are programmed to the default value.

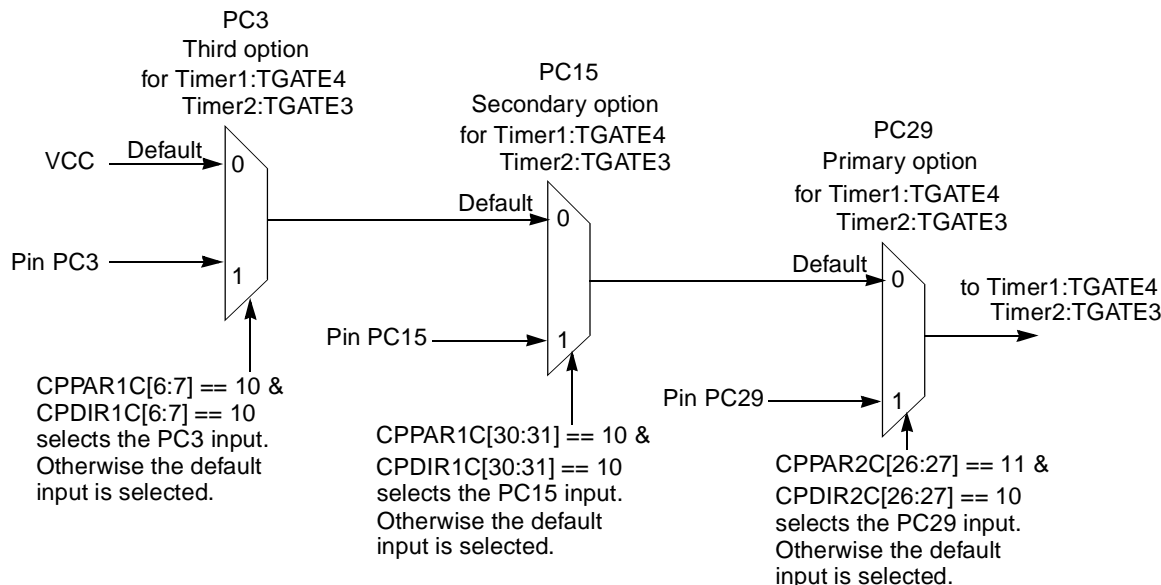


Figure 3-12. Primary, Secondary and Third Option Programming

In the tables below, for input functions that can come from two different pins, the default value for a primary option is simply a reference to the secondary option. In the secondary option, the programming is relevant only if the primary option is not used for the function. A similar approach is used to describe input functions that can come from three different pins.

Table 3-16. Port A Dedicated Pin Assignment

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PA0	IN	GPI_PA0	10	—	UCC1:Gigabit RX_CLK Enet	10	GND	—	—	—	—	—	—
	OUT	GPO_PA0	01	—	—	—	—	—	—	—	—	—	—
PA1	IN	GPI_PA1	10	—	SPI2:MDIO	11	GND	CE MUX:MDIO	11	GND	—	—	—
	OUT	GPO_PA1	01	—							—	—	—
PA2	IN	GPI_PA2	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PA2	01	—	CE MUX:MDC	01	—	SPI2:MDC	01	—	—	—	—

Table 3-16. Port A Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PA3	IN	GPI_PA3	10	—	—	—	—	—	—	—	—	—	
	OUT	GPO_PA3	01	—	UCC1:TxD[0]/T CG[0] Enet UCC1:TxD[0]/ UCC1:TxD (Serial) SER	01	—	TDMa:TxD[1]	01	—	—	—	
PA4	IN	GPI_PA4	10	—	—	—	—	—	—	—	—	—	
	OUT	GPO_PA4	01	—	UCC1:TxD[1]/T CG[1] Enet UCC1:TxD[1] SER	01	—	TDMa:TxD[2]	01	—	—	—	
PA5	IN	GPI_PA5	10	—	—	—	—	—	—	—	—	—	
	OUT	GPO_PA5	01	—	UCC1:TxD[2]/T CG[2] Enet UCC1:TxD[2] SER	01	—	TDMa:RQ	01	—	—	—	
PA6	IN	GPI_PA6	10	—	—	—	—	TDMa:TSYNC/G RANT (Secondary Option)	10	GND	—	—	
	OUT	GPO_PA6	01	—	UCC1:TxD[3]/T CG[3] Enet UCC1:TxD[3] SER	01	—	—	—	—	—	—	
PA7	IN	GPI_PA7	10	—	—	—	—	—	—	—	—	—	
	OUT	GPO_PA7	01	—	UCC1:TX_EN/T CG[8] Enet UCC1:RTS SER	01	—	—	—	—	—	—	

Table 3-16. Port A Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARx[SELn]=00			CPPARx[SELn]=01			CPPARx[SELn]=10			CPPARx[SELn]=11		
		Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input
PA8	IN	GPI_PA8	10	—	UPC1:RxClav[1] UTOPIA UPC1:PRPA[1]/ DRPA[1] POS (Secondary Option)	10	GND	TDMa:RxD[0]/ TDMa:RxD (Serial) (Secondary Option)	11	GND	UPC2:TxAddr[1] UTOPIA slave UPC2:TADR[1] POS slave (Secondary Option)	10	GND
	OUT	GPO_PA8	01	—	UCC1:TX_ER/T CG[9] Enet	01	—				UPC2:RxAddr[1] UTOPIA master UPC2:RADR[1] POS master	01	—
PA9	IN	GPI_PA9	10	—	UCC1:RxD[0]/R CG[0] Enet UCC1:RxD[0]/ UCC1:RxD (Serial) SER	10	GND	TDMa:RxD[1]	10	GND	—	—	—
	OUT	GPO_PA9	01	—	—	—	—	—	—	—	—	—	—
PA10	IN	GPI_PA10	10	—	UCC1:RxD[1]/R CG[1] Enet UCC1:RxD[1] SER	10	GND	TDMa:RxD[2]	10	GND			
	OUT	GPO_PA10	01	—	—	—	—	—	—	—	—	—	—
PA11	IN	GPI_PA11	10	—	UCC1:RxD[2]/R CG[2] Enet UCC1:RxD[2] SER	10	GND	—	—	—	—	—	—
	OUT	GPO_PA11	01	—	—	—	—	Si:STRB1	01				
PA12	IN	GPI_PA12	10	—	UCC1:RxD[3]/R CG[3] Enet UCC1:RxD[3] SER	10	GND	—	—	—	—	—	—
	OUT	GPO_PA12	01	—	—	—	—	TDMa:CLKO	01	—			

Table 3-16. Port A Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PA13	IN	GPI_PA13	10	—	UCC1:RxD[4]/R CG[4]COL Enet UCC1:RxD[4] SER	10	GND	TDMa:TxD[0]/ TDMa:TxD (Serial) (Secondary Option)	11	GND	UPC1:TxClav[1] UTOPIA UPC1:PTPA[1]/ DTPA[1] POS (Secondary Option)	10	GND
	OUT	GPO_PA13	01	—	—	—	—	—	—	—	UPC2:TxData[4] UTOPIA 16 ¹ UPC2:TDAT[4] POS	01	—
PA14	IN	GPI_PA14	10	—	UCC1:CRS/SD ET Enet	10	GND	TDMa:RSYNC (Secondary Option)	10	GND	UPC2:TxAddr[3] UTOPIA slave UPC2:TADR[3] POS slave (Secondary Option)	10	GND
	OUT	GPO_PA14	01	—	UPC1:RxEnb[1] UTOPIA UPC1:RENB[1] POS	01	—	—	—	—	UPC2:RxAddr[3] UTOPIA master UPC2:RADR[3] POS master	01	—
PA15	IN	GPI_PA15	10	—	UCC1:RX_DV/R CG[8] Enet UCC1:CTS SER	10	GND	TDMa:RxD[3]	10	GND	—	—	—
	OUT	GPO_PA15	01	—	—	—	—	—	—	—	—	—	—
PA16	IN	GPI_PA16	10	—	UCC1:RX_ER/R CG[9] Enet UCC1:CD SER	10	GND	TDMa:TSYNC/G RANT (Primary Option)	10	PA6	UPC2:RxAddr[3] UTOPIA slave UPC2:RADR[3] POS slave (Secondary Option)	10	GND
	OUT	GPO_PA16	01	—	UPC1:TxEnb[1] UTOPIA UPC1:TENB[1] POS	01	—	TDMa:TxD[3]	01	—	UPC2:TxAddr[3] UTOPIA master UPC2:TADR[3] POS master	01	—

Table 3-16. Port A Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PA17	IN	GPI_PA17	10	—	—	—	—	—	—	—	TDMe:TSYNC/GRANT (Third Option)	10	GND
	OUT	GPO_PA17	01	—	UCC2:TxD[0]/TCG[0] Enet UCC2:TxD[0]/UCC2:TxD (Serial) SER	01	—	—	—	—	TDMb:TxD[1]	01	—
PA18	IN	GPI_PA18	10	—	—	—	—	—	—	—	TDMd:TSYNC/GRANT (Third Option)	10	GND
	OUT	GPO_PA18	01	—	UCC2:TxD[1]/TCG[1] Enet UCC2:TxD[1] SER	01	—	TDMb:TxD[2]	01	—	—	—	—
PA19	IN	GPI_PA19	10	—	—	—	—	UPC2:RERR POS master UPC2:TERR POS slave (Secondary Option)	10	GND	TDMd:TxD[0]/TDMd:TxD (Serial) (Third Option)	11	GND
	OUT	GPO_PA19	01	—	UCC2:TxD[2]/TCG[2] Enet UCC2:TxD[2] SER	01	—	TDMb:RQ	01	—	—	—	—
PA20	IN	GPI_PA20	10	—	UPC2:STPA POS master (Secondary Option)	10	GND	TDMb:TSYNC/GRANT (Secondary Option)	10	GND	—	—	—
	OUT	GPO_PA20	01	—	UCC2:TxD[3]/TCG[3] Enet UCC2:TxD[3] SER	01	—	UPC2:STPA POS slave	01	—	—	—	—

Table 3-16. Port A Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PA2 1	IN	GPI_PA21	10	—	—	—	TDMf:TSYNC/G RANT (Third Option)	10	GND	—	—	—	
	OUT	GPO_PA21	01	—	UCC2:TX_ENT/ CG[8] Enet UCC2:RTS SER	01	—	—	—	—	—	—	
PA2 2	IN	GPI_PA22	10	—	UPC1:RxClav[2] UTOPIA UPC1:PRPA[2]/ DRPA[2] POS (Secondary Option)	10	GND	TDMb:TxD[0] TDMb:TxD (Secondary Option)	11	GND	UPC2:TxAddr[2] UTOPIA slave UPC2:TADR[2] POS slave (Secondary Option)	10	GND
	OUT	GPO_PA22	01	—	UCC2:TX_ER/T CG[9] Enet	01	—				UPC2:RxAddr[2] UTOPIA master UPC2:RADR[2] POS master	01	—
PA2 3	IN	GPI_PA23	10	—	UCC2:RxD[0]/R CG[0] Enet UCC2:RxD[0]/ UCC2:RxD (Serial) SER (Primary Option)	10	PF23	TDMb:RxD[1]	10	GND	TDMf:RSYNC (Third Option)	10	GND
	OUT	GPO_PA23	01	—	—	—	—	—	—	—	—	—	—
PA2 4	IN	GPI_PA24	10	—	UCC2:RxD[1]/R CG[1] Enet UCC2:RxD[1] SER (Primary Option)	10	PF24	TDMb:RxD[2]	10	GND	TDMd:RSYNC (Third Option)	10	GND
	OUT	GPO_PA24	01	—	—	—	—	—	—	—	—	—	—

Table 3-16. Port A Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PA25	IN	GPI_PA25	10	—	UCC2:RxD[2]/R CG[2] Enet UCC2:RxD[2] SER (Primary Option)	10	PG30	TDMd:RxD[0]/ TDMd:RxD (Serial) (Third Option)	11	GND	—	—	—
	OUT	GPO_PA25	01	—	SI:STRB2	01	—						
PA26	IN	GPI_PA26	10	—	UCC2:RxD[3]/R CG[3] Enet UCC2:RxD[3] SER (Primary Option)	10	PF25	TDMe:RSYNC (Third Option)	10	GND	—	—	—
	OUT	GPO_PA26	01	—	UPC2:TMOD POS master UPC2:RMOD POS slave	01	—	TDMb:CLKO	01	—			
PA27	IN	GPI_PA27	10	—	UCC2:RxD[4]/R CG[4]COL Enet UCC2:RxD[4] SER (Primary Option)	10	PG3	TDMb:RxD[0]/ TDMb:RxD (Serial) (Secondary Option)	11	GND	UPC1:TxClav[2] UTOPIA UPC1:PTPA[2]/ DTPA[2] POS (Secondary Option)	10	GND
	OUT	GPO_PA27	01	—	UPC2:TxSOC UTOPIA master UPC2:RxSOC UTOPIA slave UPC2:TSOP POS master UPC2:RSOP POS slave	01	—				—	—	
PA28	IN	GPI_PA28	10	—	UCC2:CRS/SD ET Enet (Primary Option)	10	PF28	TDMb:RSYNC (Secondary Option)	10	GND	—	—	—
	OUT	GPO_PA28	01	—	DMA_DDONE0 (Secondary Option)	01	—	UPC2:RxAddr[5] UTOPIA UPC2:RADR[5] POS	01	—	Timer1:TOUT1	01	—

Table 3-16. Port A Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PA29	IN	GPI_PA29	10	—	UCC2:RX_DV/R CG[8] Enet UCC2:CTS SER (Primary Option)	10	PG31	TDMb:RxD[3]	10	GND	TDMe:RxD[0]/ TDMe:RxD (Serial) (Third Option)	11	GND
	OUT	GPO_PA29	01	—	—	—	—	—	—	—	—	—	—
PA30	IN	GPI_PA30	10	—	UCC2:RX_ER/R CG[9] Enet UCC2:CD SER (Primary Option)	10	PF20	TDMb:TSYNC/G RANT (Primary Option)	10	PA20	TDMe:TxD[0]/ TDMe:TxD (Serial) (Third Option)	11	GND
	OUT	GPO_PA30	01	—	DMA_DDONE3 (Secondary Option)	01	—	TDMb:TxD[3]	01	—	—	—	—
PA31	IN	GPI_PA31	10	—	UCC2:Gigabit RX_CLK Enet (Secondary Option)	10	GND	UPC2:RVAL POS master (Secondary Option)	10	GND	Timer1:TGATE2 Timer2:TGATE1 (Secondary Option)	10	VCC
	OUT	GPO_PA31	01	—	—	—	—	UPC2:RVAL POS slave	01	—	—	—	—

¹ This pin name applies to UTOPIA master mode only. For UTOPIA slave mode, replace TxData with RxData and RxData with TxData.

Table 3-17. Port B Dedicated Pin Assignment

Pin	Pin Functions												
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input
PB0	IN	GPI_PB0	10	—	TDMd:RSYNC (Secondary Option)	10	PA24	UCC1:RxD[6]/R CG[6] Enet UCC1:RxD[6] SER (Secondary Option)	10	GND	UPC2:TxAddr[0] UTOPIA slave UPC2:TADR[0] POS slave (Secondary Option)	10	GND
	OUT	GPO_PB0	01	—	UCC3:TxD[0] Enet UCC3:TxD[0]/ UCC3:TxD (Serial) SER	01	—	TDMc:TxD[1]	01	—	UPC2:RxAddr[0] UTOPIA master UPC2:RADR[0] POS master	01	—
PB1	IN	GPI_PB1	10	—	UPC2:RxData[1] UTOPIA 8 ¹ UPC2:RxData[9] UTOPIA 16 ¹ UPC2:RDAT[9] POS (Secondary Option)	10	GND	UCC1:RxD[5]/R CG[5] Enet UCC1:RxD[5] SER (Secondary Option)	10	GND	TDMd:TxD[0]/ TDMd:TxD (Serial) (Secondary Option)	11	PA19
	OUT	GPO_PB1	01	—	UCC3:TxD[1] Enet UCC3:TxD[1] SER	01	—	TDMc:TxD[2]	01	—			
PB2	IN	GPI_PB2	10	—	UCC4:RxD[1] Enet UCC4:RxD[1] SER (Secondary Option)	10	GND	UPC2:RxData[7] UTOPIA 8 ¹ UPC2:RxData[1 5] UTOPIA 16 ¹ UPC2:RDAT[15] POS (Secondary Option)	10	GND	TDMc:RSYNC (Secondary Option)	10	GND
	OUT	GPO_PB2	01	—	UCC2:TxD[4]/TC G[4] Enet UCC2:TxD[4] SER	01	—	UCC3:TxD[2] Enet UCC3:TxD[2] SER	01	—	USB:OE	01	—

Table 3-17. Port B Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB DIRn	Default Input	Function	CPDIRxB DIRn	Default Input	Function	CPDIRxB DIRn	Default Input	Function	CPDIRxB DIRn	Default Input
PB3	IN	GPI_PB3	10	—	UPC2:RxData[6] UTOPIA 8 ¹ UPC2:RxData[14] UTOPIA 16 ¹ UPC2:RDAT[14] ¹ POS (Secondary Option)	10	GND	TDMc:TSYNC/G RANT (Secondary Option)	10	GND	UCC4:RX_DV Enet UCC4:CTS SER (Secondary Option)	10	GND
	OUT	GPO_PB3	01	—	UCC3:TxD[3] Enet UCC3:TxD[3] SER	01	—	UCC2:TxD[5]/T CG[5] Enet UCC2:TxD[5] SER	01	—	USB:TP	01	—
PB4	IN	GPI_PB4	10	—	Timer1:TIN3 Timer2:TIN4 1588_EXT_TRIG 1 (Secondary Option)	10	GND	UCC1:RxD[7]/R CG[7] Enet UCC1:RxD[7] SER (Secondary Option)	10	GND	UPC2:RxAddr[0] UTOPIA slave UPC2:RADR[0] POS slave (Secondary Option)	10	GND
	OUT	GPO_PB4	01	—	UCC3:TX_EN Enet UCC3:RTS SER	01	—	DMA_DACK2 (Secondary Option)	01	—	UPC2:TxAddr[0] UTOPIA master UPC2:TADR[0] POS master	01	—
PB5	IN	GPI_PB5	10	—	DMA_DREQ2 (Secondary Option)	10	VCC	TDMc:RxD[0]/ TDMc:RxD (Serial) (Secondary Option)	11	GND	UCC4:RX_ER Enet UCC4:CD SER (Secondary Option)	10	GND
	OUT	GPO_PB5	01	—	UCC3:TX_ER Enet	01	—	UCC2:TxD[6]/TC G[6] Enet UCC2:TxD[6] SER			01	—	

Table 3-17. Port B Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB DIRn	Default Input	Function	CPDIRxB DIRn	Default Input	Function	CPDIRxB DIRn	Default Input	Function	CPDIRxB DIRn	Default Input
PB6	IN	GPI_PB6	10	—	UCC3:RxD[0] Enet UCC3:RxD[0]/ UCC3:RxD (Serial) SER	10	GND	TDMc:RxD[1]	10	GND	TDMd:TSYNC/G RANT (Secondary Option)	10	PA18
	OUT	GPO_PB6	01	—	—	—	DMA_DDONE2 (Secondary Option)	01	—	UCC1:TxD[4]/TC G[4] Enet UCC1:TxD[4] SER	01	—	
PB7	IN	GPI_PB7	10	—	UCC3:RxD[1] Enet UCC3:RxD[1] SER	10	GND	TDMc:RxD[2]	10	GND	TDMd:RxD[0]/ TDMd:RxD (Serial) (Secondary Option)	11	PA25
	OUT	GPO_PB7	01	—	UCC1:TxD[5]/TC G[5] Enet UCC1:TxD[5] SER	01	—	UPC2:TxData[6] UTOPIA 8 ¹ UPC2:TxData[1 4] UTOPIA 16 ¹ UPC2:TDAT[14] POS	01	—			
PB8	IN	GPI_PB8	10	—	UCC3:RxD[2] Enet UCC3:RxD[2] SER	10	GND	TDMc:TxD[0]/ TDMc:TxD (Serial) (Secondary Option)	11	GND	UCC4:RxD[0] Enet UCC4:RxD[0]/ UCC4:RxD (Serial) SER (Secondary Option)	10	GND
	OUT	GPO_PB8	01	—	USB:TN	01	—	UCC2:TxD[7]/TC G[7] Enet UCC2:TxD[7] SER			01		

Table 3-17. Port B Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB DIRnI	Default Input	Function	CPDIRxB DIRnI	Default Input	Function	CPDIRxB DIRnI	Default Input	Function	CPDIRxB DIRnI	Default Input
PB9	IN	GPI_PB9	10	—	UCC3:RxD[3] Enet UCC3:RxD[3] SER	10	GND	UPC2:RxData[4] UTOPIA 8 ¹ UPC2:RxData[1 2] UTOPIA 16 ¹ UPC2:RDAT[12 ¹ POS (Secondary Option)	10	GND	USB:RP (Secondary Option)	10	GND
	OUT	GPO_PB9	01	—	TDMc:CLKO	01	—	UCC1:TxD[6]/T CG[6] Enet UCC1:TxD[6] SER	01	—	UCC4:TxD[0] Enet UCC4:TxD[0]/ UCC4:TxD (Serial) SER	01	—
PB10	IN	GPI_PB10	10	—	UCC3:COL Enet	10	GND	UPC2:RxData[3] UTOPIA 8 ¹ UPC2:RxData[1 1] UTOPIA 16 ¹ UPC2:RDAT[11 ¹ POS (Secondary Option)	10	GND	USB:RXD (Secondary Option)	10	GND
	OUT	GPO_PB10	01	—	SI:STRB3	01	—	UCC1:TxD[7]/T CG[7] Enet UCC1:TxD[7] SER	01	—	UCC4:TxD[1] Enet UCC4:TxD[1] SER	01	—
PB11	IN	GPI_PB11	10	—	UCC3:CRS Enet	10	GND	UCC2:RxD[7]/R CG[7] Enet UCC2:RxD[7] SER (Primary Option)	10	PF27	USB:RN (Secondary Option)	10	GND
	OUT	GPO_PB11	01	—	UPC2:TxD[2] UTOPIA 8 ¹ UPC2:TxD[10] UTOPIA 16 ¹ UPC2:TDAT[10] POS	01	—	TDMc: \overline{RQ}	01	—	UCC4:TX_EN Enet UCC4:RTS SER	01	—

Table 3-17. Port B Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB DIRn	Default Input	Function	CPDIRxB DIRn	Default Input	Function	CPDIRxB DIRn	Default Input	Function	CPDIRxB DIRn	Default Input
PB12	IN	GPI_PB12	10	—	UCC3:RX_DV Enet UCC3:CTS SER	10	GND	UCC2:RxD[5]/R CG[5] Enet UCC2:RxD[5] SER (Primary Option)	10	PG13	UPC2:RxPrty UTOPIA master UPC2:TxPrty UTOPIA slave UPC2:RPRTY POS master UPC2:TPRTY POS slave (Secondary Option)	10	GND
	OUT	GPO_PB12	01	—	DMA_DACK3 (Secondary Option)	01	—	TDMc:TxD[3]	01	—	—	—	—
PB13	IN	GPI_PB13	10	—	UCC3:RX_ER Enet UCC3:CD SER	10	GND	TDMc:RxD[3]	10	GND	UCC2:RxD[6]/RC G[6] Enet UCC2:RxD[6] SER (Primary Option)	10	PF29
	OUT	GPO_PB13	01	—	—	—	—	DMA_DDONE0 (Primary Option)	01	—	UPC2:TxPrty UTOPIA master UPC2:RxPrty UTOPIA slave UPC2:TPRTY POS master UPC2:RPRTY POS slave	01	—
PB14	IN	GPI_PB14	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PB14	01	—	UCC4:TxD[0] Enet UCC4:TxD[0]/ UCC4:TxD (Serial) SER	01	—	TDMd:TxD[1]	01	—	UPC1:TxSOC UTOPIA master UPC1:RxSOC UTOPIA slave UPC1:TSOP POS master UPC1:RSOP POS slave	01	—

Table 3-17. Port B Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB DIRn	Default Input	Function	CPDIRxB DIRn	Default Input	Function	CPDIRxB DIRn	Default Input	Function	CPDIRxB DIRn	Default Input
PB15	IN	GPI_PB15	10	—	—	—	—	—	—	—	UPC1:RxEnb UTOPIA slave UPC1:RENB POS slave	10	VCC
	OUT	GPO_PB15	01	—	UCC4:TxD[1] Enet UCC4:TxD[1] SER	01	—	TDMd:TxD[2]	01	—	UPC1:TxEnb[0] UTOPIA master UPC1:TENB[0] POS master	01	—
PB16	IN	GPI_PB16	10	—	UPC1:TxClav[0] UTOPIA master UPC1:PTPA[0]/DT PA[0] POS master	10	GND	TDMd:RSYNC (Primary Option)	10	PB0	—	—	—
	OUT	GPO_PB16	01	—	UCC4:TxD[2] Enet UCC4:TxD[2] SER	01	—	UPC1:RxClav UTOPIA slave UPC1:PRPA/DR PA POS slave	01	—	—	—	—
PB17	IN	GPI_PB17	10	—	—	—	—	TDMd:TSYNC/G RANT (Primary Option)	10	PB6	—	—	—
	OUT	GPO_PB17	01	—	UCC4:TxD[3] Enet UCC4:TxD[3] SER	01	—	UPC1:TxData[7] UTOPIA 8 ¹ UPC1:TxData[1 5] UTOPIA 16 ¹ UPC1:TDAT[15] POS	01	—	—	—	—
PB18	IN	GPI_PB18	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PB18	01	—	UCC4:TX_EN Enet UCC4:RTS SER	01	—	UPC1:TxData[6] UTOPIA 8 ¹ UPC1:TxData[1 4] UTOPIA 16 ¹ UPC1:TDAT[14] POS	01	—	—	—	—

Table 3-17. Port B Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB DIRnI	Default Input	Function	CPDIRxB DIRnI	Default Input	Function	CPDIRxB DIRnI	Default Input	Function	CPDIRxB DIRnI	Default Input
PB19	IN	GPI_PB19	10	—	TDMd:TxD[0]/ TDMd:TxD (Serial) (Primary Option)	11	PB1	—	—	—	—	—	
	OUT	GPO_PB19	01	—				UCC4:TX_ER Enet	01	—	UPC1:TxData[5] UTOPIA 8 ¹ UPC1:TxData[13] UTOPIA 16 ¹ UPC1:TDAT[13] POS	01	—
PB20	IN	GPI_PB20	10	—	UCC4:RxD[0] Enet UCC4:RxD[0]/ UCC4:RxD (Serial) SER (Primary Option)	10	PB8	TDMd:RxD[1]	10	GND	—	—	
	OUT	GPO_PB20	01	—	—	—	—	UPC1:TxData[4] UTOPIA 8 ¹ UPC1:TxData[1 2]] UTOPIA 16 ¹ UPC1:TDAT[12] POS	01	—	—	—	
PB21	IN	GPI_PB21	10	—	UCC4:RxD[1] Enet UCC4:RxD[1] SER (Primary Option)	10	PB2	TDMd:RxD[2]	10	GND	—	—	
	OUT	GPO_PB21	01	—	—	—	—	UPC1:TxData[3] UTOPIA 8 ¹ UPC1:TxData[1 1]] UTOPIA 16 ¹ UPC1:TDAT[11] POS	01	—	—	—	

Table 3-17. Port B Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB DIRnI	Default Input	Function	CPDIRxB DIRnI	Default Input	Function	CPDIRxB DIRnI	Default Input	Function	CPDIRxB DIRnI	Default Input
PB22	IN	GPI_PB22	10	—	UCC4:RxD[2] Enet UCC4:RxD[2] SER	10	GND	TDMd:RxD[0]/ TDMd:RxD (Serial) (Primary Option)	11	PB7	—	—	
	OUT	GPO_PB22	01	—	UPC1:TxData[2] UTOPIA 8 ¹ UPC1:TxData[10] UTOPIA 16 ¹ UPC1:TDAT[10] POS	01	—				—		
PB23	IN	GPI_PB23	10	—	UCC4:RxD[3] Enet UCC4:RxD[3] SER	10	GND	—	—	—	—	—	
	OUT	GPO_PB23	01	—	UPC1:TxData[1] UTOPIA 8 ¹ UPC1:TxData[9] UTOPIA 16 ¹ UPC1:TDAT[9] POS	01	—	TDMd:CLKO	01	—	—	—	
PB24	IN	GPI_PB24	10	—	UCC4:COL Enet	10	GND	CE:EXT_REQ1	10	GND	—	—	
	OUT	GPO_PB24	01	—	UPC1:TxData[0] UTOPIA 8 ¹ UPC1:TxData[8] UTOPIA 16 ¹ UPC1:TDAT[8] POS	01	—	SI:STRB4	01	—	—	—	
PB25	IN	GPI_PB25	10	—	UCC4:CRS Enet	10	GND	—	—	—	TDMa:RxD[0]/ TDMa:RxD (Serial) (Primary Option)	11	PA8
	OUT	GPO_PB25	01	—	UPC1:TxData[7] UTOPIA 16 ¹ UPC1:TDAT[7] POS	01	—	TDMd:RQ	01	—			

Table 3-17. Port B Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input
PB26	IN	GPI_PB26	10	—	UCC4:RX_DV Enet UCC4:CTS SER (Primary Option)	10	PB3	TDMa:TxD[0]/ TDMa:TxD (Serial) (Primary Option)	11	PA13	—	—	
	OUT	GPO_PB26	01	—	UPC1:TxData[6] UTOPIA 16 ¹ UPC1:TDAT[6] POS	01	—	—	—	TDMd:TxD[3]	01	—	
PB27	IN	GPI_PB27	10	—	UCC4:RX_ER Enet UCC4:CD SER (Primary Option)	10	PB5	TDMd:RxD[3]	10	GND	TDMa:RSYNC (Primary Option)	10	PA14
	OUT	GPO_PB27	01	—	—	—	—	UPC1:TxData[5] UTOPIA 16 ¹ UPC1:TDAT[5] POS	01	—	—	—	—

¹ This pin name applies to UTOPIA master mode only. For UTOPIA slave mode, replace TxData with RxData and RxData with TxData.

Table 3-18. Port C Dedicated Pin Assignment

Pin	Pin Functions												
	Direction	CPPARCx[SELn]=00			CPPARCx[SELn]=01			CPPARCx[SELn]=10			CPPARCx[SELn]=11		
		Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input
PC0	IN	GPI_PC0	10	—	CLK1	10	GND	Timer1:TIN1 Timer2:TIN2 (Secondary Option)	10	GND	—	—	
	OUT	GPO_PC0	01	—	DMA_DACK3 (Primary Option)	01	—	PTP_ALARM1 (Primary Option)	01	—	BRGO1	01	—

Table 3-18. Port C Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARCx[SELn]=00			CPPARCx[SELn]=01			CPPARCx[SELn]=10			CPPARCx[SELn]=11		
		Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input
PC1	IN	GPI_PC1	10	—	CLK2	10	GND	Timer1: $\overline{\text{TGATE1}}$ Timer2: $\overline{\text{TGATE2}}$ (Secondary Option)	10	VCC	—	—	—
	OUT	GPO_PC1	01	—	—	—	—	$\overline{\text{DMA_DACK0}}$ (Secondary Option)	01	—	BRGO2	01	—
PC2	IN	GPI_PC2	10	—	CLK3	10	GND	—	—	—	—	—	—
	OUT	GPO_PC2	01	—	—	—	—	UCC2:GTx Clock Enet	01	—	UCC2:CLKO	01	—
PC3	IN	GPI_PC3	10	—	CLK4	10	GND	Timer1: $\overline{\text{TGATE4}}$ Timer2: $\overline{\text{TGATE3}}$ (Third Option)	10	VCC	—	—	—
	OUT	GPO_PC3	01	—	SI:STRB1	01	—	UCC4:CLKO	01	—	BRGO3	01	—
PC4	IN	GPI_PC4	10	—	CLK5	10	GND	—	—	—	—	—	—
	OUT	GPO_PC4	01	—	SI:STRB8	01	—	UCC8:CLKO	01	—	BRGO4	01	—
PC5	IN	GPI_PC5	10	—	CLK6	10	GND	—	—	—	—	—	—
	OUT	GPO_PC5	01	—	—	—	—	BRGO5	01	—	PTP_PPS3 (Primary Option)	01	—
PC6	IN	GPI_PC6	10	—	CLK7	10	GND	—	—	—	—	—	—
	OUT	GPO_PC6	01	—	—	—	—	—	—	—	UPC2:CLKO	01	—
PC7	IN	GPI_PC7	10	—	CLK8	10	GND	—	—	—	—	—	—
	OUT	GPO_PC7	01	—	—	—	—	UCC6:CLKO	01	—	BRGO6	01	—
PC8	IN	GPI_PC8	10	—	CLK9	10	GND	—	—	—	—	—	—
	OUT	GPO_PC8	01	—	—	—	—	—	—	—	BRGO7	01	—
PC9	IN	GPI_PC9	10	—	CLK10	10	GND	—	—	—	—	—	—
	OUT	GPO_PC9	01	—	—	—	—	UCC1:CLKO	01	—	UCC1:GTx Clock Enet	01	—

Table 3-18. Port C Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARCx[SELn]=00			CPPARCx[SELn]=01			CPPARCx[SELn]=10			CPPARCx[SELn]=11		
		Function	CPDIRx[C DIRn]	Default Input	Function	CPDIRx[C DIRn]	Default Input	Function	CPDIRx[C DIRn]	Default Input	Function	CPDIRx[C DIRn]	Default Input
PC10	IN	GPI_PC10	10	—	CLK11	10	GND	UPC2:RMOD POS master UPC2:TMOD POS slave (Secondary Option)	10	GND	UCC1:COL Enet (Secondary Option)	10	PA13
	OUT	GPO_PC10	01	—	Timer1:TOUT4	01	—	SI:STRB2	01	—	UCC3:CLKO	01	—
PC11	IN	GPI_PC11	10	—	CLK12	10	GND	Timer1:TIN4 Timer2:TIN3 1588_EXT_TRIG 2 (Secondary Option)	10	GND	UCC2:COL Enet (Secondary Option)	10	PA27
	OUT	GPO_PC11	01	—	DMA_DDONE2 (Primary Option)	01	—	UPC2:TERR POS master UPC2:RERR POS slave	01	—	BRGO8	01	—
PC12	IN	GPI_PC12	10	—	CLK13	10	GND	CE:EXT_REQ2	10	GND	—	—	—
	OUT	GPO_PC12	01	—	—	—	—	PTP_REF_CLK (Primary Option)	01	—	UPC1:CLKO	01	—
PC13	IN	GPI_PC13	10	—	CLK14	10	GND	DMA_DREQ0 (Secondary Option)	10	VCC	—	—	—
	OUT	GPO_PC13	01	—	UPC1:TxEnb[3] UTOPIA UPC1:TENB[3] POS	01	—	UCC5:CLKO	01	—	—	—	—
PC14	IN	GPI_PC14	10	—	CLK15	10	GND	Timer1:TIN3 Timer2:TIN4 1588_EXT_TRIG 1 (Primary Option)	10	PB4	UPC2:REOP POS master UPC2:TEOP POS slave (Secondary Option)	10	GND
	OUT	GPO_PC14	01	—	—	—	—	UCC7:CLKO	01	—	BRGO9	01	—

Table 3-18. Port C Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARCx[SELn]=00			CPPARCx[SELn]=01			CPPARCx[SELn]=10			CPPARCx[SELn]=11		
		Function	CPDIRx[C DIRn]	Default Input	Function	CPDIRx[C DIRn]	Default Input	Function	CPDIRx[C DIRn]	Default Input	Function	CPDIRx[C DIRn]	Default Input
PC15	IN	GPI_PC15	10	—	CLK16	10	GND	Timer1: $\overline{\text{TGATE4}}$ Timer2: $\overline{\text{TGATE3}}$ (Secondary Option)	10	PC3	Timer1: $\overline{\text{TGATE2}}$ Timer2: $\overline{\text{TGATE1}}$ (Primary Option)	10	PA31
	OUT	GPO_PC15	01	—	—	—	—	—	—	—	BRGO10	01	—
PC16	IN	GPI_PC16	10	—	CLK17	10	GND	UPC1:TxClav[3] UTOPIA UPC1:PTPA[3]/D TPA[3] POS (Secondary Option)	10	GND	—	—	—
	OUT	GPO_PC16	01	—	$\overline{\text{DMA_DACK2}}$ (Primary Option)	01	—	—	—	—	BRGO11	01	—
PC17	IN	GPI_PC17	10	—	CLK18	10	GND	$\overline{\text{DMA_DREQ3}}$ (Secondary Option)	10	VCC	UPC1:RxClav[3] UTOPIA UPC1:PRPA[3]/ DRPA[3] POS (Secondary Option)	10	GND
	OUT	GPO_PC17	01	—	—	—	—	BRGO12	01	—	PTP_PPS2 (Primary Option)	01	—
PC18	IN	GPI_PC18	10	—	CLK19	10	GND	—	—	—	—	—	—
	OUT	GPO_PC18	01	—	$\overline{\text{DMA_DDONE3}}$ (Primary Option)	01	—	BRGO13	01	—	Timer2: $\overline{\text{TOUT3}}$	01	—
PC19	IN	GPI_PC19	10	—	CLK20	10	GND	—	—	—	—	—	—
	OUT	GPO_PC19	01	—	—	—	—	UPC1:RxEnb[3] UTOPIA UPC1:RENB[3] POS	01	—	BRGO14	01	—

Table 3-18. Port C Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARCx[SELn]=00			CPPARCx[SELn]=01			CPPARCx[SELn]=10			CPPARCx[SELn]=11		
		Function	CPDIRx[C DIRn]	Default Input	Function	CPDIRx[C DIRn]	Default Input	Function	CPDIRx[C DIRn]	Default Input	Function	CPDIRx[C DIRn]	Default Input
PC20	IN	GPI_PC20	10	—	CLK21	10	GND	—	—	—	—	—	
	OUT	GPO_PC20	01	—	UPC2:TEOP POS master UPC2:REOP POS slave	01	—	UPC1:TxEnb[2] UTOPIA UPC1:TENB[2] POS	01	—	BRGO15	01	—
PC21	IN	GPI_PC21	10	—	CLK22	10	GND	—	—	—	—	—	
	OUT	GPO_PC21	01	—	—	—	—	PTP_PPS1 (Primary Option)	01	—	UPC1:RxEnb[2] UTOPIA UPC1:RENB[2] POS	01	—
PC22	IN	GPI_PC22	10	—	CLK23	10	GND	—	—	—	DMA_DREQ3 (Primary Option)	10	PC17
	OUT	GPO_PC22	01	—	UPC1:TMOD POS master UPC1:RMOD POS slave	01	—	UCC1:TxD[7]/TC G[7] Enet UCC1:TxD[7] SER	01	—	UPC2:TxAddr[5] UTOPIA UPC2:TADR[5] POS	01	—
PC23	IN	GPI_PC23	10	—	CLK24	10	GND	UPC1:RMOD POS master UPC1:TMOD POS slave (Secondary Option)	10	GND	DMA_DREQ0 (Primary Option)	10	PC13
	OUT	GPO_PC23	01	—	PTP_ALARM2 (Primary Option)	01	—	UCC1:TxD[6]/TC G[6] Enet UCC1:TxD[6] SER	01	—	—	—	—

Table 3-18. Port C Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARCx[SELn]=00			CPPARCx[SELn]=01			CPPARCx[SELn]=10			CPPARCx[SELn]=11		
		Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input
PC24	IN	GPI_PC24	10	—	UPC1:REOP POS master UPC1:TEOP POS slave	10	GND	UPC2:TxAddr[4] UTOPIA slave UPC2:TADR[4] POS slave (Secondary Option)	10	GND	—	—	—
	OUT	GPO_PC24	01	—	UCC1:TxD[5]/TCG[5] Enet UCC1:TxD[5] SER	01	—	UPC2:RxAddr[4] UTOPIA master UPC2:RADR[4] POS master	01	—	Timer1:TOUT3	01	—
PC25	IN	GPI_PC25	10	—	Timer1:TGATE3 Timer2:TGATE4 (Secondary Option)	10	VCC	UPC2:RxData[5] UTOPIA 8 ¹ UPC2:RxData[13]] UTOPIA 16 ¹ UPC2:RDAT[13] ¹ POS (Secondary Option)	10	GND	—	—	—
	OUT	GPO_PC25	01	—	BRGO16	01	—	UCC1:TxD[4]/TCG[4] Enet UCC1:TxD[4] SER	01	—	UPC1:TEOP POS master UPC1:REOP POS slave	01	—
PC26	IN	GPI_PC26	10	—	RTC_CLK/PIT_CLK	10	GND	—	—	—	—	—	—
	OUT	GPO_PC26	01	—	—	—	—	SI:STRB3	01	—	—	—	—
PC27	IN	GPI_PC27	10	—	Timer1:TIN4 Timer2:TIN3 1588_EXT_TRIG2 (Primary Option)	10	PC11	TDMb:RxData[0]/ TDMb:RxData (Serial) (Primary Option)	11	PA27	TDMf:RSYNC (Secondary Option)	10	PA23
	OUT	GPO_PC27	01	—	SI:STRB4	01	—	—	—	—	—	—	—

Table 3-18. Port C Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARCx[SELn]=00			CPPARCx[SELn]=01			CPPARCx[SELn]=10			CPPARCx[SELn]=11		
		Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input
PC28	IN	GPI_PC28	10	—	UCC2:TBI RX_CLK1 Enet (Secondary Option)	10	GND	TDMb:TxD[0]/ TDMb:TxD (Serial) (Primary Option)	11	PA22	TDMf:TSYNC/GRA NT (Secondary Option)	10	PA21
	OUT	GPO_PC28	01	—	Timer1:TOUT4	01	—				SI:STRB5	01	—
PC29	IN	GPI_PC29	10	—	UCC1:TBI RX_CLK1 Enet	10	GND	TDMb:RSYNC (Primary Option)	10	PA28	Timer1:TGATE4 Timer2:TGATE3 (Primary Option)	10	PC15
	OUT	GPO_PC29	01	—	Timer2:TOUT3	01	—	SI:STRB6	01	—	DMA_DACK0 (Primary Option)	01	—
PC30	IN	GPI_PC30	10	—	PTP_REF_CLK	10	GND	—	—	—	—	—	—
	OUT	GPO_PC30	01	—	—	—	—	—	—	—	—	—	—

¹ This pin name applies to UTOPIA master mode only. For UTOPIA slave mode, replace TxData with RxData and RxData with TxData.

Table 3-19. Port D Dedicated Pin Assignment

Pin	Pin Functions												
	Direction	CPPARDx[SELn]=00			CPPARDx[SELn]=01			CPPARDx[SELn]=10			CPPARDx[SELn]=11		
		Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input
PD0	IN	GPI_PD0	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PD0	01	—	UCC5:TxD[0] Enet UCC5:TxD[0]/ UCC5:TxD (Serial) SER	01	—	TDMe:TxD[1]	01	—	UPC1:TxData[4] UTOPIA 16 ¹ UPC1:TDAT[4] POS	01	—

Table 3-19. Port D Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARDx[SELn]=00			CPPARDx[SELn]=01			CPPARDx[SELn]=10			CPPARDx[SELn]=11		
		Function	CPDIRx[D]DIRn]	Default Input	Function	CPDIRx[D]DIRn]	Default Input	Function	CPDIRx[D]DIRn]	Default Input	Function	CPDIRx[D]DIRn]	Default Input
PD1	IN	GPI_PD1	10	—	—	—	—	—	—	—	—	—	
	OUT	GPO_PD1	01	—	UCC5:TxD[1] Enet UCC5:TxD[1] SER	01	—	TDMe:TxD[2]	01	—	UPC1:TxData[3] UTOPIA 16 ¹ UPC1:TDAT[3] POS	01	—
PD2	IN	GPI_PD2	10	—	—	—	—	TDMe:RSYNC (Secondary Option)	10	PA26	—	—	
	OUT	GPO_PD2	01	—	UCC5:TxD[2] Enet UCC5:TxD[2] SER	01	—	UPC1:TxData[2] UTOPIA 16 ¹ UPC1:TDAT[2] POS	01	—	—	—	
PD3	IN	GPI_PD3	10	—	—	—	—	TDMe:TSYNC/GR ANT (Secondary Option)	10	PA17	TDMg:TxD[0]/ TDMg:TxD (Serial) (Secondary Option)	11	GND
	OUT	GPO_PD3	01	—	UCC5:TxD[3] Enet UCC5:TxD[3] SER	01	—	UPC1:TxData[1] UTOPIA 16 UPC1:TDAT[1] POS	01	—			
PD4	IN	GPI_PD4	10	—	—	—	—	—	—	—	—	—	
	OUT	GPO_PD4	01	—	UCC5:TX_EN Enet UCC5:RTS SER	01	—	—	—	—	UPC1:TxData[0] UTOPIA 16 ¹ UPC1:TDAT[0] POS	01	—
PD5	IN	GPI_PD5	10	—	UPC1:RxSOC UTOPIA master UPC1:TxSOC UTOPIA slave UPC1:RSOP POS master UPC1:TSOP POS slave	10	GND	TDMe:TxD[0]/ TDMe:TxD (Serial) (Secondary Option)	11	PA30	—	—	—
	OUT	GPO_PD5	01	—	UCC5:TX_ER Enet	01	—				—	—	—

Table 3-19. Port D Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARDx[SELn]=00			CPPARDx[SELn]=01			CPPARDx[SELn]=10			CPPARDx[SELn]=11		
		Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input
PD6	IN	GPI_PD6	10	—	UCC5:RxD[0] Enet UCC5:RxD[0]/ UCC5:RxD (Serial) SER (Secondary Option)	10	GND	TDMe:RxD[1]	10	GND	UPC1:TxEnb UTOPIA slave UPC1:TENB POS slave	10	VCC
	OUT	GPO_PD6	01	—	—	—	—	—	—	—	UPC1:RxEnb[0] UTOPIA master UPC1:RENb[0] POS master	01	—
PD7	IN	GPI_PD7	10	—	UCC5:RxD[1] Enet UCC5:RxD[1] SER (Secondary Option)	10	GND	TDMe:RxD[2]	10	GND	UPC1:RxClav[0] UTOPIA master UPC1:PRPA[0]/ DRPA[0] POS master	10	GND
	OUT	GPO_PD7	01	—	—	—	—	—	—	—	UPC1:TxClav UTOPIA slave UPC1:PTPA/DT PA POS slave	01	—
PD8	IN	GPI_PD8	10	—	UCC5:RxD[2] Enet UCC5:RxD[2] SER	10	GND	TDMe:RxD[0]/ TDMe:RxD (Serial) (Secondary Option)	11	PA29	UPC1:RxData[7] UTOPIA 8 ¹ UPC1:RxData[1 5] UTOPIA 16 ¹ UPC1:RDAT[15] POS	10	GND
	OUT	GPO_PD8	01	—	—	—	—	—	—	—	—	—	—
PD9	IN	GPI_PD9	10	—	UCC5:RxD[3] Enet UCC5:RxD[3] SER	10	GND	UPC1:RxData[6] UTOPIA 8 ¹ UPC1:RxData[14] UTOPIA 16 ¹ UPC1:RDAT[14] POS	10	GND	—	—	—
	OUT	GPO_PD9	01	—	—	—	—	TDMe:CLKO	01	—	—	—	—

Table 3-19. Port D Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARDx[SELn]=00			CPPARDx[SELn]=01			CPPARDx[SELn]=10			CPPARDx[SELn]=11		
		Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input
PD10	IN	GPI_PD10	10	—	UCC5:COL Enet	10	GND				UPC1:RxData[5] UTOPIA 8 ¹ UPC1:RxData[13] UTOPIA 16 ¹ UPC1:RDAT[13] POS	10	GND
	OUT	GPO_PD10	01	—	—	—	—	SI:STRB5	01	—	—	—	—
PD11	IN	GPI_PD11	10	—	UCC5:CRS Enet	10	GND	—	—	—	UPC1:RxData[4] UTOPIA 8 ¹ UPC1:RxData[12] UTOPIA 16 ¹ UPC1:RDAT[12] POS	10	GND
	OUT	GPO_PD11	01	—	—	—	—	TDMe:RQ	01	—	—	—	—
PD12	IN	GPI_PD12	10	—	UCC5:RX_DV Enet UCC5:CTS SER (Secondary Option)	10	GND	UPC1:RxData[3] UTOPIA 8 ¹ UPC1:RxData[11] UTOPIA 16 ¹ UPC1:RDAT[11] POS	10	GND	—	—	—
	OUT	GPO_PD12	01	—	—	—	—	TDMe:TxD[3]	01	—	—	—	—
PD13	IN	GPI_PD13	10	—	UCC5:RX_ER Enet UCC5:CD SER (Secondary Option)	10	GND	TDMe:RxD[3]	10	GND	UPC1:RxData[2] UTOPIA 8 ¹ UPC1:RxData[10] UTOPIA 16 ¹ UPC1:RDAT[10] POS	10	GND
	OUT	GPO_PD13	01	—	—	—	—	—	—	—	—	—	—

Table 3-19. Port D Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARDx[SELn]=00			CPPARDx[SELn]=01			CPPARDx[SELn]=10			CPPARDx[SELn]=11		
		Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input
PD14	IN	GPI_PD14	10	—	—	—	—	CE:EXT_REQ3	10	GND	UPC1:RxData[1] UTOPIA 8 ¹ UPC1:RxData[9] UTOPIA 16 ¹ UPC1:RDAT[9] POS	10	GND
	OUT	GPO_PD14	01	—	UCC6:TxD[0] Enet UCC6:TxD[0]/ UCC6:TxD (Serial) SER	01	—	TDMf:TxD[1]	01	—	—	—	—
PD15	IN	GPI_PD15	10	—	—	—	—	CE:EXT_REQ4	10	GND	UPC1:RxData[0] UTOPIA 8 ¹ UPC1:RxData[8] UTOPIA 16 ¹ UPC1:RDAT[8] POS	10	GND
	OUT	GPO_PD15	01	—	UCC6:TxD[1] Enet UCC6:TxD[1] SER	01	—	TDMf:TxD[2]	01	—	—	—	—
PD16	IN	GPI_PD16	10	—	TDMf:RSYNC (Primary Option)	10	PC27	Timer1:TIN2 Timer2:TIN1 (Secondary Option)	10	GND	UPC2:RxAddr[2] UTOPIA slave UPC2:RADR[2] POS slave (Secondary Option)	10	GND
	OUT	GPO_PD16	01	—	UCC6:TxD[2] Enet UCC6:TxD[2] SER	01	—	UPC1:TERR POS master UPC1:RERR POS slave	01	—	UPC2:TxAddr[2] UTOPIA master UPC2:TADR[2] POS master	01	—

Table 3-19. Port D Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARDx[SELn]=00			CPPARDx[SELn]=01			CPPARDx[SELn]=10			CPPARDx[SELn]=11		
		Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input
PD17	IN	GPI_PD17	10	—	UPC1:RERR POS master UPC1:TERR POS slave	10	GND	TDMf:TSYNC/GR ANT (Primary Option)	10	PC28	UPC2:RxAddr[1] UTOPIA slave UPC2:RADR[1] POS slave (Secondary Option)	10	GND
	OUT	Timer2:TO UT1	01	—	UCC6:TxD[3] Enet UCC6:TxD[3] SER	01	—	Timer1:TOU2	01	—	UPC2:TxAddr[1] UTOPIA master UPC2:TADR[1] POS master	01	—
PD18	IN	GPI_PD18	10	—	—	—	—	UPC1:RxData[7] UTOPIA 16 ¹ UPC1:RDAT[7] POS	10	GND	—	—	—
	OUT	GPO_PD1 8	01	—	UCC6:TX_EN Enet UCC6:RTS SER	01	—	—	—	—	—	—	—
PD19	IN	GPI_PD19	10	—	DMA_DREQ2 (Primary Option)	10	PB5	TDMf:RxData[0]/ TDMf:RxData (Serial) (Secondary Option)	11	GND	UPC1:RxPrty UTOPIA master UPC1:TxPrty UTOPIA slave UPC1:RPRTY POS master UPC1:TPRTY POS slave	10	GND
	OUT	GPO_PD1 9	01	—	UCC6:TX_ER Enet	01	—	—	—	—	PTP_PPS2 (Secondary Option)	01	—
PD20	IN	GPI_PD20	10	—	UCC6:RxData[0] Enet UCC6:RxData[0]/ UCC6:RxData (Serial) SER	10	GND	TDMf:RxData[1]	10	GND	UPC1:RxData[6] UTOPIA 16 ¹ UPC1:RDAT[6] POS	10	GND
	OUT	GPO_PD2 0	01	—	—	—	—	—	—	—	—	—	—

Table 3-19. Port D Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARDx[SELn]=00			CPPARDx[SELn]=01			CPPARDx[SELn]=10			CPPARDx[SELn]=11		
		Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input
PD21	IN	GPI_PD21	10	—	UCC6:RxD[1] Enet UCC6:RxD[1] SER	10	GND	TDMf:RxD[2]	10	GND	UPC1:RxData[5] UTOPIA 16 ¹ UPC1:RDAT[5] POS	10	GND
	OUT	GPO_PD2 1	01	—	—	—	—	—	—	—	—	—	—
PD22	IN	GPI_PD22	10	—	UCC6:RxD[2] Enet UCC6:RxD[2] SER	10	GND	TDMf:TxD[0]/ TDMf:TxD (Serial) (Secondary Option)	11	GND	—	—	—
	OUT	GPO_PD2 2	01	—	UPC1:TxPrty UTOPIA master UPC1:RxPrty UTOPIA slave UPC1:TPRTY POS master UPC1:RPRTY POS slave	01	—						
PD23	IN	GPI_PD23	10	—	UCC6:RxD[3] Enet UCC6:RxD[3] SER	10	GND	TDMg:TSYNC/GR ANT (Secondary Option)	10	GND	UPC1:RxData[4] UTOPIA 16 ¹ UPC1:RDAT[4] POS	10	GND
	OUT	GPO_PD2 3	01	—	—	—	—	TDMf:CLKO	01	—	—	—	—
PD24	IN	GPI_PD24	10	—	UCC6:COL Enet	10	GND	UPC1:RxData[3] UTOPIA 16 ¹ UPC1:RDAT[3] POS	10	GND	TDMg:RxD[0]/ TDMg:RxD (Serial) (Secondary Option)	11	GND
	OUT	GPO_PD2 4	01	—	—	—	—	SI:STRB6	01	—			
PD25	IN	GPI_PD25	10	—	UCC6:CRS Enet	10	GND	UPC1:RxData[2] UTOPIA 16 ¹ UPC1:RDAT[2] POS	10	GND	TDMg:RSYNC (Secondary Option)	10	GND
	OUT	GPO_PD2 5	01	—	TDMf:RQ	01	—	—	—	—	—	—	—

Table 3-19. Port D Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARDx[SELn]=00			CPPARDx[SELn]=01			CPPARDx[SELn]=10			CPPARDx[SELn]=11		
		Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input
PD26	IN	GPI_PD26	10	—	UCC6:RX_DV Enet UCC6:CTS SER	10	GND	UPC1:RxData[1] UTOPIA 16 ¹ UPC1:RDAT[1] POS	10	GND	—	—	—
	OUT	GPO_PD26	01	—	—	—	—	TDMf:TxD[3]	01	—	—	—	—
PD27	IN	GPI_PD27	10	—	UCC6:RX_ER Enet UCC6:CD SER	10	GND	TDMf:RxData[3]	10	GND	UPC1:RxData[0] UTOPIA 16 ¹ UPC1:RDAT[0] POS	10	GND
	OUT	GPO_PD27	01	—	—	—	—	—	—	—	—	—	—

¹ This pin name applies to UTOPIA master mode only. For UTOPIA slave mode, replace TxData with RxData and RxData with TxData.

Table 3-20. Port E Dedicated Pin Assignment

Pin	Pin Functions												
	Direction	CPPAREx[SELn]=00			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input
PE0	IN	GPI_PE0	10	—	—	—	—	TDMh:TxData[0]/ TDMh:TxData (Serial) (Secondary Option)	11	GND	—	—	—
	OUT	GPO_PE0	01	—	UCC7:TxData[0] Enet UCC7:TxData[0]/ UCC7:TxData (Serial) SER	01	—	—	—	—	UPC1:RxAddr[5] UTOPIA UPC1:RADDR[5] POS	01	—

Table 3-20. Port E Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPAREx[SELn]=0			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input
PE1	IN	GPI_PE1	10	—	—	—	TDMc:TSYNC/RANT (Primary Option)	10	PB3	UPC1:RxAddr[3] UTOPIA slave UPC1:RADR[3] POS slave	10	GND	
	OUT	GPO_PE1	01	—	UCC7:TxD[1] Enet UCC7:TxD[1] SER	01	—	UPC1:TxAddr[3] UTOPIA master UPC1:TADR[3] POS master	01	—	—	—	
PE2	IN	GPI_PE2	10	—	TDMg:RSYNC (Primary Option)	10	PD25	UPC1:TxAddr[0] UTOPIA slave UPC1:TADR[0] POS slave	10	GND	—	—	
	OUT	GPO_PE2	01	—	UCC7:TxD[2] Enet UCC7:TxD[2] SER	01	—	UPC1:RxAddr[0] UTOPIA master UPC1:RADR[0] POS master	01	—	—	—	
PE3	IN	GPI_PE3	10	—	TDMg:TSYNC/RANT (Primary Option)	10	PD23	UPC1:RxAddr[0] UTOPIA slave UPC1:RADR[0] POS slave	10	GND	—	—	
	OUT	GPO_PE3	01	—	UCC7:TxD[3] Enet UCC7:TxD[3] SER	01	—	UPC1:TxAddr[0] UTOPIA master UPC1:TADR[0] POS master	01	—	—	—	
PE4	IN	GPI_PE4	10	—	—	—	UPC1:TxAddr[2] UTOPIA slave UPC1:TADR[2] POS slave	10	GND	TDMh:TSYNC/RANT (Secondary Option)	10	GND	
	OUT	GPO_PE4	01	—	UCC7:TX_EN Enet UCC7:RTS SER	01	—	UPC1:RxAddr[2] UTOPIA master UPC1:RADR[2] POS master	01	—	—	—	

Table 3-20. Port E Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPAREx[SELn]=0			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input
PE5	IN	GPI_PE 5	10	—	—	—	TDMg:TxD[0]/ TDMg:TxD (Serial) (Primary Option)	11	PD3	UPC1:TxAddr[1] UTOPIA slave UPC1:TADR[1] POS slave	10	GND	
	OUT	GPO_P E5	01	—	UCC7:TX_ER Enet	01	—	—	—	UPC1:RxAddr[1] UTOPIA master UPC1:RADR[1] POS master	01	—	
PE6	IN	GPI_PE 6	10	—	UCC7:RxD[0] Enet UCC7:RxD[0]/ UCC7:RxD (Serial) SER	10	GND	TDMh:RSYNC (Secondary Option)	10	GND	—	—	
	OUT	GPO_P E6	01	—	PTP_ALARM2 (Secondary Option)	01	—	—	—	UPC1:TxAddr[5] UTOPIA UPC1:TADR[5] POS	01	—	
PE7	IN	GPI_PE 7	10	—	UCC7:RxD[1] Enet UCC7:RxD[1] SER	10	GND	UPC1:TxAddr[4] UTOPIA slave UPC1:TADR[4] POS slave	10	GND	TDMc:TxD[0]/ TDMc:TxD (Serial) (Primary Option)	11	PB8
	OUT	GPO_P E7	01	—	—	—	—	UPC1:RxAddr[4] UTOPIA master UPC1:RADR[4] POS master	01	—	—	—	
PE8	IN	GPI_PE 8	10	—	UCC7:RxD[2] Enet UCC7:RxD[2] SER	10	GND	TDMg:RxD[0]/ TDMg:RxD (Serial) (Primary Option)	11	PD24	UPC1:RxAddr[1] UTOPIA slave UPC1:RADR[1] POS slave	10	GND
	OUT	GPO_P E8	01	—	—	—	—	—	—	UPC1:TxAddr[1] UTOPIA master UPC1:TADR[1] POS master	01	—	

Table 3-20. Port E Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPAREx[SELn]=0 0			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input
PE9	IN	GPI_PE 9	10	—	UCC7:RxD[3] Enet UCC7:RxD[3] SER	10	GND	TDMf:RxD[0]/ TDMf:RxD (Serial) (Primary Option)	11	PD19	UPC1:STPA POS master	10	GND
	OUT	GPO_P E9	01	—	TDMg:CLKO	01					UPC1:STPA POS slave	01	—
PE10	IN	GPI_PE 10	10	—	UCC7:COL Enet	10	GND	TDMf:TxD[0]/ TDMf:TxD (Serial) (Primary Option)	11	PD22	UPC1:RVAL POS master (Secondary Option)	10	GND
	OUT	GPO_P E10	01	—	SI:STRB7	01					UPC1:RVAL POS slave	01	—
PE11	IN	GPI_PE 11	10	—	UCC7:CRS Enet	10	GND	UPC1:TxAddr[3] UTOPIA slave UPC1:TADR[3] POS slave	10	GND	TDMc:RxD[0]/ TDMc:RxD (Serial) (Primary Option)	11	PB5
	OUT	GPO_P E11	01	—	UPC1:RxAddr[3] UTOPIA master UPC1:RADR[3] POS master	01	—	TDMg:RQ	01	—			
PE12	IN	GPI_PE 12	10	—	UCC7:RX_DV Enet UCC7:CTS SER	10	GND	UPC1:RxAddr[2] UTOPIA slave UPC1:RADR[2] POS slave	10	GND	TDMh:RxD[0]/ TDMh:RxD (Serial) (Secondary Option)	11	GND
	OUT	GPO_P E12	01	—	—	—	—	UPC1:TxAddr[2] UTOPIA master UPC1:TADR[2] POS master	01	—			
PE13	IN	GPI_PE 13	10	—	UCC7:RX_ER Enet UCC7:CD SER	10	GND	UPC1:RxAddr[4] UTOPIA slave UPC1:RADR[4] POS slave	10	GND	TDMc:RSYNC (Primary Option)	10	PB2
	OUT	GPO_P E13	01	—	—	—	—	—	—	—	UPC1:TxAddr[4] UTOPIA master UPC1:TADR[4] POS master	01	—

Table 3-20. Port E Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPAREx[SELn]=0			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input
PE14	IN	GPI_PE14	10	—	—	—	TDMe:RSYNC (Primary Option)	10	PD2	UPC2:RxData[0] UTOPIA 8 ¹ UPC2:RxData[8] UTOPIA 16 ¹ UPC2:RDAT[8] POS (Secondary Option)	10	GND	
	OUT	GPO_PE14	01	—	PTP_ALARM1 (Secondary Option)	01	—	UCC8:TxD[0] Enet UCC8:TxD[0]/UCC8:TxD (Serial) SER	01	—	TDMh:TxD[1]	01	—
PE15	IN	GPI_PE15	10	—	TDMe:TSYNC/G RANT (Primary Option)	10	PD3	TDMh:RxD[2]	10	GND	UPC2:RxEnb UTOPIA slave UPC2:RENb POS slave (Secondary Option)	10	VCC
	OUT	GPO_PE15	01	—	UCC8:TxD[1] Enet UCC8:TxD[1] SER	01	—	UPC1:TxEnb[2] UTOPIA UPC1:TENb[2] POS	01	—	UPC2:TxEnb[0] UTOPIA master UPC2:TENb[0] POS master	01	—
PE16	IN	GPI_PE16	10	—	UPC1:RxClav[1] UTOPIA UPC1:PRPA[1]/DRPA[1] POS (Primary Option)	10	PA8	TDMh:RSYNC (Primary Option)	10	PE6	UPC2:TxClav[0] UTOPIA master UPC2:PTPA[0]/TPA[0] POS master (Secondary Option)	10	GND
	OUT	GPO_PE16	01	—	UCC8:TxD[2] Enet UCC8:TxD[2] SER	01	—	UCC5:TxD[0] Enet UCC5:TxD[0]/UCC5:TxD (Serial) SER	01	—	UPC2:RxClav UTOPIA slave UPC2:PRPA/DRPA POS slave	01	—

Table 3-20. Port E Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPAREx[SELn]=0			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input
PE17	IN	GPI_PE17	10	—	UPC1:TxClav[1] UTOPIA UPC1:PTPA[1]/D TPA[1] POS (Primary Option)	10	PA13	TDMh:TSYNC/G RANT (Primary Option)	10	PE4	UART2:SIN (Secondary Option)	10	VCC
	OUT	GPO_P E17	01	—	UCC8:TxData[3] Enet UCC8:TxData[3] SER	01	—	UPC2:TxData[7] UTOPIA 8 ¹ UPC2:TxData[15] UTOPIA 16 ¹ UPC2:TDAT[15] POS	01	—	UCC5:TxData[1] Enet UCC5:TxData[1] SER	01	—
PE18	IN	GPI_PE18	10	—	—	—	—	—	—	—	TDMe:TxData[0]/ TDMe:TxData (Serial) (Primary Option)	11	PD5
	OUT	GPO_P E18	01	—	UCC8:TX_EN Enet UCC8:RTS SER	01	—	UPC1:RxEnb[3] UTOPIA UPC1:RENb[3] POS	01	—	—	—	—
PE19	IN	GPI_PE19	10	—	UPC2:RxData[2] UTOPIA 8 ¹ UPC2:RxData[10] UTOPIA 16 ¹ UPC2:RDAT[10] ¹ POS (Secondary Option)	10	GND	TDMh:RxData[0]/ TDMh:RxData (Serial) (Primary Option)	11	PE12	UCC5:RxData[1] Enet UCC5:RxData[1] SER (Primary Option)	10	PD7
	OUT	GPO_P E19	01	—	UCC8:TX_ER Enet	01	—	—	—	—	UPC1:RxEnb[1] UTOPIA UPC1:RENb[1] POS	01	—

Table 3-20. Port E Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPAREx[SELn]=0			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input
PE20	IN	GPI_PE20	10	—	UCC8:RxD[0] Enet UCC8:RxD[0]/ UCC8:RxD (Serial) SER	10	GND	TDMh:RxD[1]	10	GND	UPC2:RxAddr[4] UTOPIA slave UPC2:RADR[4] POS slave (Secondary Option)	10	GND
	OUT	GPO_PE20	01	—	—	—	—	PTP_PPS1 (Secondary Option)	01	—	UPC2:TxAddr[4] UTOPIA master UPC2:TADR[4] POS master	01	—
PE21	IN	GPI_PE21	10	—	UCC8:RxD[1] Enet UCC8:RxD[1] SER	10	GND	UPC1:RxClav[3] UTOPIA UPC1:PRPA[3]/ DRPA[3] POS (Primary Option)	10	PC17	TDMe:RxD[0]/ TDMe:RxD (Serial) (Primary Option)	11	PD8
	OUT	GPO_PE21	01	—	TDMh:TxD[2]	01	—	UPC2:TxData[3] UTOPIA 8 ¹ UPC2:TxData[11] UTOPIA 16 ¹ UPC2:TDAT[11] POS	01	—			
PE22	IN	GPI_PE22	10	—	UCC8:RxD[2] Enet UCC8:RxD[2] SER	10	GND	UCC5:RxD[0] Enet UCC5:RxD[0]/ UCC5:RxD (Serial) SER (Primary Option)	10	PD6	TDMh:TxD[0]/ TDMh:TxD (Serial) (Primary Option)	11	PE0
	OUT	GPO_PE22	01	—	UPC1:TxEnb[1] UTOPIA UPC1:TENB[1] POS	01	—	UPC2:TxData[5] UTOPIA 8 ¹ UPC2:TxData[13] UTOPIA 16 ¹ UPC2:TDAT[13] POS	01	—			

Table 3-20. Port E Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPAREx[SELn]=0 0			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input
PE23	IN	GPI_PE23	10	—	UCC8:RxD[3] Enet UCC8:RxD[3] SER	10	GND	UPC1:RxClav[2] UTOPIA UPC1:PRPA[2]/DRPA[2] POS (Primary Option)	10	PA22	UART2:CTS (Secondary Option)	10	GND
	OUT	GPO_PE23	01	—	TDMh:CLKO	01	—	UPC2:TxData[1] UTOPIA 8 ¹ UPC2:TxData[9] UTOPIA 16 ¹ UPC2:TDAT[9] POS	01	—	UCC5:TX_EN Enet UCC5:RTS SER	01	—
PE24	IN	GPI_PE24	10	—	UCC8:COL Enet	10	GND	UPC1:TxClav[2] UTOPIA UPC1:PTPA[2]/DTPA[2] POS (Primary Option)	10	PA27	UCC5:RX_DV Enet UCC5:CTS SER (Primary Option)	10	PD12
	OUT	GPO_PE24	01	—	SI:STRB8	01	—	UART2:SOUT	01	—	UPC2:TxData[0] UTOPIA 8 ¹ UPC2:TxData[8] UTOPIA 16 ¹ UPC2:TDAT[8] POS	01	—
PE25	IN	GPI_PE25	10	—	UCC8:CRS Enet	10	GND	UPC2:RxSOC UTOPIA master UPC2:TxSOC UTOPIA slave UPC2:RSOP POS master UPC2:TSOP POS slave (Secondary Option)	10	GND	UCC5:RX_ER Enet UCC5:CD SER (Primary Option)	10	PD13
	OUT	GPO_PE25	01	—	UPC1:RxE $\overline{\text{nb}}$ [2] UTOPIA UPC1:RENB[2] POS	01	—	TDMh:RQ	01	—	UART2:RTS	01	—

Table 3-20. Port E Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPAREx[SELn]=0			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input	Function	CPDIRXE[DIRn]	Default Input
PE26	IN	GPI_PE26	10	—	UCC8:RX_DV Enet UCC8:CTS SER	10	GND	TDMh:RxD[3]	10	GND	UPC2:TxEnb UTOPIA slave UPC2:TENB POS slave (Secondary Option)	10	VCC
	OUT	GPO_P E26	01	—	PTP_REF_CLK (Secondary Option)	01	—	UPC1:TxEnb[3] UTOPIA UPC1:TENB[3] POS	01	—	UPC2:RxEnb[0] UTOPIA master UPC2:RENb[0] POS master	01	—
PE27	IN	GPI_PE27	10	—	UCC8:RX_ER Enet UCC8:CD SER	10	GND	UPC1:TxClav[3] UTOPIA UPC1:PTPA[3]/ TPA[3] POS (Primary Option)	10	PC16	UPC2:RxClav[0] UTOPIA master UPC2:PRPA[0]/ DRPA[0] POS master (Secondary Option)	10	GND
	OUT	GPO_P E27	01	—	DMA_DDONE1 (Secondary Option)	01	—	TDMh:TxD[3]	01	—	UPC2:TxClav UTOPIA slave UPC2:PTPA/ DTPA POS slave	01	—
PE28	IN	GPI_PE28	10	—	—	—	—	—	—	—	SPI:SPIMOSI	11	VCC
	OUT	GPO_P E28	01	—	—	—	—	—	—	—			
PE29	IN	GPI_PE29	10	—	—	—	—	—	—	—	SPI:SPIMISO	11	SPIMOSI
	OUT	GPO_P E29	01	—	—	—	—	—	—	—			
PE30	IN	GPI_PE30	10	—	—	—	—	—	—	—	SPI:SPICLK	11	GND
	OUT	GPO_P E30	01	—	—	—	—	—	—	—			

Table 3-20. Port E Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPAREx[SELn]=0 0			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input
PE31	IN	GPI_PE31	10	—	Timer1: TIN2 Timer2: TIN1 (Primary Option)	10	PD16	—	—	—	SPI: SPISEL	10	VCC
	OUT	GPO_PE31	01	—	SI: STRB7	01	—	Timer1: $\overline{\text{TOUT3}}$	01	—	—	—	—

¹ This pin name applies to UTOPIA master mode only. For UTOPIA slave mode, replace TxData with RxData and RxData with TxData.

Table 3-21. Port F Dedicated Pin Assignment

Pin	Pin Functions												
	Direction	CPPARFx[SELn]=00			CPPARFx[SELn]=01			CPPARFx[SELn]=10			CPPARFx[SELn]=11		
		Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input
PF0	IN	GPI_PF0	10	—	Timer1: $\overline{\text{TGATE1}}$ Timer2: $\overline{\text{TGATE2}}$ (Primary Option)	10	PC1	UPC2:RxData[5] UTOPIA 16 ¹ UPC2:RDAT[5] POS	10	GND	UPC1:RMODE POS master UPC1:TMODE POS slave (Primary Option)	10	PC23
	OUT	GPO_PF0	01	—	—	—	—	UART2:SOUT	01	—	DMA_DACK1 (Secondary Option)	01	—
PF1	IN	GPI_PF1	10	—	UPC2:RxData[1] UTOPIA 16 ¹ UPC2:RDAT[1] POS	10	GND	Timer1:TIN1 Timer2:TIN2 (Primary Option)	10	PC0	UART2: $\overline{\text{CTS}}$ (Primary Option)	10	PE23
	OUT	GPO_PF1	01	—	—	—	—	UPC1:TMODE POS master UPC1:RMODE POS slave	01	—	DMA_DDONE1 (Primary Option)	01	—

Table 3-21. Port F Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARF _x [SEL _n]=00			CPPARF _x [SEL _n]=01			CPPARF _x [SEL _n]=10			CPPARF _x [SEL _n]=11		
		Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input
PF2	IN	GPI_PF2	10	—	—	—	UPC2:RxData[0] UTOPIA 16 ¹ UPC2:RDAT[0] POS	10	GND	UPC1:RVAL POS master (Primary Option)	10	PE10	
	OUT	GPO_PF2	01	—	UART2:RTS	01	—	—	—	UPC1:RVAL POS slave	01	—	
PF3	IN	GPI_PF3	10	—	DMA_DREQ1 (Secondary Option)	10	VCC	UART2:SIN (Primary Option)	10	PE17	UPC2:RxData[1] UTOPIA 8 ¹ UPC2:RxData[9] UTOPIA 16 ¹ UPC2:RDAT[9] POS (Primary Option)	10	PB1
	OUT	GPO_PF3	01	—	Timer1:TOUT2	01	—	—	—	—	Timer2:TOUT1	01	—
PF4	IN	GPI_PF4	10	—	—	—	UPC2:RxData[6] UTOPIA 16 ¹ UPC2:RDAT[6] POS	10	GND	M66EN	10	VCC	
	OUT	GPO_PF4	01	—	—	—	—	—	—	—	—	—	
PF5	IN	GPI_PF5	10	—	UPC2:TxAddr[0] UTOPIA slave UPC2:TADR[0] POS slave (Primary Option)	10	PB0	—	—	—	—	—	
	OUT	GPO_PF5	01	—	UPC2:RxAddr[0] UTOPIA master UPC2:RADR[0] POS master	01	—	UCC2:TxD[0]/T CG[0] Enet UCC2:TxD[0]/ UCC2:TxD (Serial) SER	01	—	PCI_INTA/IRQ_ OUT	01	—

Table 3-21. Port F Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARF _x [SEL _n]=00			CPPARF _x [SEL _n]=01			CPPARF _x [SEL _n]=10			CPPARF _x [SEL _n]=11		
		Function	CPDIRx[FIDIRn]	Default Input	Function	CPDIRx[FIDIRn]	Default Input	Function	CPDIRx[FIDIRn]	Default Input	Function	CPDIRx[FIDIRn]	Default Input
PF6	IN	GPI_Pf6	10	—	UPC2:TxAddr[1] UTOPIA slave UPC2:TADR[1] POS slave (Primary Option)	10	PA8	—	—	—	—	—	
	OUT	GPO_Pf6	01	—	UPC2:RxAddr[1] UTOPIA master UPC2:RADR[1] POS master	01	—	UCC2:TxD[1]/T CG[1] Enet UCC2:TxD[1] SER	01	—	PCI_RESET_O UT	01	—
PF7	IN	GPI_Pf7	10	—	—	—	—	—	—	—	PCI_C/ \overline{BE} [0]	11	GND
	OUT	GPO_Pf7	01	—	UPC2:TxData[3] UTOPIA 16 ¹ UPC2:TDAT[3] POS	01	—	—	—	—	—	—	—
PF8	IN	GPI_Pf8	10	—	—	—	—	—	—	—	PCI_C/ \overline{BE} [1]	11	GND
	OUT	GPO_Pf8	01	—	UPC2:TxData[2] UTOPIA 16 ¹ UPC2:TDAT[2] POS	01	—	—	—	—	—	—	—
PF9	IN	GPI_Pf9	10	—	—	—	—	—	—	—	PCI_C/ \overline{BE} [2]	11	GND
	OUT	GPO_Pf9	01	—	UPC2:TxData[1] UTOPIA 16 ¹ UPC2:TDAT[1] POS	01	—	—	—	—	—	—	—
PF10	IN	GPI_Pf10	10	—	—	—	—	—	—	—	PCI_C/ \overline{BE} [3]	11	GND
	OUT	GPO_Pf10	01	—	UPC2:TxData[0] UTOPIA 16 ¹ UPC2:TDAT[0] POS	01	—	—	—	—	—	—	—

Table 3-21. Port F Dedicated Pin Assignment (continued)

Pin	Pin Functions														
	Direction	CPPARFx[SELn]=00			CPPARFx[SELn]=01			CPPARFx[SELn]=10			CPPARFx[SELn]=11				
		Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input		
PF11	IN	GPI_PF11	10	—	UPC2:RxSOC UTOPIA master UPC2:TxSOC UTOPIA slave UPC2:RSOP POS master UPC2:TSOP POS slave (Primary Option)	10	PE25	—	—	—	—	—	PCI_PAR	11	GND
	OUT	GPO_PF1 1	01	—	—	—	—	—	—	—	—	—	—	—	—
PF12	IN	GPI_PF12	10	—	UPC2:TxEnb UTOPIA slave UPC2:TENB POS slave (Primary Option)	10	PE26	—	—	—	—	—	PCI_FRAME	11	VCC
	OUT	GPO_PF1 2	01	—	UPC2:RxEnb[0] UTOPIA master UPC2:RENb[0] POS master	01	—	—	—	—	—	—	—	—	—
PF13	IN	GPI_PF13	10	—	UPC2:RxClav[0] UTOPIA master UPC2:PRPA[0]/ DRPA[0] POS master (Primary Option)	10	PE27	—	—	—	—	—	PCI_TRDY	11	VCC
	OUT	GPO_PF1 3	01	—	UPC2:TxClav UTOPIA slave UPC2:PTPA/ DTPA POS slave	01	—	—	—	—	—	—	—	—	—

Table 3-21. Port F Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARFx[SELn]=00			CPPARFx[SELn]=01			CPPARFx[SELn]=10			CPPARFx[SELn]=11		
		Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input
PF14	IN	GPI_PF14	10	—	UPC2:RxData[7] UTOPIA 8 ¹ UPC2:RxData[15] UTOPIA 16 ¹ UPC2:RDAT[15] POS (Primary Option)	10	PB2	—	—	—	PCI_IRDY	11	VCC
	OUT	GPO_PF14	01	—	—	—	—	—	—	—	—	—	—
PF15	IN	GPI_PF15	10	—	UPC2:RxData[6] UTOPIA 8 ¹ UPC2:RxData[14] UTOPIA 16 ¹ UPC2:RDAT[14] POS (Primary Option)	10	PB3	—	—	—	PCI_STOP	11	VCC
	OUT	GPO_PF15	01	—	—	—	—	—	—	—	—	—	—
PF16	IN	GPI_PF16	10	—	UPC2:RxData[5] UTOPIA 8 ¹ UPC2:RxData[13] UTOPIA 16 ¹ UPC2:RDAT[13] POS (Primary Option)	10	PC25	—	—	—	PCI_DEVSEL	11	VCC
	OUT	GPO_PF16	01	—	—	—	—	—	—	—	—	—	—
PF17	IN	GPI_PF17	10	—	UPC2:RxData[4] UTOPIA 8 ¹ UPC2:RxData[12] UTOPIA 16 ¹ UPC2:RDAT[12] POS (Primary Option)	10	PB9	—	—	—	PCI_IDSEL	10	GND
	OUT	GPO_PF17	01	—	—	—	—	—	—	—	—	—	—

Table 3-21. Port F Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARF _x [SEL _n]=00			CPPARF _x [SEL _n]=01			CPPARF _x [SEL _n]=10			CPPARF _x [SEL _n]=11		
		Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input
PF18	IN	GPI_PF18	10	—	UPC2:RxData[3] UTOPIA 8 ¹ UPC2:RxData[1 1] UTOPIA 16 ¹ UPC2:RDAT[11] POS (Primary Option)	10	PB10	—	—	—	PCI_SERR	11	VCC
	OUT	GPO_PF18	01	—	—	—	—	—	—	—	—	—	—
PF19	IN	GPI_PF19	10	—	UPC2:RxData[2] UTOPIA 8 ¹ UPC2:RxData[1 0] UTOPIA 16 ¹ UPC2:RDAT[10] POS (Primary Option)	10	PE19	—	—	—	PCI_PERR	11	VCC
	OUT	GPO_PF19	01	—	—	—	—	—	—	—	—	—	—
PF20	IN	UPC2:STPA POS master (Primary Option)	10	PA20	—	—	—	UCC2:Gigabit RX_CLK Enet (Primary Option)	10	PA31	PCI_REQ[0]	11	VCC
	OUT	GPO_PF20	01	—	UPC2:STPA POS slave	01	—	UPC2:TxEnb[2] UTOPIA UPC2:TENB[2] POS	01	—	—	—	—
PF21	IN	GPI_PF21	10	—	UPC2:TxAddr[2] UTOPIA slave UPC2:TADR[2] POS slave (Primary Option)	10	PA22	CPCI_HS_ES	10	GND	PCI_REQ[1]	10	VCC
	OUT	GPO_PF21	01	—	UCC2:TxD[2]/T CG[2] Enet UCC2:TxD[2] SER	01	—	UPC2:RxAddr[2] UTOPIA master UPC2:RADR[2] POS master	01	—	—	—	—

Table 3-21. Port F Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARF _x [SEL _n]=00			CPPARF _x [SEL _n]=01			CPPARF _x [SEL _n]=10			CPPARF _x [SEL _n]=11		
		Function	CPDIRx[FIDIRn]	Default Input	Function	CPDIRx[FIDIRn]	Default Input	Function	CPDIRx[FIDIRn]	Default Input	Function	CPDIRx[FIDIRn]	Default Input
PF22	IN	GPI_PF22	10	—	UPC2:RxAddr[3] UTOPIA slave UPC2:RADR[3] POS slave (Primary Option)	10	PA16	—	—	—	PCI_REQ[2]	10	VCC
	OUT	GPO_PF22	01	—	UPC2:TxAddr[3] UTOPIA master UPC2:TADR[3] POS master	01	—	UCC2:TX_EN/T CG[8] Enet UCC2:RTS SER	01	—	—	—	—
PF23	IN	GPI_PF23	10	—	UPC2:TxAddr[3] UTOPIA slave UPC2:TADR[3] POS slave (Primary Option)	10	PA14	UCC2:RxD[0]/R CG[0] Enet UCC2:RxD[0]/ UCC2:RxD (Serial) SER (Secondary Option)	10	GND	PCI_GNT[0]	11	VCC
	OUT	GPO_PF23	01	—	UPC2:RxAddr[3] UTOPIA master UPC2:RADR[3] POS master	01	—	—	—	—	—	—	—
PF24	IN	GPI_PF24	10	—	UCC2:RxD[1]/R CG[1] Enet UCC2:RxD[1] SER (Secondary Option)	10	GND	UPC2:TxAddr[4] UTOPIA slave UPC2:TADR[4] POS slave (Primary Option)	10	PC24	—	—	—
	OUT	GPO_PF24	01	—	UPC2:RxAddr[4] UTOPIA master UPC2:RADR[4] POS master	01	—	CPCI_HS_LED	01	—	PCI_GNT[1]	01	—

Table 3-21. Port F Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARF _x [SEL _n]=00			CPPARF _x [SEL _n]=01			CPPARF _x [SEL _n]=10			CPPARF _x [SEL _n]=11		
		Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input	Function	CPDIRx[DIRn]	Default Input
PF25	IN	GPI_PF25	10	—	UCC2:Rx \overline{D} [3]/R CG[3] Enet UCC2:Rx \overline{D} [3] SER (Secondary Option)	10	GND	—	—	—	—	—	
	OUT	UPC2:TE OP POS master UPC2:RE OP POS slave	01	—	UPC2:RxEnb[1] UTOPIA UPC2:RENb[1] POS	01	—	CPCI_HS_ENU M	01	—	$\overline{\text{PCI_GNT}}[2]$	01	—
PF26	IN	GPI_PF26	10	—	—	—	—	DMA_DREQ1 (Primary Option)	10	PF3	—	—	
	OUT	GPO_PF2 6	01	—	UPC2:RxEnb[3] UTOPIA UPC2:RENb[3] POS	01	—	UCC2:GTx Clock Enet	01	—	PCI_CLK_OUT[0]	01	—
PF27	IN	GPI_PF27	10	—	—	—	—	USB:RXD (Primary Option)	10	PB10	—	—	
	OUT	GPO_PF2 7	01	—	UPC2:TxEnb[3] UTOPIA UPC2:TENb[3] POS	01	—	DMA_DACK1 (Primary Option)	01	—	PCI_CLK_OUT[1]	01	—
PF28	IN	GPI_PF28	10	—	Timer1: $\overline{\text{TGATE3}}$ Timer2: $\overline{\text{TGATE4}}$ (Primary Option)	10	PC25	—	—	—	UPC2:RxClav[3] UTOPIA UPC2:PRPA[3]/ DRPA[3] POS	10	GND
	OUT	GPO_PF2 8	01	—	—	—	—	UPC2:CLKO	01	—	PCI_CLK_OUT[2]	01	—

Table 3-21. Port F Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARF _x [SEL _n]=00			CPPARF _x [SEL _n]=01			CPPARF _x [SEL _n]=10			CPPARF _x [SEL _n]=11		
		Function	CPDIR _x F[DIR _n]	Default Input	Function	CPDIR _x F[DIR _n]	Default Input	Function	CPDIR _x F[DIR _n]	Default Input	Function	CPDIR _x F[DIR _n]	Default Input
PF29	IN	GPI_P29	10	—	USB:RN (Primary Option)	10	PB11	—	—	—	UPC2:TxClav[3] UTOPIA UPC2:PTPA[3]/ DTPA[3] POS	10	GND
	OUT	GPO_P29	01	—	—	—	—	PTP_PPS3 (Secondary Option)	01	—	PCI_SYNC_OUT	01	—

¹ This pin name applies to UTOPIA master mode only. For UTOPIA slave mode, replace TxData with RxData and RxData with TxData.

Table 3-22. Port G Dedicated Pin Assignment

Pin	Pin Functions												
	Direction	CPPARG _x [SEL _n]=00			CPPARG _x [SEL _n]=01			CPPARG _x [SEL _n]=10			CPPARG _x [SEL _n]=11		
		Function	CPDIR _x G[DIR _n]	Default Input	Function	CPDIR _x G[DIR _n]	Default Input	Function	CPDIR _x G[DIR _n]	Default Input	Function	CPDIR _x G[DIR _n]	Default Input
PG0	IN	GPI_PG0	10	—	UPC2:RxData[0] UTOPIA 8 ¹ UPC2:RxData[8] UTOPIA 16 ¹ UPC2:RDAT[8] POS (Primary Option)	10	PE14	—	—	—	PCI_AD[0]	11	GND
	OUT	GPO_PG0	01	—	—	—	—	—	—	—	—	—	—
PG1	IN	GPI_PG1	10	—	UPC2:RxData[4] UTOPIA 16 ¹ UPC2:RDAT[4] POS	10	GND	—	—	—	PCI_AD[1]	11	GND
	OUT	GPO_PG1	01	—	—	—	—	—	—	—	—	—	—

Table 3-22. Port G Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARGx[SELn]=00			CPPARGx[SELn]=01			CPPARGx[SELn]=10			CPPARGx[SELn]=11		
		Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input
PG2	IN	GPI_PG2	10	—	UPC2:RxData[3] UTOPIA 16 ¹ UPC2:RDAT[3] POS	10	GND	—	—	—	PCI_AD[2]	11	GND
	OUT	GPO_PG2	01	—	—	—	—	—	—	—	—	—	—
PG3	IN	GPI_PG3	10	—	UPC2:RxAddr[0] UTOPIA slave UPC2:RADR[0] POS slave (Primary Option)	10	PB4	—	—	—	PCI_AD[3]	11	GND
	OUT	GPO_PG3	01	—	UPC2:TxAddr[0] UTOPIA master UPC2:TADR[0] POS master	01	—	—	—	—	—	—	—
PG4	IN	GPI_PG4	10	—	—	—	—	UPC2:RxAddr[1] UTOPIA slave UPC2:RADR[1] POS slave (Primary Option)	10	PD17	PCI_AD[4]	11	GND
	OUT	GPO_PG4	01	—	—	—	—	UPC2:TxAddr[1] UTOPIA master UPC2:TADR[1] POS master	01	—	—	—	—
PG5	IN	GPI_PG5	10	—	USB:RP (Primary Option)	10	PB9	UPC2:RxAddr[2] UTOPIA slave UPC2:RADR[2] POS slave (Primary Option)	10	PD16	PCI_AD[5]	11	GND
	OUT	GPO_PG5	01	—	UPC2:TxAddr[2] UTOPIA master UPC2:TADR[2] POS master	01	—	—	—	—	—	—	—

Table 3-22. Port G Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARGx[SELn]=00			CPPARGx[SELn]=01			CPPARGx[SELn]=10			CPPARGx[SELn]=11		
		Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input
PG6	IN	GPI_PG6	10	—	—	—	—	UPC2:RxData[7] UTOPIA 16 ¹ UPC2:RDAT[7] POS	10	GND	PCI_AD[6]	11	GND
	OUT	GPO_PG6	01	—	—	—	—	—	—	—	—	—	—
PG7	IN	GPI_PG7	10	—	UPC2:RxAddr[4] UTOPIA slave UPC2:RADR[4] POS slave (Primary Option)	10	PE20	—	—	—	PCI_AD[7]	11	GND
	OUT	GPO_PG7	01	—	UPC2:TxAddr[4] UTOPIA master UPC2:TADR[4] POS master	01	—	—	—	—	—	—	—
PG8	IN	GPI_PG8	10	—	UPC2:RxData[2] UTOPIA 16 ¹ UPC2:RDAT[2] POS	10	GND	—	—	—	PCI_AD[8]	11	GND
	OUT	GPO_PG8	01	—	—	—	—	—	—	—	—	—	—
PG9	IN	GPI_PG9	10	—	—	—	—	—	—	—	PCI_AD[9]	11	GND
	OUT	GPO_PG9	01	—	UPC2:TxData[3] UTOPIA 8 ¹ UPC2:TxData[11]] UTOPIA 16 ¹ UPC2:TDAT[11] POS	01	—	—	—	—	—	—	—
PG10	IN	GPI_PG10	10	—	—	—	—	—	—	—	PCI_AD[10]	11	GND
	OUT	GPO_PG10	01	—	UPC2:TxData[5] UTOPIA 16 ¹ UPC2:TDAT[5] POS	01	—	—	—	—	—	—	—

Table 3-22. Port G Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARGx[SELn]=00			CPPARGx[SELn]=01			CPPARGx[SELn]=10			CPPARGx[SELn]=11		
		Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input
PG11	IN	GPI_PG1 1	10	—	UPC2:RxClav[1] UTOPIA UPC2:PRPA[1]/ DRPA[1] POS	10	GND	UPC2:RMOD POS master UPC2:TMOD POS slave (Primary Option)	10	PC10	PCI_AD[11]	11	GND
	OUT	USB:OE	01	—	—	—	—	—	—	—	—	—	—
PG12	IN	GPI_PG1 2	10	—	UPC2:TxClav[2] UTOPIA UPC2:PTPA[2]/D TPA[2] POS	10	GND	UPC2:RVAL POS master (Primary Option)	10	PA31	PCI_AD[12]	11	GND
	OUT	USB:TP	01	—	UPC2:RVAL POS slave	01	—	—	—	—	—	—	—
PG13	IN	GPI_PG1 3	10	—	UPC2:TxClav[1] UTOPIA UPC2:PTPA[1]/D TPA[1] POS	10	GND	—	—	—	PCI_AD[13]	11	GND
	OUT	GPO_PG 13	01	—	UPC2:TERR POS master UPC2:RERR POS slave	01	—	USB:TN	01	—	—	—	—
PG14	IN	GPI_PG1 4	10	—	—	—	—	—	—	—	PCI_AD[14]	11	GND
	OUT	GPO_PG 14	01	—	UPC2:TxData[4] UTOPIA 16 ¹ UPC2:TDAT[4] POS	01	—	—	—	—	—	—	—

Table 3-22. Port G Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARGx[SELn]=00			CPPARGx[SELn]=01			CPPARGx[SELn]=10			CPPARGx[SELn]=11		
		Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input
PG15	IN	GPI_PG15	10	—	—	—	UPC2:RxPrty UTOPIA master UPC2:TxPrty UTOPIA slave UPC2:RPRTY POS master UPC2:TPRTY POS slave (Primary Option)	10	PB12	PCI_AD[15]	11	GND	
	OUT	GPO_PG15	01	—	—	—	—	—	—	—	—	—	
PG16	IN	GPI_PG16	10	—	—	—	—	—	—	PCI_AD[16]	11	GND	
	OUT	GPO_PG16	01	—	UPC2:TxPrty UTOPIA master UPC2:RxPrty UTOPIA slave UPC2:TPRTY POS master UPC2:RPRTY POS slave	01	—	—	—	—	—	—	
PG17	IN	GPI_PG17	10	—	—	—	—	—	—	PCI_AD[17]	11	GND	
	OUT	GPO_PG17	01	—	UPC2:TxSOC UTOPIA master UPC2:RxSOC UTOPIA slave UPC2:TSOP POS master UPC2:RSOP POS slave	01	—	—	—	—	—	—	
PG18	IN	GPI_PG18	10	—	—	—	UPC2:RxEnb UTOPIA slave UPC2:RENB POS slave (Primary Option)	10	PE15	PCI_AD[18]	11	GND	
	OUT	GPO_PG18	01	—	—	—	UPC2:TxEnb[0] UTOPIA master UPC2:TENB[0] POS master	01	—	—	—	—	

Table 3-22. Port G Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARGx[SELn]=00			CPPARGx[SELn]=01			CPPARGx[SELn]=10			CPPARGx[SELn]=11		
		Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input
PG19	IN	GPI_PG19	10	—	UPC2:TxClav[0] UTOPIA master UPC2:PTPA[0]/DTPA[0] POS master (Primary Option)	10	PE16	—	—	—	PCI_AD[19]	11	GND
	OUT	GPO_PG19	01	—	UPC2:RxClav UTOPIA slave UPC2:PRPA/DRPA POS slave	01	—	—	—	—			
PG20	IN	GPI_PG20	10	—	—	11	GND	—	—	—	PCI_AD[20]	11	GND
	OUT	GPO_PG20	01	—				UPC2:TxData[7] UTOPIA 8 ¹ UPC2:TxData[15]] UTOPIA 16 ¹ UPC2:TDAT[15] POS	01	—			
PG21	IN	GPI_PG21	10	—	—	11	GND	—	—	—	PCI_AD[21]	11	GND
	OUT	GPO_PG21	01	—				UPC2:TxData[6] UTOPIA 8 ¹ UPC2:TxData[14]] UTOPIA 16 ¹ UPC2:TDAT[14] POS	01	—			
PG22	IN	GPI_PG22	10	—	—	11	GND	—	—	—	PCI_AD[22]	11	GND
	OUT	GPO_PG22	01	—				UPC2:TxData[5] UTOPIA 8 ¹ UPC2:TxData[13]] UTOPIA 16 ¹ UPC2:TDAT[13] POS	01	—			

Table 3-22. Port G Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARGx[SELn]=00			CPPARGx[SELn]=01			CPPARGx[SELn]=10			CPPARGx[SELn]=11		
		Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input
PG23	IN	GPI_PG2 3	10	—	—	11	GND	—	—	—	PCI_AD[23]	11	GND
	OUT	GPO_PG 23	01	—	—	—	—	UPC2:TxData[4] UTOPIA 8 ¹ UPC2:TxData[12] UTOPIA 16 ¹ UPC2:TDAT[12] POS	01	—			
PG24	IN	GPI_PG2 4	10	—	UPC2:RxClav[2] UTOPIA UPC2:PRPA[2]/ DRPA[2] POS	10	GND	UPC2:RERR POS master UPC2:TERR POS slave (Primary Option)	10	PA19	PCI_AD[24]	11	GND
	OUT	GPO_PG 24	01	—	—	—	—	UCC2:TxD[3]/TC G[3] Enet UCC2:TxD[3] SER	01	—			
PG25	IN	GPI_PG2 5	10	—	—	—	—	—	—	—	PCI_AD[25]	11	GND
	OUT	GPO_PG 25	01	—	UPC2:TxData[2] UTOPIA 8 ¹ UPC2:TxData[10] UTOPIA 16 ¹ UPC2:TDAT[10] POS	01	—	—	—	—			
PG26	IN	GPI_PG2 6	10	—	—	—	—	—	—	—	PCI_AD[26]	11	GND
	OUT	GPO_PG 26	01	—	UPC2:TxData[1] UTOPIA 8 ¹ UPC2:TxData[9] UTOPIA 16 ¹ UPC2:TDAT[9] POS	01	—	—	—	—			

Table 3-22. Port G Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARGx[SELn]=00			CPPARGx[SELn]=01			CPPARGx[SELn]=10			CPPARGx[SELn]=11		
		Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input
PG27	IN	GPI_PG27	10	—	—	—	—	—	—	—	PCI_AD[27]	11	GND
	OUT	GPO_PG27	01	—	UPC2:TxData[0] UTOPIA 8 ¹ UPC2:TxData[8] UTOPIA 16 ¹ UPC2:TDAT[8] POS	01	—	—	—	—	—	—	—
PG28	IN	GPI_PG28	10	—	—	—	—	—	—	—	PCI_AD[28]	11	GND
	OUT	GPO_PG28	01	—	UPC2:TxData[7] UTOPIA 16 ¹ UPC2:TDAT[7] POS	01	—	—	—	—	—	—	—
PG29	IN	GPI_PG29	10	—	—	—	—	—	—	—	PCI_AD[29]	11	GND
	OUT	GPO_PG29	01	—	UPC2:TxData[6] UTOPIA 16 ¹ UPC2:TDAT[6] POS	01	—	—	—	—	—	—	—
PG30	IN	GPI_PG30	10	—	UPC2:REOP POS master UPC2:TEOP POS slave (Primary Option)	10	PC14	UCC2:RxD[2]/R CG[2] Enet UCC2:RxD[2] SER (Secondary Option)	10	GND	PCI_AD[30]	11	GND
	OUT	GPO_PG30	01	—	UPC2:TxEnb[1] UTOPIA UPC2:TENB[1] POS	01	—	—	—	—	—	—	—

Table 3-22. Port G Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARGx[SELn]=00			CPPARGx[SELn]=01			CPPARGx[SELn]=10			CPPARGx[SELn]=11		
		Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input
PG31	IN	GPI_PG3 1	10	—				UCC2:RX_DV/R CG[8] Enet UCC2:CTS SER (Secondary Option)	10	GND	PCI_AD[31]	11	GND
	OUT	GPO_PG 31	01	—	UPC2:RxEnb[2] UTOPIA UPC2:RENB[2] POS	01	—	UPC2:TMOD POS master UPC2:RMOD POS slave	01	—			

¹ This pin name applies to UTOPIA master mode only. For UTOPIA slave mode, replace TxData with RxData and RxData with TxData.

Part II

Reset and Configuration

Part II includes the following chapters:

- [Chapter 4, “Reset, Clocking, and Initialization,”](#) describes the hard and soft resets, the power-on reset sequence, power-on reset (POR) configuration, clocking, and initialization of the MPC8360E.
- [Chapter 5, “System Configuration,”](#) describes several functions of the MPC8360E that control the local access windows, system configuration, protection and general utilities.

Chapter 4

Reset, Clocking, and Initialization

The reset, clocking, and control signals offer many options for operating the device. Various modes and features can be configured during hard reset or power-on reset. Most configurable features are loaded to the device through a reset configuration word, and a few device signals are used as reset configuration inputs during the reset sequence.

4.1 External Signals

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

4.1.1 Reset Signals

Table 4-1 describes the reset signals of the device. Section 4.3.2, “Reset Configuration Words,” describes the signals that also function as reset configuration signals.

Table 4-1. System Control Signals

Signal	I/O	Description
PORESET	I	Power-on reset. Initiates the power-on reset flow that resets the device and configures various attributes of the device, including its clock modes.
		State Meaning Asserted—An external agent has triggered a power-on reset sequence. Negated—No power-on reset.
		Timing See the hardware specifications for timing information.
		Reset State Always input.
HRESET	I/O	Hard reset. Causes the device to abort all current internal and external transactions and set most registers to their default values. HRESET can be asserted completely asynchronously with respect to all other signals. The device can detect an external assertion of HRESET while the device is not asserting hard reset. During HRESET, SRESET is asserted. HRESET is an open-drain signal.
		State Meaning Asserted—An external agent or internal hardware has triggered a hard reset. The internal hardware drives HRESET until the sequence completes. Negated—No hard reset.
		Timing Assertion—Occur at any time, asynchronously to any clock. Negation—Must be asserted for at least 32 CLKIN (PCI host mode) or PCI_CLK (PCI agent mode) cycles.
		Requirements An open-drain signal. An external pull-up is required.
		Reset State Output, driven low during power-on and hard reset flows. High impedance after reset flow completes.

Table 4-1. System Control Signals (continued)

Signal	I/O	Description	
$\overline{\text{SRESET}}$	I/O	Soft reset. Causes the device to abort all current internal transactions, set most registers to their default values, and cause the core to enter its reset state. The I/O signal functionality and direction as well as the memory controller operation are unaffected by $\overline{\text{SRESET}}$. $\overline{\text{SRESET}}$ can be asserted completely asynchronously with respect to all other signals. The device can detect an external assertion of $\overline{\text{SRESET}}$ while the device is not asserting hard or soft reset. $\overline{\text{SRESET}}$ is an open-drain signal.	
		State Meaning	Asserted—An external agent or internal hardware has triggered a soft reset sequence. The internal hardware drives $\overline{\text{SRESET}}$ until the sequence completes.
		Timing	Assertion—Occurs at any time, asynchronously to any clock. Negation—Must be asserted for at least 32 CLKIN (PCI host mode) or PCI_CLK (PCI agent mode) cycles.
		Requirements	An open-drain signal. An external pull-up is required.
		Reset State	Output, driven low during power-on, hard reset, and soft reset flows. High impedance after reset flow completes.
CFG_RESET_SOURCE[0:2]	I	Reset configuration word source selection. These signals are on device pins that have other functions when the device is not in reset. They are sampled during the assertion of $\overline{\text{PORESET}}$ to determine the interface from which the device loads the reset configuration words.	
		State Meaning	See Section 4.3.1.1, “Reset Configuration Word Source.”
		Timing	These signals are sampled during the assertion of $\overline{\text{PORESET}}$ after a stable clock is supplied ($\overline{\text{PORESET}}$ flow) and must be pulled high or low by external resistors as long as $\overline{\text{HRESET}}$ is asserted.
		Requirements	During $\overline{\text{PORESET}}$ and $\overline{\text{HRESET}}$ flows, all other signal drivers connected to these signals must be in the high-impedance state. Refer to the hardware specifications for proper resistor values to pull reset configuration signals high or low.
		Reset State	Input during power-on and hard reset flows. Functional signal after reset flow completes.
CFG_CLKIN_DIV	I	Clock in division selection. This signal is located on a device pin that has another function when the device is not in reset state. This signal is sampled during the assertion of $\overline{\text{PORESET}}$ to determine whether CLKIN is divided by two.	
		State Meaning	See Section 4.3.1.2, “CLKIN Division.”
		Timing	This signal is sampled during the assertion of $\overline{\text{PORESET}}$ after a stable clock is supplied ($\overline{\text{PORESET}}$ flow), and it must be pulled high or low by external resistors as long as $\overline{\text{HRESET}}$ is asserted.
		Requirements	During $\overline{\text{PORESET}}$ and $\overline{\text{HRESET}}$ flows, all other signal drivers connected to this signal must be in the high-impedance state. Refer to the hardware specifications for proper resistors values to pull reset configuration signals high or low.
		Reset State	Input during power-on and hard reset flows. Functional signal after reset flow completes.
PCI_MODE	I	A system configuration signal, but its state is sampled during the assertion of $\overline{\text{PORESET}}$ and affects the default values of registers. For a description of this signal, see Table 5-31 and Table 5-32 in Chapter 5, “System Configuration” .	

4.1.2 Clock Signals

In [Table 4-2](#), some clock signals are specific to blocks within the device. Although some of their functionality is described in [Section 4.4, “Clocking,”](#) they are defined in detail in their respective chapters. See [Figure 4-9](#) for the internal distribution of clocks in the device.

Table 4-2. External Clock Signals

Signal	I/O	Description	
CLKIN	I	System clock. In PCI host mode, CLKIN is the primary input clock. CLKIN directly feeds the PCI output clock dividers and is driven out on the PCI_SYNC_OUT signal for de-skewing external PCI clocks routing. In PCI agent mode, this signal should be tied to GND. When the device is in PCI host mode but the PCI_CLK_OUT[0:2] signals are not driven, PCI clock distribution should be done externally and The CLKIN input can be used in case the PCI output clocks dividers functionality is needed. In any case, when CLKIN input is not used, it should be tied to GND. If PCI is not used, the device clock source must be connected to PCI_CLK/PCI_SYNC_IN, not CLKIN.	
		Timing	Assertion/Negation—See the hardware specifications for timing information.
		Requirements	Should be tied low in PCI agent mode.
		Reset State	Always input.
PCI_CLK/ PCI_SYNC_IN	I	PCI clock/ PCI synchronization clock (PCI_CLK/PCI_SYNC_IN). In PCI agent mode or in PCI host mode when the PCI_CLK_OUT[0:2] signals are not driven, PCI_CLK is the primary clock input to the device. In PCI host mode with PCI_CLK_OUT[0:2] driven, PCI_SYNC_IN is connected externally to PCI_SYNC_OUT If PCI is not used, the device clock source must be connected to PCI_CLK/PCI_SYNC_IN, not CLKIN.	
		Timing	Assertion/Negation—See the hardware specifications for timing information
		Reset State	Always input.
PCI_SYNC_OUT	O	Reference PCI output synchronization clock (PCI_SYNC_OUT). In PCI host mode with the PCI_CLK_OUT[0:2] signals driven, PCI_SYNC_OUT is connected externally to PCI_SYNC_IN signal for de-skewing external PCI clocks routing. PCI_SYNC_OUT has the same frequency as CLKIN or CLKIN/2 depending on the state of CFG_CLKIN_DIV at reset. See Section 4.3.1.2, “CLKIN Division.” In PCI agent mode, this signal is typically not used.	
		Timing	Assertion/Negation—See the hardware specifications for timing information.
		Reset State	Always output, toggling in PCI host mode.
PCI_CLK_OUT[0:2]	O	PCI output clocks bank. In PCI host mode, the device provides three separate clock output signals for feeding PCI agent devices. Driving of these clock signals should be enabled during reset by the PCICKDRV bit of the Reset Configuration Word High.	
		Timing	Assertion/Negation—See the hardware specifications for timing information.
		Reset State	Always output. High impedance during and after power-on reset flow. Enabled by a memory-mapped register.

4.2 Functional Description

This section describes the various ways to reset the device, the power-on reset configurations, and clocking.

4.2.1 Reset Operations

The device 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
- System bus monitor reset
- Checkstop reset
- JTAG 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 4.5.1.3, “Reset Status Register \(RSR\),”](#) indicates the last sources to cause a reset.

4.2.1.1 Reset Causes

[Table 4-3](#) describes reset causes.

Table 4-3. Reset Causes

Name	Description
Power-on reset ($\overline{\text{PORESET}}$)	Input signal. Asserting this signal initiates the power-on reset flow that resets the entire device and configures various attributes of the device including its clock modes.
Hard reset ($\overline{\text{HRESET}}$)	A bidirectional I/O signal. The device can detect an external assertion of $\overline{\text{HRESET}}$ only while it is not asserting hard reset. During $\overline{\text{HRESET}}$, $\overline{\text{SRESET}}$ is asserted. $\overline{\text{HRESET}}$ is an open-drain signal.
Soft reset ($\overline{\text{SRESET}}$)	Bidirectional I/O signal. The device can detect an external assertion of $\overline{\text{SRESET}}$ only while it is not asserting hard or soft reset. $\overline{\text{SRESET}}$ is an open-drain signal.
Software watchdog reset	After the device watchdog counts to zero, a software watchdog reset is signaled. The enabled software watchdog event then generates an internal hard reset sequence.
System bus monitor reset	After the device CSB bus monitor reaches a timeout condition, a bus monitor reset is asserted. The enabled bus monitor event then generates an internal hard reset sequence.
Checkstop reset	If the core enters checkstop state and the checkstop reset is enabled ($\text{RMR}[\text{CSRE}] = 1$), the checkstop reset is asserted. The enabled checkstop event then generates an internal hard reset sequence.
JTAG reset	When JTAG logic asserts the JTAG soft reset signal, an internal soft reset sequence is generated.

Table 4-3. Reset Causes (continued)

Name	Description
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).

4.2.1.2 Reset Actions

The reset control logic determines the cause of reset, synchronizes it if necessary, and resets the appropriate internal hardware. Each reset flow has a 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.
- Soft reset initializes the internal logic while maintaining the system configuration.

All reset types generate a reset to the core, and the impact on the application is that the core resets the MSR[IP] to the value in the BMS field of the reset configuration word high, see [Section 4.3.2.2, “Reset Configuration Word High Register \(RCWHR\).”](#)

The memory controllers, system protection logic, interrupt controller, and I/O signals are initialized only on hard reset. Soft reset initializes the internal logic while maintaining the system configuration. Asserting external $\overline{\text{SRESET}}$ generates a hard reset to the core and to the rest of the device. [Table 4-4](#) identifies the reset actions for each reset source.

Table 4-4. Reset Actions

Action	Reset Source		
	Power-On Reset	External Hard Reset Software Watchdog Bus Monitor Checkstop Software Hard Reset	JTAG Reset External Soft Reset
Resets: PLLs, clocks, RTC unit, and error capture registers	Yes	No	No
Resets: DDR, LBC, I/O multiplexors, GTM, PIT, system configuration, and local access windows	Yes	Yes	No
Resets other internal logic	Yes	Yes	Yes
Reset configuration words loaded	Yes	Yes	No
$\overline{\text{HRESET}}$ driven	Yes	Yes	No
$\overline{\text{SRESET}}$ driven	Yes	Yes	Yes
Hard reset to e300 core	Yes	Yes	Yes

4.2.2 Power-On Reset Flow

Assertion of the $\overline{\text{PORESET}}$ external 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 to the device is applied.

Directly after the negation of $\overline{\text{PORESET}}$, the device starts the configuration process. The device asserts $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ throughout the power-on reset process, including configuration. Configuration time varies according to the configuration source and CLKIN (PCI host mode) or PCI_CLK (PCI agent mode) frequency. Initially, the reset configuration inputs are sampled to determine the configuration source and the input clock division mode. Next, the device starts loading the reset configuration words. The system PLL begins to lock according to the clock mode values in the reset configuration word low. When the system PLL is locked, the clock unit starts distributing clock signals in the device. At this stage, the core PLL begins to lock. When it is locked and the reset configuration words are loaded, $\overline{\text{HRESET}}$ is released; $\overline{\text{SRESET}}$ is released 16 clocks later.

The detailed power-on reset (POR) flow for the device is as follows:

1. Power is applied to meet the specifications in the device data sheet.
2. The system asserts $\overline{\text{PORESET}}$ and $\overline{\text{TRST}}$, causing all registers to be initialized to their default states and most I/O drivers to be released to high-impedance.
Some clock, clock enabled, and system control signals remain active.
3. The system applies a stable CLKIN (PCI host mode) or PCI_CLK (PCI agent mode) signal and stable reset configuration inputs (CFG_RESET_SOURCE, CFG_CLKIN_DIV).
4. The system negates $\overline{\text{PORESET}}$ after at least 32 stable CLKIN (PCI host mode) or PCI_CLK (PCI agent mode) clock cycles.
5. The device samples the reset configuration input signals to determine the clock division and the reset configuration words source.
6. The device starts loading the reset configuration words.
Loading time depends on the reset configuration word source.
7. When the reset configuration word low is loaded, the system PLL and QUICC Engine PLL begin lock.
When the system PLL is locked, *csb_clk* is supplied to the core PLL.
8. The core PLL begins to lock.
9. The device drives $\overline{\text{HRESET}}$ asserted until the e300 PLL is locked and the reset configuration words are loaded.
10. The user optionally negates $\overline{\text{HRESET}}$ if it was not negated earlier.
JTAG logic must always be initialized by asserting $\overline{\text{TRST}}$. If the JTAG signals are not used, $\overline{\text{TRST}}$ should be connected directly to $\overline{\text{PORESET}}$. $\overline{\text{TRST}}$ must not remain asserted after the negation of $\overline{\text{PORESET}}$. There is no need to assert the $\overline{\text{SRESET}}$ signal when $\overline{\text{HRESET}}$ is asserted.
11. The internal reset to the core and the rest of the logic is negated. I/O drivers are enabled. The LBC DLL begin to lock. The PCI interface can assert $\overline{\text{DEVSEL}}$ in response to configuration cycles.
12. The device stops driving $\overline{\text{SRESET}}$ and $\overline{\text{SRESET}}$ is negated. The reset to the e300 core is negated and the core is enabled. The boot sequencer, if enabled, is released, causing it to load configuration data from serial ROMs, as described in [Section 15.4.5, “Boot Sequencer Mode.”](#)

13. Before the boot sequencer finishes, it can enable the PCI interface to accept external requests, if required, by clearing the CFG_LOCK bit in the PCI function configuration register as described in [Table 13-40](#). If the e300 core is required to proceed, the boot sequencer should enable boot vector fetch by clearing ACR[COREDIS] as described in [Section 6.2.1, “Arbiter Configuration Register \(ACR\).”](#)
14. The PCI interface can now accept external requests, if enabled, and the boot vector fetch by the core can proceed, if enabled.

The device is now in its ready state.

Figure 4-1 shows a timing diagram of the power-on reset flow.

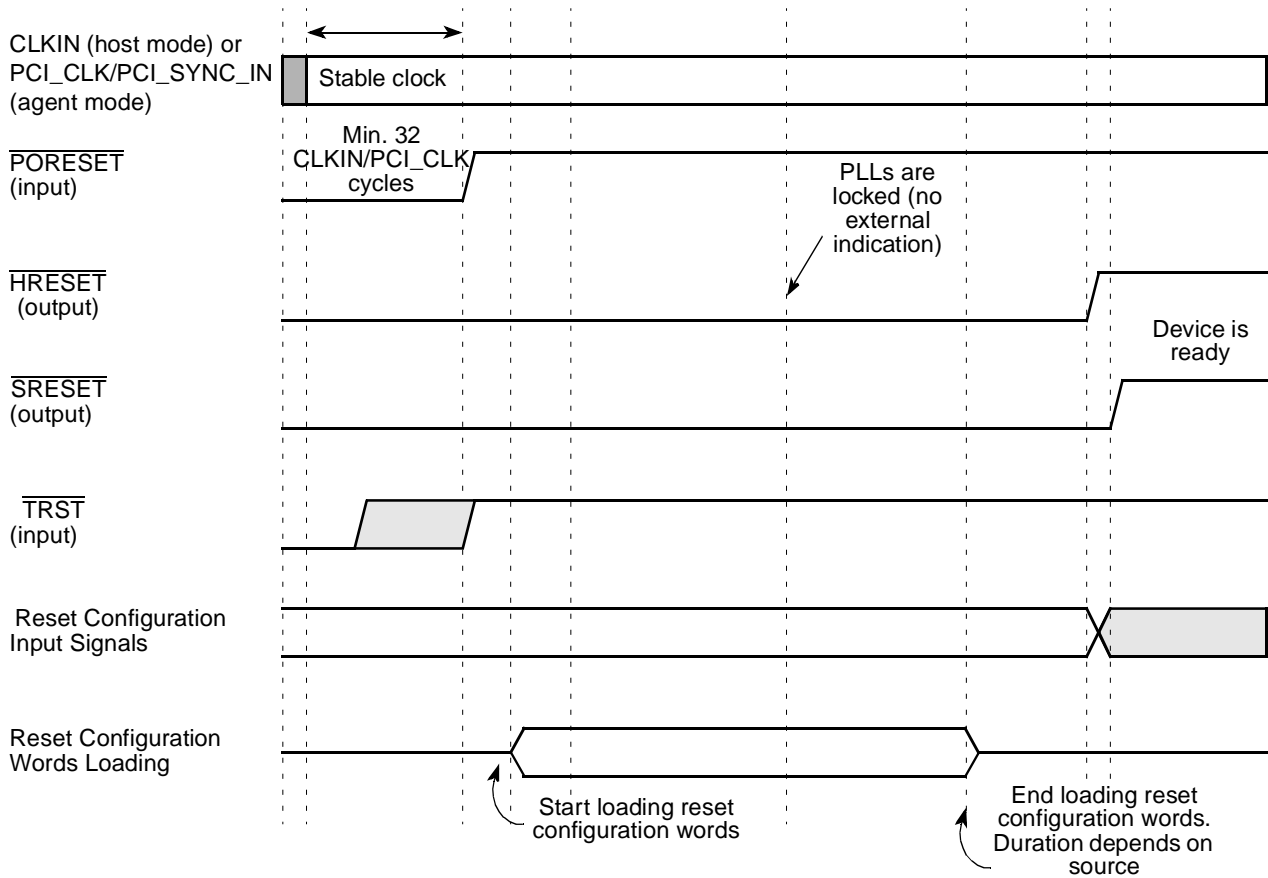


Figure 4-1. Power-On Reset Flow

4.2.3 Hard Reset Flow

The $\overline{\text{HRESET}}$ signal is initiated externally by asserting $\overline{\text{HRESET}}$ or internally when the device detects a reason to generate an internal hard reset sequence. In both cases, the device continues asserting $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ throughout the $\overline{\text{HRESET}}$ state. The hard reset sequence time varies according to the configuration source and CLKIN (PCI host mode) or PCI_CLK (PCI agent mode) frequency. The reset configuration input signals (CFG_RESET_SOURCE and CFG_CLKIN_DIV) are not sampled by hard reset (only by power-on reset), so the device immediately starts loading the reset configuration words and

configures the device as explained in Section 4.3.3, “Loading the Reset Configuration Words.” After the configuration sequence completes, the device releases both the $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ signals and exits the $\overline{\text{HRESET}}$ state. An external pull-up resistor should negate the signals. After negation is detected, a 16-cycle period is taken before testing for the presence of an external (hard/soft) reset.

NOTE

Because the device does not sample the reset configuration input signals (CFG_RESET_SOURCE, CFG_CLKIN_DIV) during a hard reset flow, setting a new value on those signals (other than that set during power-on reset) has no effect.

Figure 4-2 shows a timing diagram of the hard reset flow.

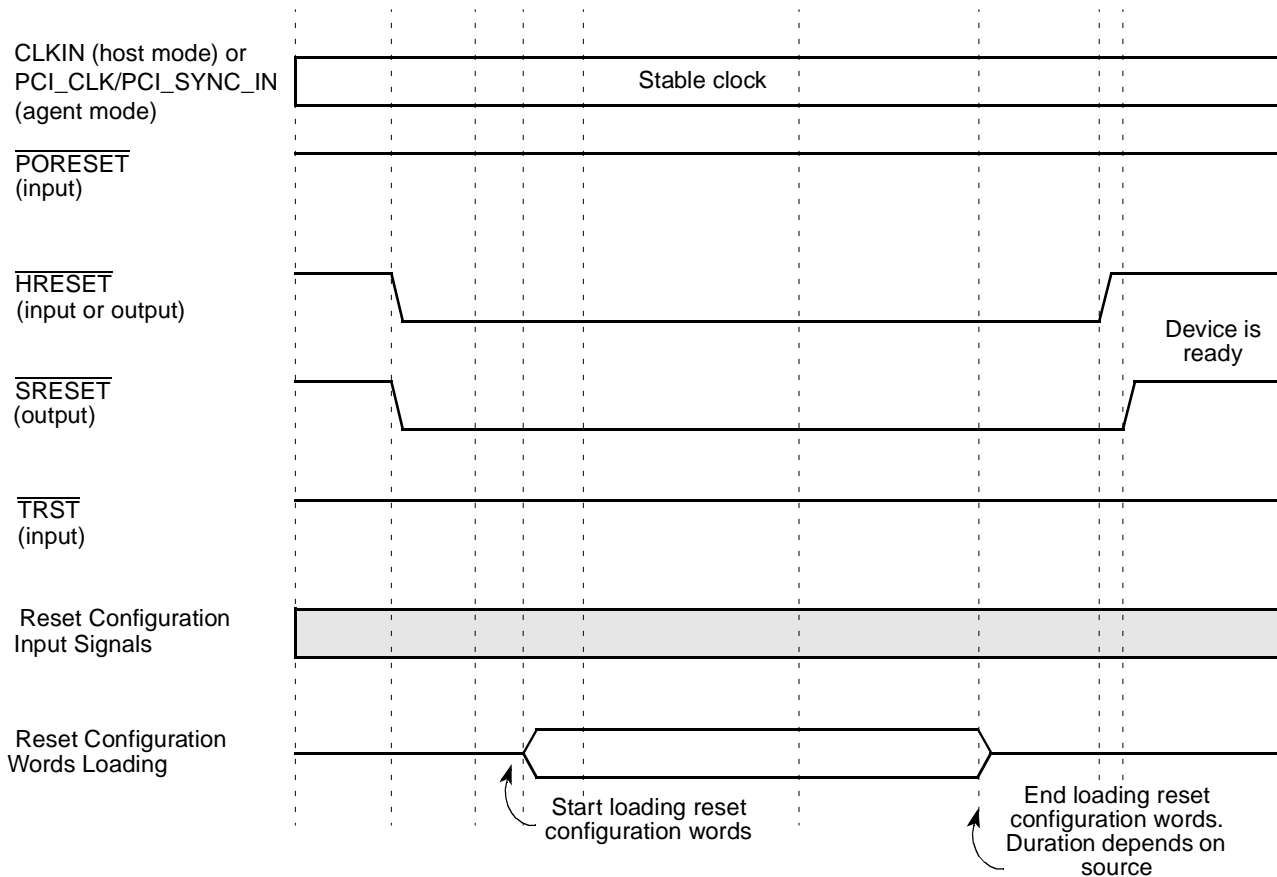


Figure 4-2. Hard Reset Flow

4.2.4 Soft Reset Flow

The $\overline{\text{SRESET}}$ signal can be initiated externally by asserting $\overline{\text{SRESET}}$ or internally when the device detects a cause to assert $\overline{\text{SRESET}}$. In both cases, the device asserts $\overline{\text{SRESET}}$ for 512 PCI_CLK/PCI_SYNC_IN/ SYNC_IN clock cycles and then releases $\overline{\text{SRESET}}$ and exits the $\overline{\text{SRESET}}$ signal. An external pull-up resistor should negate $\overline{\text{SRESET}}$; after negation is detected, a 16-cycle period is taken before testing for the

presence of an external (hard/soft) reset. While $\overline{\text{SRESET}}$ is asserted, internal hardware is reset but hard reset configuration does not change.

4.3 Reset Configuration

The device is initialized using two complementary methods, latching CFG_RESET_SOURCE and loading the reset configuration words. Initially, a few input signals are sampled during the assertion of the $\overline{\text{PORESET}}$ signal. These signals determine whether a reset configuration word is required and the device source interface from which it is loaded. According to the value on these signals, the device continues loading the reset configuration word.

4.3.1 Reset Configuration Signals

Reset configuration input signals are on device pins that have other functions when the device is not in reset state. These input signals are sampled into registers during the assertion of $\overline{\text{PORESET}}$, after a stable clock is supplied ($\overline{\text{PORESET}}$), and must be pulled high or low by external resistors as long as $\overline{\text{HRESET}}$ is asserted. While the $\overline{\text{PORESET}}$ and $\overline{\text{HRESET}}$ signals are asserted, all other signal drivers connected to these signals must be in the high-impedance state. Refer to the hardware specifications for proper resistor values for pulling reset configuration signals high or low.

This section describes the modes configured by the reset configuration signals. Note that the reset configuration inputs sampled values are accessible to software through memory-mapped registers described in [Section 4.5.1.3, “Reset Status Register \(RSR\),”](#) and [Section 4.5.2.1, “System PLL Mode Register \(SPMR\).”](#)

NOTE

Implement one of the following methods to control the selection between the reset and non-reset function of these pins.

- Resistors. Use pullup or pulldown resistors to set the desired value on the reset configuration input signals. During the power-on and hard reset sequences, these signals are inputs to the device.
- Active driving device. Use $\overline{\text{HRESET}}$ to control the driving device. When $\overline{\text{HRESET}}$ is asserted, drive reset configuration values on the pins; when $\overline{\text{HRESET}}$ is negated, stop driving the reset configuration input signals.

4.3.1.1 Reset Configuration Word Source

The reset configuration word source options, shown in [Table 4-5](#), select whether the device loads a reset configuration word from a local bus EEPROM or I²C EEPROM (I²C #1) or uses hard-coded default

options. The value of these signals also affects the duration of power-on and hard reset sequences. In any case, the reset sequence does not exceed 1 ms.

Table 4-5. Reset Configuration Words Source

CFG_RESET_SOURCE[0:2]	Meaning
000	Reset configuration word is loaded from a local bus EEPROM
001	Reset configuration word is loaded from an I ² C EEPROM. PCI_CLK/PCI_SYNC_IN is in the range of 25–44 MHz. This option will be removed from future designs. Thus, customers are recommended to use 010 option.
010	Reset configuration word is loaded from an I ² C EEPROM. PCI_CLK/PCI_SYNC_IN is valid for any PCI frequency up to 66.67 MHz (range of 25–66.67 MHz).
011	Hard-coded option 0. Reset configuration word is not loaded.
100	Hard-coded option 1. Reset configuration word is not loaded.
101	Hard-coded option 2. Reset configuration word is not loaded.
110	Hard-coded option 3. Reset configuration word is not loaded.
111	Hard-coded option 4. Reset configuration word is not loaded.

4.3.1.2 CLKIN Division

When the device is configured as a PCI host, the CFG_CLKIN_DIV configuration input selects the relationship between CLKIN and PCI_SYNC_OUT/SYNC_OUT as shown in Table 4-7. As a PCI host, the device supports three PCI_CLK output signals. The frequency of the output clocks will be equal to the PCI_SYNC_OUT frequency.

When the device is configured as a PCI agent, the CFG_CLKIN_DIV configuration input can be used to double the internal clock frequencies, if sampled as ‘1’ during power-on reset assertion. This feature is useful if a fixed internal frequency is desired regardless of whether the PCI clock is running at 33 or 66 MHz. PCI specifications require the PCI clock frequency information to be provided by the M66EN signal.

Table 4-6. Division

CFG_CLKIN_DIV	Description
0	In PCI host mode, CLKIN: PCI_SYNC_OUT = 1:1 and all PCI_CLK_OUT[0:2] clocks are running at a frequency which is equal to the CLKIN frequency.
1	In PCI agent mode, CLKIN: PCI_SYNC_OUT = 2:1 and all PCI_CLK_OUT[0:2] clocks are running at a frequency which is equal to the PCI_SYNC_OUT frequency. In PCI agent mode, internal frequency is doubled. See the <i>MPC8360E Hardware Specifications</i> for details.

4.3.1.3 Selecting Reset Configuration Input Signals

The example described in Table 4-7 shows how the user should pull down or pull up the reset configuration input signals (CFG_RESET_SOURCE, CFG_CLKIN_DIV). The reset sequence duration is measured from the negation of PORESET to the negation of SRESET. Note that the duration mentioned in this table

is typical and does not represent cases in which the process of loading the reset configuration word had to be retried due to errors.

Table 4-7. Selecting Reset Configuration Input Signals

I ² C EEPROM Configuration Words	CLKIN Frequency (Host Mode)	CFG_CLKIN_DIV (Host Mode)	PCI_CLK Frequency (Agent Mode)	CFG_RESET_SOURCE[0:2]	Reset Sequence Duration in CLKIN/PCI_CLK Cycles	Duration
No	33 MHz	0	33 MHz	000, 011–111 (not I ² C EEPROM)	15380	462 μs
No	66 MHz	0	66 MHz	000, 011–111 (not I ² C EEPROM)	15380	231 μs
No	66 MHz	1	33 MHz	000, 011–111 (not I ² C EEPROM)	30760/15380	462 μs
Yes	33 MHz	0	33 MHz	001 (I ² C EEPROM, low PCI_SYNC_IN/PCI_CLK clock frequency)	24548	736 μs
Yes	66 MHz	0	66 MHz	010 (I ² C EEPROM, high PCI_SYNC_IN/PCI_CLK clock frequency)	37908	568 μs
Yes	66 MHz	1	33 MHz	001 (I ² C EEPROM, low PCI_SYNC_IN/PCI_CLK clock frequency)	49096/24548	736 μs

4.3.2 Reset Configuration Words

The reset configuration words control the clock ratios and other basic device functions such as PCI host or agent mode, boot location, and endian mode. The reset configuration words are loaded from the local bus or from the I²C interface or from hard-coded values during the power-on or hard reset flows. See [Section 4.3.1, “Reset Configuration Signals,”](#) for information on the reset configuration word source. Although the configuration reset words are loaded during hard reset flows, the clocks and PLL modes are reset only when $\overline{\text{PORESET}}$ is asserted during a power-on reset flow. See [Section 4.2.1.2, “Reset Actions.”](#) The values of fields in the reset configuration words registers (RCWLR and RCWHR) reflect only their state during the reset flow. Some of these parameters and modes can be modified by changing their values in the memory-mapped registers of other units, which does not affect RCWLR and RCWHR.

The reset configuration settings are accessible to software through the following read-only memory-mapped registers:

- Reset configuration word low register (RCWLR)
- Reset configuration word high register (RCWHR)
- Reset status register (RSR)
- System PLL mode register (SPMR)

See [Section 4.5, “Memory Map/Register Definitions.”](#)

4.3.2.1 Reset Configuration Word Low Register (RCWLR)

RCWLR is shown in [Figure 4-3](#). This read-only register gets its values according to the reset configuration word low loaded during the reset flow.

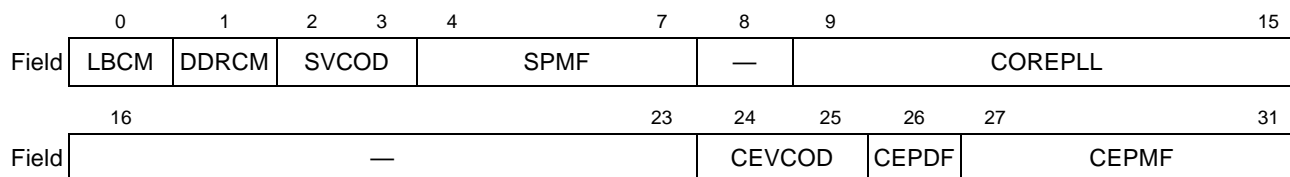


Figure 4-3. Reset Configuration Word Low Register (RCWLR)

[Table 4-8](#) defines the RCWLR bit fields.

Table 4-8. RCWLR Bit Settings

Bits	Name	Description
0	LBCM	Local bus/secondary DDR SDRAM memory controller clock mode. Selects the local bus and secondary DDR memory controller clock ratio. If this bit is set, the local bus and secondary DDR memory controllers operate at twice the frequency of the <i>csb_clk</i> . If this bit is cleared, the local bus and secondary DDR memory controllers operate at the <i>csb_clk</i> frequency. The 2:1 mode is useful when the <i>csb_clk</i> operates at low frequency. 0 <i>csb_clk</i> ratio is 1:1 1 <i>csb_clk</i> ratio is 2:1
1	DDRCM	DDR SDRAM memory controller clock mode. Selects the DDR SDRAM memory controller clock ratio. If this bit is set, the DDR SDRAM memory controller operates at twice the frequency of the <i>csb_clk</i> . If this bit is cleared, the DDR SDRAM memory controller operates at the <i>csb_clk</i> frequency. The 2:1 mode is useful mostly for a 32-bit data bus width. 0 <i>csb_clk</i> ratio is 1:1 1 <i>csb_clk</i> ratio is 2:1
2–3	SVCOD	System PLL VCO division. See Section 4.3.2.1.1, “System PLL VCO Division.”
4–7	SPMF	System PLL multiplication factor.
8	—	Reserved, should be cleared.
9–15	COREPLL	Core PLL configuration. COREPLL sets the ratio between the e300 core clock and the internal <i>csb_clk</i> of the device. The encodings for COREPLL are given in the hardware specifications for this device.
16–23	—	Reserved, should be cleared.
24–25	CEVCOD	QUICC Engine PLL VCO division. Establishes the internal ratio between the QUICC Engine PLL VCO point and the QUICC Engine PLL output. 00 4 01 8 10 2 11 Reserved, should be cleared Notes: Set CEVCOD to 01 (Division factor of 8) for QE frequency of below 100 MHz. Set CEVCOD to 00 (Division factor of 4) for QE frequency of 100–267 MHz. Set CEVCOD to 10 (Division factor of 2) for QE frequency of above 267 MHz.

Table 4-8. RCWLR Bit Settings (continued)

Bits	Name	Description
26	CEPDF	QUICC Engine PLL division factor. Selects whether the PLL output is halved. By setting this bit, non-integer ratios between the primary clock input and <i>ce_clk</i> can be achieved. 0 <i>ce_clk</i> = primary clock input × CEPMF 1 <i>ce_clk</i> = (primary clock input × CEPMF) / 2
27–31	CEPMF	QUICC Engine PLL multiplication factor. See Section 4.3.2.1.3, “QUICC Engine PLL Multiplication Factor,” for more information

4.3.2.1.1 System PLL VCO Division

The RCWLR field SVCOD (system PLL VCO division), shown in [Table 4-9](#), establishes the internal ratio between the system PLL VCO frequency and the PLL output clock frequency. The PLL output clock frequency equals *csb_clk* frequency if RCWLR[LBCM] and RCWLR[DDRCM] are both cleared or twice the *csb_clk* frequency if RCWLR[LBCM] or RCWLR[DDRCM] or both of them are set.

The RCWLR[SVCOD] value should be set such that the PLL VCO frequency will be in the range of $500 \text{ MHz} \leq \text{system PLL VCO frequency} \leq 1100 \text{ MHz}$.

[Table 4-9](#) describes the setting of SVCOD bits.

Table 4-9. System PLL VCO Division

Reset Configuration Word Low Register (RCWLR) Bits	Field Name	Value (Binary)	VCO Division Factor
2–3	SVCOD	00	4
		01	8
		10	2
		11	Reserved

4.3.2.1.2 System PLL Configuration

The system PLL ratio reset, shown in [Table 4-10](#), establishes the clock ratio between the CLKIN (PCI host mode) or PCI_CLK (PCI agent mode) input signal and the internal *csb_clk* of the device. *csb_clk* drives internal units and feeds the e300 core PLL.

Table 4-10. System PLL Ratio

RCWLR Bits	Field Name	Value (Binary)	<i>csb_clk</i> : CLKIN (PCI Host Mode) <i>csb_clk</i> : (PCI_CLK x (1+~sampled_cfg_clkin_div)) (PCI Agent Mode)
4-7	SPMF	0000	16 : 1
		0001	Reserved
		0010	2 : 1
		0011	3 : 1
		0100	4 : 1
		0101	5 : 1
		0110	6 : 1
		0111	7 : 1
		1000	8 : 1
		1001	9 : 1
		1010	10 : 1
		1011	11 : 1
		1100	12 : 1
		1101	13 : 1
		1110	14 : 1
1111	15 : 1		

NOTE

In PCI host mode, the SPMF field described in [Table 4-10](#) always selects the *csb_clk*:CLKIN ratio regardless of the CFG_CLKIN_DIV reset configuration input value during reset flow.

The SPMF field maximum allowed value is dependent on the value sampled on CFG_CLKIN_DIV during power-on reset, and the LBCM and DDRCM reset configuration word fields values. [Table 4-11](#) defines the upper limit of SPMF with respect to these values. Values for SPMF are as follows:

Table 4-11. SPMF Maximum Values

CFG_CLKIN_DIV	LBCM	DDRCM	Maximum SPMF Value (decimal)
0	0	0	16
0	0	1	8
0	1	0	8
0	1	1	8

Table 4-11. SPMF Maximum Values (continued)

CFG_CLKIN_DIV	LBCM	DDRDM	Maximum SPMF Value (decimal)
1	0	0	8
1	0	1	4
1	1	0	4
1	1	1	4

4.3.2.1.3 QUICC Engine PLL Multiplication Factor

The RCWLR field CEPMF (QUICC Engine PLL multiplication factor), shown in [Table 4-12](#), along with the QUICC Engine PLL division factor (CEPDF,) establishes the ratio between *ce_clk* and the primary input clock.

Table 4-12. QUICC Engine PLL Multiplication Factor

RCWLR Bits	Field Name	Value (Binary)	QUICC Engine Block PLL Multiplication Factor
27–31	CEPMF	00000	Reserved,
		00001	Reserved
		00010	2
		00011	3
		00100	4
		00101	5
		00110	6
		00111	7
		01000	8
		01001	9
		01010	10
		01011	11
		01100	12
		01101	13
		01110	14
		01111	15
		10000	16
		10001	17
		10010	18
		10011	19
		10100	20
		10101	21
		10110	22
		10111	23
		11000	24
		11001	25
		11010	26
		11011	27
		11100	28
		11101	29
		11110	30
11111	31		

4.3.2.2 Reset Configuration Word High Register (RCWHR)

RCWHR is shown in Figure 4-4. This read-only register gets its values according to the reset configuration word high loaded during the reset flow.

Offset 0x0_0904

Access: Read/Write

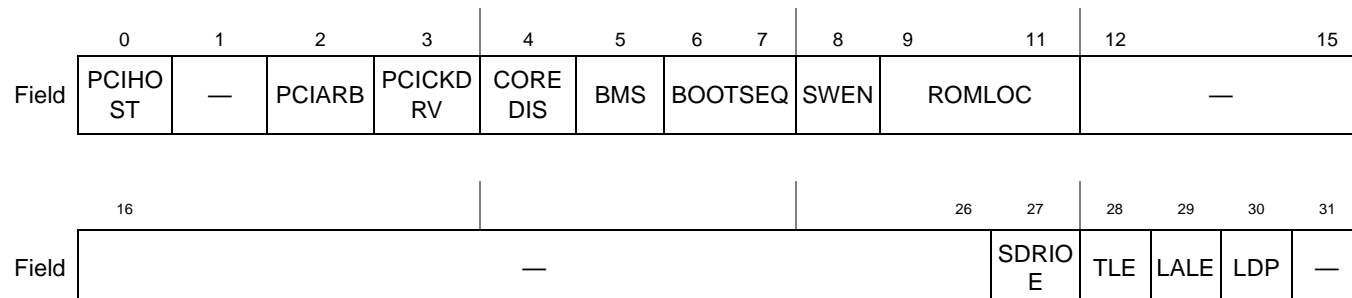


Figure 4-4. Reset Configuration Word High Register (RCWHR)

Table 4-13 defines the reset configuration word high bit fields.

Table 4-13. Reset Configuration Word High Bit Settings

Bits	Name	Description								
0	PCIHOST	PCI host mode. See Section 4.3.2.2.1, “PCI Host/Agent Configuration,” for more information.								
1	—	Reserved, should be cleared.								
2	PCIARB	PCI internal arbiter mode. Enables the on-chip PCI arbiter. 0 On-chip PCI arbiter is disabled. External arbitration is required. 1 On-chip PCI arbiter is enabled. <table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse; text-align: center;"> <thead> <tr> <th style="width: 50%;">Pin Function When PCIARB = 0</th> <th style="width: 50%;">Pin Function When PCIARB = 1</th> </tr> </thead> <tbody> <tr> <td>CPCI_HS_ES</td> <td>$\overline{\text{PCI_REQ}}[1]$</td> </tr> <tr> <td>CPCI_HS_LED</td> <td>$\overline{\text{PCI_GNT}}[1]$</td> </tr> <tr> <td>CPCI_HS_ENUM</td> <td>$\overline{\text{PCI_GNT}}[2]$</td> </tr> </tbody> </table>	Pin Function When PCIARB = 0	Pin Function When PCIARB = 1	CPCI_HS_ES	$\overline{\text{PCI_REQ}}[1]$	CPCI_HS_LED	$\overline{\text{PCI_GNT}}[1]$	CPCI_HS_ENUM	$\overline{\text{PCI_GNT}}[2]$
Pin Function When PCIARB = 0	Pin Function When PCIARB = 1									
CPCI_HS_ES	$\overline{\text{PCI_REQ}}[1]$									
CPCI_HS_LED	$\overline{\text{PCI_GNT}}[1]$									
CPCI_HS_ENUM	$\overline{\text{PCI_GNT}}[2]$									
3	PCICKDRV	PCI clock output drive. Enables the output buffers of the PCI clock output signals (PCI_CLK_OUT[0:2]) when the device is in host mode. Disabling the PCI clock output buffers allows for external PCI clock distribution and alternate QUICC Engine functionality on the PCI_CLK_OUT[0:2] signals. In host mode when PCICKDRV = 0, the PCI_SYNC_IN signal becomes the primary clock input for the device. 0 Buffers are disabled. 1 Buffers are enabled.								

Table 4-13. Reset Configuration Word High Bit Settings (continued)

Bits	Name	Description
4	COREDISE	Core disable mode. Specifies the e300 core mode out of reset. If COREDIS is set, the core cannot fetch boot code until it is configured by an external master. The external master frees the core to boot by clearing the COREDIS bit in the arbiter configuration register as described in Section 6.2.1, "Arbiter Configuration Register (ACR)." This bit must be set when the boot sequencer is enabled to initiate the device (BOOTSEQ is not 0b00). Otherwise, unpredictable behavior occurs. 0 The core can boot without waiting for configuration by an external master. 1 Core boot holdoff mode. The core is prevented from booting until it is configured by an external master.
5	BMS	Boot memory space. See Section 4.3.2.2.2, "Boot Memory Space (BMS)," for more information.
6–7	BOOTSEQ	Boot sequencer configuration. See Section 4.3.2.2.3, "Boot Sequencer Configuration," for more information.
8	SWEN	Software watchdog enable. Selects whether the software watchdog is enabled to start counting down immediately when coming out of reset. The user can override this value by writing to the system watchdog control register (SWCRR[SWEN]) during system initialization. 0 Disabled 1 Enabled
9–11	ROMLOC	Boot ROM interface location. See Section 4.3.2.2.4, "Boot ROM Location," for more information.
12–15	—	Reserved, should be cleared.
16–17	—	Reserved, should be cleared.
18–19	—	Reserved, should be cleared.
20–26	—	Reserved, should be cleared.
27	SDDRIOE	Secondary DDR IO enable. See Section 4.3.2.2.5, "Secondary DDR IO Enable," for more information.
28	TLE	True little-endian. See Section 4.3.2.2.6, "e300 Core True Little-Endian," for more information.
29	LALE	Local bus LALE signal timing. See Section 4.3.2.2.7, "LALE Configuration," for more information.
30	LDP	LDP pin mux state after reset. See Section 4.3.2.2.8, "LDP Configuration," for more information.
31	—	Reserved, should be cleared.

4.3.2.2.1 PCI Host/Agent Configuration

The PCIHOST configuration parameter, shown in [Table 4-14](#), configures the device to act as a PCI host or as a PCI agent device. In host mode, the device can immediately master transactions to the PCI interface. If the device is a PCI agent device, the device is disabled from mastering PCI transactions until the external host enables it to do so. The external host does this by setting the control registers of the device's interfaces

appropriately. See details in the PCI programming model described in [Section 13.3, “PCI Memory Map/Register Definitions.”](#)

Table 4-14. PCI Host/Agent Configuration

RCWHR Bit	Field Name	Value (Binary)	Meaning
0	PCIHOST	0	The device acts as a PCI agent device.
		1	The device acts as the host processor (default).

NOTE

If the device is a PCI agent, and the e300 core is not in holdoff mode (as described in [Section 4.3.2.2, “Reset Configuration Word High Register \(RCWHR\)”](#)), the boot ROM should not be located on the PCI interface because the device is not enabled to master reads onto the PCI bus.

4.3.2.2.2 Boot Memory Space (BMS)

BMS defines the initial value of the e300 core MSR[IP] bit, which specifies the location of the interrupt vectors (including the hard reset exception vector). The device defines the default boot ROM memory space to be 8 Mbytes at addresses All zeros to 0x007F_FFFF or 0xFF80_0000 to 0xFFFF_FFFF. When the core comes out of reset, if it is enabled to boot, it begins fetching boot code from one of two addresses: 0x0000_0100 or 0xFFFF0_0100, and exceptions are vectored to physical addresses 0x000n_nnnn or 0xFFFFn_nnnn appropriately. This bit specifies whether an interrupt vector offset is prepended with 0xFFF or 0x000. In the description below, *n_nnnn* is the offset of the exception vector.

The boot memory space reset configuration word field, shown in [Table 4-15](#), specifies both the device boot ROM address window and the initial e300 core boot address.

Table 4-15. Boot Memory Space

RCWHR Bit	Field Name	Value (Binary)	Meaning
5	BMS	0	Boot memory space is 8 Mbytes at All zeros to 0x007F_FFFF. e300 core register MSR[IP] initial value is 0b0. The core, if enabled to boot, begins fetching boot code from address 0x0000_0100 and exceptions are vectored to the physical address of 0x000n_nnnn.
		1	Boot memory space is 8 Mbytes at 0xFF80_0000 to 0xFFFF_FFFF. e300 core register MSR[IP] initial value is 0b1. The core, if enabled to boot, begins fetching boot code from address 0xFFFF0_0100 and exceptions are vectored to the physical address of 0xFFFFn_nnnn.

4.3.2.2.3 Boot Sequencer Configuration

The boot sequencer configuration options, shown in [Table 4-16](#), allow the boot sequencer to load configuration data from the serial ROM located on the I²C port before the host tries to configure the device.

These options also specify normal or extended I²C addressing modes. See [Section 15.4.5, “Boot Sequencer Mode.”](#)

Table 4-16. Boot Sequencer Configuration

RCWHR Bits	Field Name	Value (Binary)	Meaning
6–7	BOOTSEQ	00	Boot sequencer is disabled. No I ² C ROM is accessed.
		01	Normal I ² C addressing mode is used. Boot sequencer is enabled and loads configuration information from a ROM on the I ² C interface. A valid ROM must be present.
		10	Extended I ² C addressing mode is used. Boot sequencer is enabled and loads configuration information from a ROM on the I ² C interface. A valid ROM must be present.
		11	Reserved, should be cleared.

NOTE

When the boot sequencer is enabled, the e300 core must be prevented from fetching boot code, by setting the core disable reset configuration word field (COREDIS) as described in [Section 4.3.2.2, “Reset Configuration Word High Register \(RCWHR\).”](#) If the e300 core is required to proceed, the boot sequencer should enable boot vector fetch by clearing ACR[COREDIS] as described in [Section 6.2.1, “Arbiter Configuration Register \(ACR\).”](#)

4.3.2.2.4 Boot ROM Location

The device defines the default boot ROM address range to be 8 Mbytes at addresses All zeros to 0x007F_FFFF or 0xFF80_0000 to 0xFFFF_FFFF (selected by the BMS reset configuration word field). However, the on-chip peripheral that manages these boot ROM accesses can be selected at power up.

The boot ROM location reset configuration word field, shown in [Table 4-17](#), establishes the location of boot ROM. Accesses to the boot vector and the default boot ROM region of the local address map are directed to the interface specified by this field.

Table 4-17. Boot ROM Location

Reset Configuration Word High Register (RCWHR) Bits	Field Name	Value (Binary)	Meaning
9–11	ROMLOC	000	DDR SDRAM
		001	PCI
		010	Reserved, should be cleared.
		011	Reserved, should be cleared.
		100	Reserved
		101	Local bus GPCM—8-bit ROM
		110	Local bus GPCM—16-bit ROM
		111	Local Bus GPCM—32-bit ROM

The local access window of the selected boot ROM interface is enabled and initialized with the proper base address and size, as described in [Section 5.2, “Local Memory Map Overview and Example.”](#)

NOTE

In PCI host mode, although selecting the PCI option with ROMLOC sets the appropriate local access window, PCI_RESET_OUT remains asserted and PCI_CLK_OUT[x] are disabled after reset.

In this case the e300 core must be prevented from fetching boot code, by setting the core disable reset configuration word field (COREDIS) as described in [Section 4.3.2.2, “Reset Configuration Word High Register \(RCWHR\).”](#) The boot sequencer should write to the appropriate registers to negate PCI_RESET_OUT and enable the appropriate clocks to the PCI ROM device. Only then should it enable the boot vector fetch by clearing ACR[COREDIS] as described in [Section 6.2.1, “Arbiter Configuration Register \(ACR\).”](#)

4.3.2.2.5 Secondary DDR IO Enable

The SDDRIOE bit reset configuration word field, shown in [Table 4-18](#), enables the Secondary DDR memory controller I/O pins.

Table 4-18. Secondary DDR Configuration

Reset Configuration Word High Register (RCWHR) Bit	Field Name	Value (Binary)	Meaning
27	SDDRIOE	0	Secondary DDR memory controller I/O is disabled. All DDR I/O pins of the device are assigned to the system DDR memory controller which can supports 64-bit or 32-bit data bus width in this mode.
		1	Secondary DDR memory controller I/O is enabled. Both System DDR and Secondary DDR memory controllers are assigned part of the DDR I/O pins of the device. Both controllers can support 32-bit data bus width in this mode.

4.3.2.2.6 e300 Core True Little-Endian

The true little endian reset configuration word field, shown in [Table 4-19](#), selects whether the e300 core operates in big-endian mode or true little-endian mode at reset.

Table 4-19. e300 Core True Little-Endian

Reset Configuration Word High Register (RCWHR) Bit	Field Name	Value (Binary)	Meaning
28	TLE	0	Big-endian mode
		1	True little-endian mode

4.3.2.2.7 LALE Configuration

The LALE reset configuration word field, shown in [Table 4-20](#), configures the timing of the local bus LALE signal. Refer to the hardware specifications for specific timing information.

Table 4-20. LALE Configuration

Reset Configuration Word High Register (RCWHR) Bit	Field Name	Value (Binary)	Meaning
29	LALE	0	Normal LALE timing
		1	LALE is negated 1/2 a LBC_controller_clk earlier

4.3.2.2.8 LDP Configuration

The LDP reset configuration word field configures the initial state of SICRL[LDP_A], which controls the functionality of the LPD[0:3] pins. [Table 4-21](#) shows the LDP configuration.

Table 4-21. LDP Configuration

Reset Configuration Word High Register (RCWHR) Bit	Field Name	Meaning
30	LDP	0 Initial value of SICRL[LDP_A] is 1, meaning that LDP0–LDP3 are used for local data parity. 1 Initial value of SICRL[LDP_A] is 0; the meaning is as follows: <ul style="list-style-type: none"> • LDP0 is used as $\overline{\text{CKSTOP_OUT}}$ • LDP1 is used as $\overline{\text{CKSTOP_IN}}$. • LDP2 is used as LCS6 • LDP3 is used as LCS7

4.3.3 Loading the Reset Configuration Words

The device loads the reset configuration words from a local bus EEPROM or an I²C serial EEPROM, or uses hard-coded configuration, as selected by the reset configuration inputs described in [Section 4.3.1](#), “Reset Configuration Signals.” The following sections describe each of these options.

4.3.3.1 Loading from Local Bus EEPROM

The reset configuration words are assumed to reside in an EEPROM device connected to $\overline{\text{LCS0}}$ of the device local bus. Because the port size of this EEPROM is unknown, the device reads all configuration words byte-by-byte only from locations that are independent of port size. +LCS0 is the default for GPCM, so GPCM controlled is used to read the reset configuration word from EEPROM. /LGTA should be high to avoid unintended early termination of the read cycle.

[Table 4-22](#) shows addresses that should be used to contain the reset configuration words. Byte addresses that do not appear in this table have no effect on the configuration of the device. The values of the bytes in [Table 4-22](#) are always read on byte lane LAD[0:7] regardless of the port size.

Table 4-22. Local Bus Configuration EEPROM Addresses

Reset Configuration Word	Bits [0:7] Address	Bits [8:15] Address	Bits [16:23] Address	Bits [24:31] Address
Low	0x00	0x08	0x10	0x18
High	0x20	0x28	0x30	0x38

The device first reads a value from address 0x00 then reads a value from addresses 0x08, 0x10, and 0x18. These four bytes form the reset configuration word low. It then proceeds reading the bytes from addresses 0x20, 0x28, 0x30, and 0x38, which form the reset configuration word high.

Table 4-23 shows the data structure of the local bus device containing the reset configuration words (RCWL and RCWH).

Table 4-23. Local Bus Reset Configuration Words Data Structure

EEPROM Address	EEPROM Data Bits			
	[0:7]	[8:15]	[16:23]	[24:31]
0x00	RCWL[0:7]			
0x04				
0x08	RCWL[8:15]			
0x0C				
0x10	RCWL[16:23]			
0x14				
0x18	RCWL[24:31]			
0x1C				
0x20	RCWH[0:7]			
0x24				
0x28	RCWH[8:15]			
0x2C				
0x30	RCWH[16:23]			
0x34				
0x38	RCWH[24:31]			
0x3C				

For loading the reset configuration word from a local bus EEPROM, the PCI_SYNC_IN/PCI_CLK input clock is divided by 32 to enable operation of slow frequency memories. Figure 4-5 and Figure 4-6 show the timing of EEPROM operation.

As the figures indicate, if the HRCW is loaded through the local bus, the LA[27:31] pins are used and not the LAD[27:31] pins. The LAD[27:31] pins are not driven during HRCW loading. Note that in Figure 4-5 and Figure 4-6, only the LA[27:31] are shown incrementing during the load of the HRCW. In essence, the device does a type of burst access to the flash memory when it loads the HRCW. It drives the high-order bits of the address on LAD[0:26], which is also the first address in a 4-byte sequence, asserts LALE, latches the first byte, and then increments LA[27:31] to get the next 3 bytes. It then drives the high-order bits for the second access, asserts LALE, latches a byte, and again increments the LA[27:31] to get the next 3 bytes. Out of reset, the LA[27:31] and LAD[27:31] mirror each other (while LALE is asserted). Note that $\overline{\text{LCS0}}$ is asserted low during the assertion of $\overline{\text{PORESET}}$ and $\overline{\text{HRESET}}$. Also, $\overline{\text{PORESET}}$ negation to LALE assertion is 36 cycles of PCI_SYNC_IN/PCI_CLK clock signal.

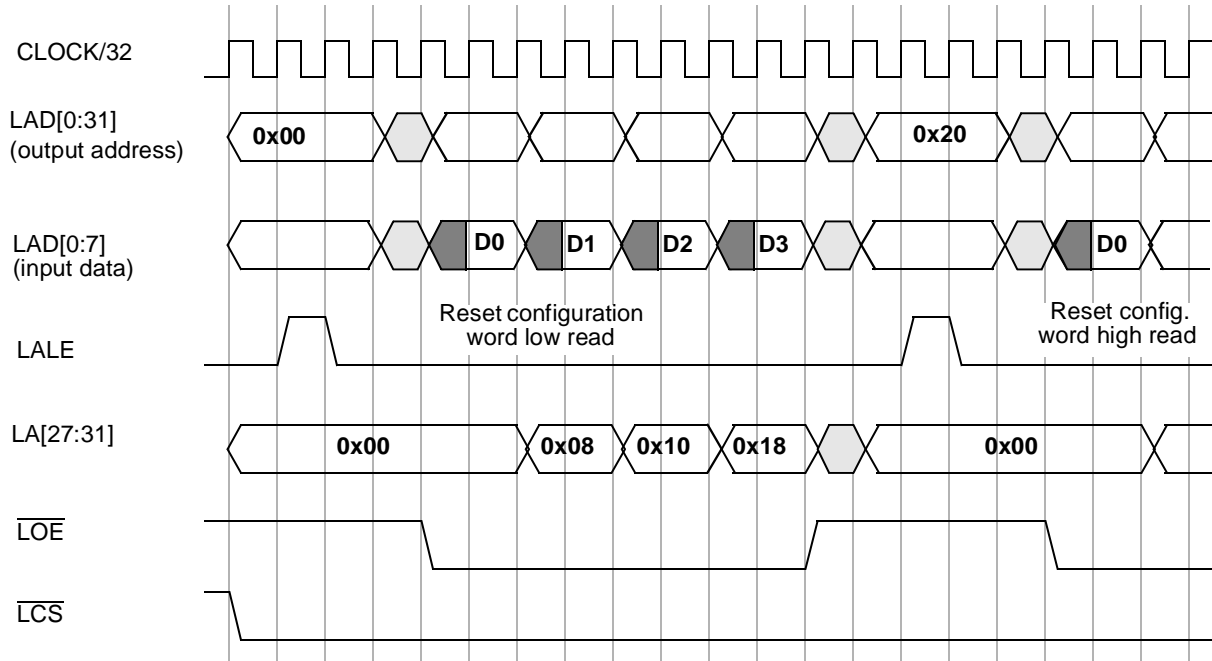


Figure 4-5. Loading Reset Configuration Words from Local Bus

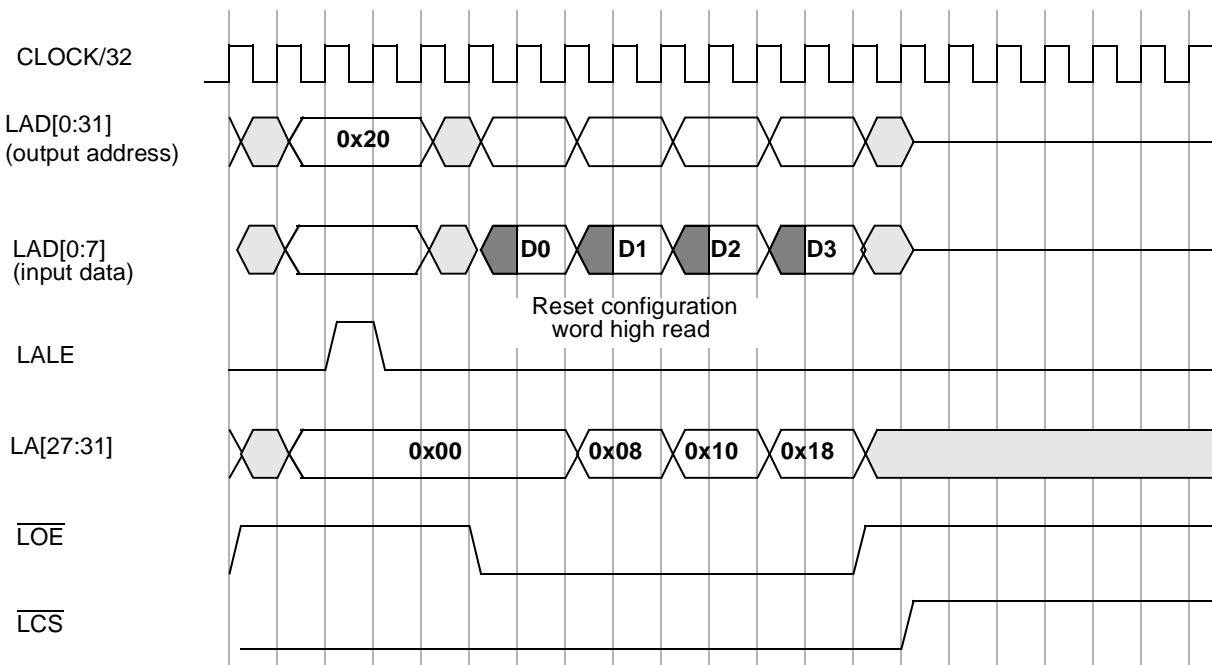


Figure 4-6. Loading Reset Configuration Words from Local Bus (continued)

4.3.3.2 Loading from I²C EEPROM

The device is capable of loading the reset configuration word from the I²C interfaces (the device has two I²C interfaces, but only I²C #1 can be used for this purpose). If the device is configured to load the reset configuration word from the I²C interface, according to the reset configuration input signals, it uses the I²C unit boot sequencer in a special mode. In this mode, the I²C boot sequencer is activated while the rest of the device is still in reset state ($\overline{\text{HRESET}}$ asserted) to load the reset configuration words from an I²C serial EEPROM.

Note that this does not prevent using the I²C boot sequencer to initiate the device in the normal functional mode after reset state has completed. The only restriction is that the first two EEPROM data structures contain dedicated reset information.

4.3.3.2.1 Using the Boot Sequencer Reset Configuration

For a detailed description about the I²C interface and the boot sequencer refer to [Section 15.4.5, “Boot Sequencer Mode.”](#)

NOTE

When reset configuration words are loaded from an I²C EEPROM, an I²C serial EEPROM of extended addressing type must be used.

If the I²C interface is used for loading the reset configuration words, the I²C module addresses the EEPROM and reads the first two data structures (after reading the preamble). Upon being read, the reset configuration words are latched inside the device and the I²C module enters its reset state until $\overline{\text{HRESET}}$ is negated. There should be no other I²C traffic when the boot sequencer is active.

After $\overline{\text{HRESET}}$ is negated, the functional boot sequencer, in extended I²C addressing mode, may be activated if the BOOTSEQ field of the reset configuration word high is set to 0b10.

4.3.3.2.2 EEPROM Calling Address

The device uses 0b101_0000 for the EEPROM calling address. The EEPROM to be addressed must contain the reset configuration information and be programmed to respond to this address. No additional EEPROMs are accessed by the boot sequencer in reset configuration mode.

4.3.3.2.3 EEPROM Data Format in Reset Configuration Mode

The I²C module expects that a particular data format be used for data in the EEPROM. A preamble should be the first 3 bytes programmed into the EEPROM. It should have a value of 0xAA55AA. The I²C module checks to ensure that this preamble is correctly detected before proceeding further. Following the preamble, there should be the two reset configuration words, programmed according to a particular format, as shown in [Figure 4-7](#).

The first 3 bytes hold the attributes and address offset. The addresses of the two reset configuration words must be programmed to the offset of the reset configuration word low register (RCWLR) and reset configuration word high register (RCWHR) respectively (see [Section 4.5.1.1, “Reset Configuration Word Low Register \(RCWLR\),”](#) and [Section 4.5.1.2, “Reset Configuration Word High Register \(RCWHR\)”](#)).

The attributes should be programmed as follows: alternate configuration space (ACS) should be cleared (0b0), byte enables should be all ones, and continue (CONT) should be set.

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.

IMMRBAR value is prepended to the EEPROM address to generate the complete memory-mapped register's address.

When the I²C operates in reset configuration mode, the cyclic redundancy check (CRC) is ignored, as well as any registers following the first two reset configuration words.

0	1	4	5	6	7
ACS (0)	BYTE_EN (1111)			CONT (1)	RCWLR ADDR[12–13]
RCWLR ADDR[14:21]					
RCWLR ADDR[22:29]					
Reset configuration word low [0–7]					
Reset configuration word low [8–15]					
Reset configuration word low [16–23]					
Reset configuration word low [24–31]					
ACS (0)	BYTE_EN (1111)			CONT (1)	RCWHR ADDR[12–13]
RCWHR ADDR[14–21]					
RCWHR ADDR[22–29]					
Reset configuration word high [0–7]					
Reset configuration word high [8–15]					
Reset configuration word high [16–23]					
Reset configuration word high [24–31]					

Figure 4-7. EEPROM Data Format for Reset Configuration Words Preload Command

Figure 4-8 shows an example of the EEPROM contents, including the preamble, reset configuration words and additional initialization data, and CRC. In this example, it is assumed that the EEPROM contains

information additional to the reset configuration words, which should be loaded in the functional state after the device completes its reset flow.

0	1	2	3	4	5	6	7	Preamble	
1	0	1	0	1	0	1	0		
0	1	0	1	0	1	0	1		
1	0	1	0	1	0	1	0		
0	1	1	1	1	1	RCWLR ADDR[12:13]		Reset configuration word low preload command	
RCWLR ADDR[14-21]									
RCWLR ADDR[22-29]									
Reset configuration word low [0-7]									
Reset configuration word low [8-15]									
Reset configuration word low [16-23]									
Reset configuration word low [24-31]									
0	1	1	1	1	1	RCWHR ADDR[12:13]		Reset configuration word high preload command	
RCWHR ADDR[14-21]									
RCWHR ADDR[22-29]									
Reset configuration word high [0-7]									
Reset configuration word high [8-15]									
Reset configuration word high [16-23]									
Reset configuration word high [24-31]									
*									
ACS	BYTE_EN				1	ADDR[12-13]			Last configuration preload command
ADDR[14-21]									
ADDR[22-29]									
DATA[0-7]									
DATA[8-15]									
DATA[16-23]									
DATA[24-31]									
0	0	0	0	0	0	0	0	End command	
0	0	0	0	0	0	0	0		
0	0	0	0	0	0	0	0		
CRC[0-7]							Cyclic redundancy check		
CRC[8-15]									
CRC[16-23]									
CRC[24-31]									

Figure 4-8. EEPROM Contents

4.3.3.2.4 Reset Configuration Load Fail

Failure of reset configuration load by the I²C boot sequencer can be caused by an incorrect EEPROM data structure or I²C bus problem. If a reset configuration load failure occurs, due to preamble fail or any other I²C bus error detection, the device will continuously attempt to reload the hard reset configuration words from the I²C bus. The device does not negate $\overline{\text{HRESET}}$ and remains in hard reset state until the HRCWs are successfully loaded or the PORESET flow is restarted.

4.3.3.3 Default Reset Configuration Words

If the device is configured not to load the reset configuration words from a local bus EEPROM or I²C EEPROM, it can also be initialized with one of five hard-coded default options, selected by the reset configuration input signals, CFG_RESET_SOURCE[0:2]. In this mode, the device is assumed to be a PCI agent, and therefore only clock modes differ among the four options.

The reset configuration words are driven internally with the values shown in [Table 4-24](#) and [Table 4-25](#).

NOTE

In this mode the device is also configured to accept PCI configuration cycles when completing its reset sequence (In PCI function configuration register, the CFG_LOCK bit is cleared). In addition, the inbound window size of the PCI inbound window attribute registers (PIWAR_n[IWS]) is set to 0b010100, defining 2-Mbyte ($2^{(20+1)}$) memory windows. See [Section 13.3.3.24, “PCI Function Configuration Register.”](#)

Table 4-24. Hard-Coded Reset Configuration Word Low Fields Values

Bits	Name	CFG_RESET_SOURCE Value					Meaning
		011	100	101	110	111	
0	LBCM	0	0	1	1	0	LBC controller clock: <i>csb_clk</i> 0 1:1 1 2:1
1	DDRCM	0	0	1	1	0	DDR controller clock: <i>csb_clk</i> 0 1:1 1 2:1
2–3	Res	10	10	10	10	10	—
4–7	SPMF	0100	1000	0100	0101	0100	<i>csb_clk</i> : PCI_CLK ratio SPMF:1
8	Res	0	0	0	0	0	—
9–15	COREPLL	0100011	0100011	0100100	0100100	0000100	Core clock: <i>csb_clk</i> ratio
16–31	CEPMF	0x0006	0x000C	0x0008	0x000A	0x0006	QUICC Engine Clock: PCI_CLK ratio CEPMF:1—

Table 4-25 defines the hard-coded reset configuration word high fields values. These values select hard-coded reset configuration words options, as described in Section 4.3.1.1, “Reset Configuration Word Source.”

Table 4-25. Hard-Coded Reset Configuration Word High Field Values

Bits	Name	CFG_RESET_SOURCE[0:2] = 011–111	Meaning
0	PCIHOST	0	PCI agent mode
1	Reserved	0	—
2	PCIARB	0	External arbiter is used
3	PCICKDRV	0	PCI_CLK_OUT[0:2] signals are not driven (I/O buffer disabled).
4	COREDIS	1	e300 core is disabled (boot holdoff)
5	BMS	1	Boot memory space is 0xFF80_0000–0xFFFF_FFFF. MSR[IP] initial value is 0b1
6–7	BOOTSEQ	00	Boot sequencer is disabled.
8	SWEN	0	Software watchdog disabled.
9–11	ROMLOC	000	Boot ROM interface location.
12–15	Reserved	0000	—
16–19	Reserved	0000	—
20–26	Reserved	0000_0000	—
27	SDDRIOE	0	Secondary DDR IO enable
28	TLE	0	Big endian mode
29	LALE	0	Normal timing
30	LDP	0	LPD pins used for local data parity
31	Reserved	0	—

4.3.3.3.1 Examples for Hard-Coded Reset Configuration Words Usage

Examples for various clock modes are listed in Table 4-26.

Table 4-26. Examples For Hard-Coded Reset Configuration Words Usage

CFG_RESET_SOURCE[0:2]	011	100	101	110	111
PCI_CLK (MHz)	66	33	33	33	66
<i>csb_clk</i> (MHz)	266	266	133	166	266
Core Clock (MHz)	400	400	266	333	533
QUICC Engine Clock (MHz)	400	400	266	333	400

4.4 Clocking

Figure 4-9 shows the internal distribution of clocks within the device.

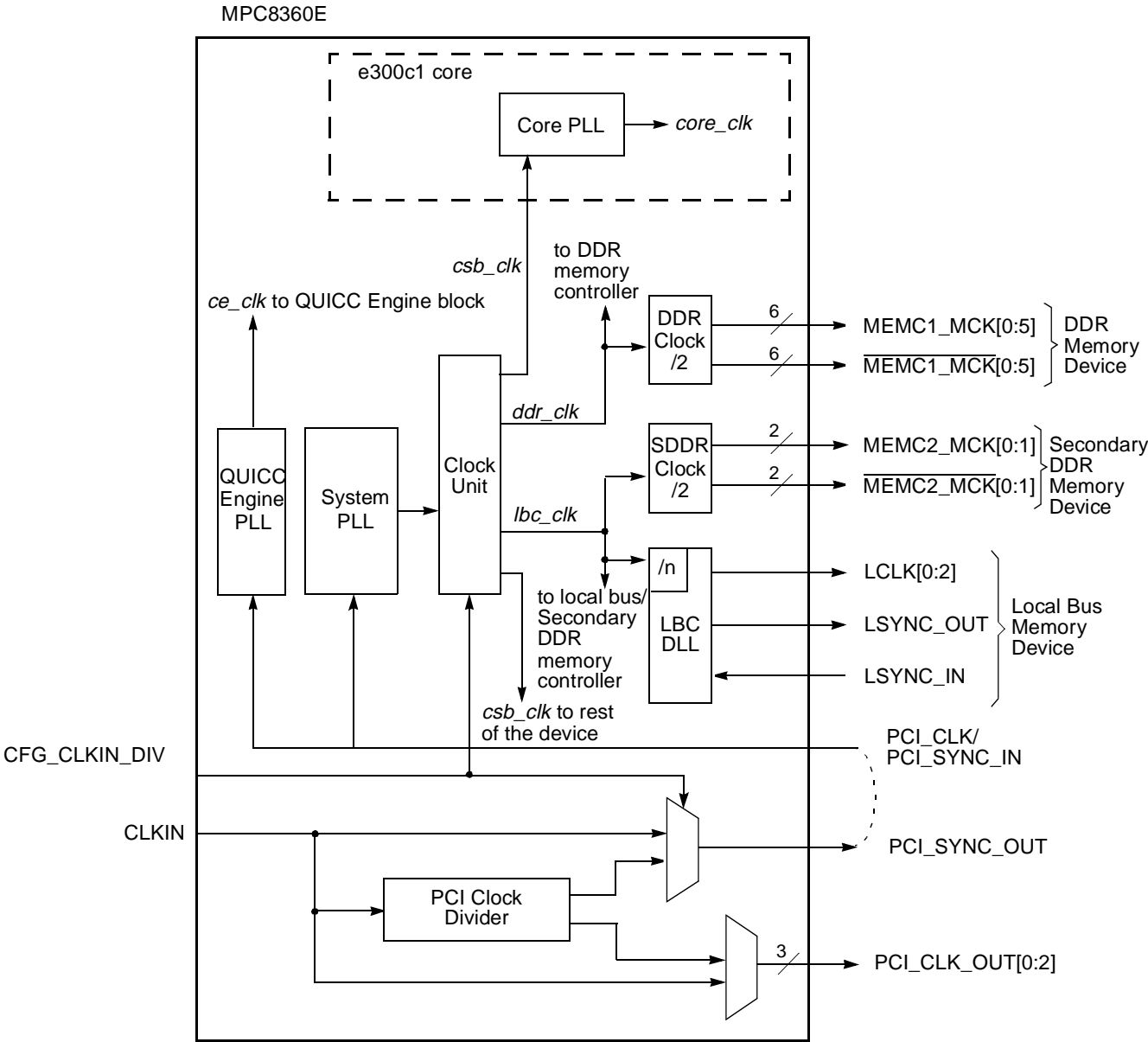


Figure 4-9. Clock Subsystem Block Diagram

The primary clock source for the device can be one of two inputs, CLKIN or PCI_CLK, depending on whether the device is configured in PCI host or PCI agent mode, respectively. Note that in PCI host mode, the primary clock input also depends on whether PCI clock output buffers are enabled with RCWH[PCICKDRV].

4.4.1 Clocking in PCI Host Mode

When the MPC8360 is configured as a PCI host device ($RCWH[PCIHOST] = 1$) and PCI clock output buffers are enabled ($RCWH[PCICKDRV] = 1$), CLKIN is the primary input clock. CLKIN feeds the PCI clock divider ($\div 2$) and the PCI_SYNC_OUT and PCI_CLK_OUT multiplexors. The CFG_CLKIN_DIV configuration input selects whether CLKIN or CLKIN/2 is driven out on the PCI_SYNC_OUT signal.

PCI_SYNC_OUT is connected externally to PCI_SYNC_IN to allow the internal clock subsystem to synchronize to the system PCI clocks. PCI_SYNC_OUT must be connected properly to PCI_SYNC_IN, with equal delay to all PCI agent devices in the system, to allow the MPC8360E to function.

When the MPC8360 is configured as a PCI host device ($RCWH[PCIHOST] = 1$) and PCI clock output buffers are disabled ($RCWH[PCICKDRV] = 0$), PCI clock distribution should be done externally on the board. Therefore, PCI_SYNC_IN is the primary input clock when PCI clock output buffers are disabled.

4.4.1.1 PCI Clock Outputs (PCI_CLK_OUT[0:2])

When the device is configured as a PCI host and PCI clock output buffers are enabled in $RCWH[PCICKDRV]$, it provides three clock output signals, PCI_CLK_OUT[0:2], for external PCI agents. Each clock output can be configured independently by a memory-mapped register. See [Section 4.5.2.2, “Output Clock Control Register \(OCCR\).”](#)

The PCI clock output buffers are enabled or disabled according to $RCWH[PCICKDRV]$. If PCI clock output buffers are enabled, each of the individual clock outputs can be enabled (enable toggling of the clock) by setting its corresponding $OCCR[PCICOEn]$ bit. All output clocks are phase aligned to each other and to PCI_SYNC_OUT.

4.4.2 Clocking In PCI Agent Mode

When the device is configured as a PCI agent, PCI_CLK is the primary input clock. In agent mode, the CLKIN signal should be tied to GND, and the clock output signals, PCI_CLK_OUT n and PCI_SYNC_OUT, are not used.

In agent mode, the CFG_CLKIN_DIV configuration input can be used to double the internal clock frequencies, if sampled as 1 during PORESET assertion. This feature is useful if a fixed internal frequency is desired regardless of whether the PCI clock is running at 33 or 66 MHz. PCI specifications require that the signal M66EN provides the PCI clock frequency information.

4.4.3 System Clock Domains

As shown in [Figure 4-9](#), the primary clock input (PCI_CLK/PCI_SYNC_IN) frequency is multiplied up by the system phase-locked loop (PLL) and the clock unit to create four major clock domains:

- The coherent system bus clock (*csb_clk*)
- The QUICC engine clock (*ce_clk*)
- The internal clock for the DDR controller (*ddr_clk*)
- The internal clock for the local bus interface unit and the Secondary DDR controller (*lbc_clk*)

The *csb_clk* frequency is derived from a complex set of factors that can be simplified into the following equation:

$$csb_clk = [PCI_SYNC_IN \times (1 + CFG_CLKIN_DIV)] \times SPMF$$

In PCI host mode, $PCI_SYNC_IN \times (1 + CFG_CLKIN_DIV)$ is the CLKIN frequency.

The *csb_clk* serves as the clock input to the e300 core. A second PLL inside the core multiplies up the *csb_clk* frequency to create the internal clock for the core (*core_clk*). The system and core PLL multipliers are selected by the SPMF and COREPLL fields in the reset configuration word low (RCWL), which is loaded at power-on reset or by one of the hard-coded reset options. See [Section 4.3, “Reset Configuration.”](#)

The *ce_clk* frequency is determined by the QUICC Engine PLL multiplication factor (RCWL[CEPMF]) and the QUICC Engine PLL division factor (RCWL[CEPDF]) according to the following equations:

When CLKIN is the primary input clock,

$$ce_clk = (\text{primary clock input} \times CEPMF) \div (1 + CEPDF)$$

When PCI_CLK is the primary input clock,

$$ce_clk = [\text{primary clock input} \times CEPMF \times (1 + \sim CFG_CLKIN_DIV)] \div (1 + CEPDF)$$

See [Section 4.3.2.1.3, “QUICC Engine PLL Multiplication Factor,”](#) and [Section 4.3.2.1, “Reset Configuration Word Low Register \(RCWLR\),”](#) for more information.

The internal *ddr_clk* frequency (for DDR) is determined by RCWL[DDRCM]. Note that the *lb_clk* clock frequency (for Secondary DDR) is determined by RCWL[LBCM]. See [Section 4.3.2.1, “Reset Configuration Word Low Register \(RCWLR\).”](#) Note that *ddr_clk* is not the external memory bus frequency; *ddr_clk* passes through the DDR clock divider ($\div 2$) to create the differential DDR memory bus clock outputs (MCK and \overline{MCK}). However, the data rate is the same frequency as *ddr_clk*.

The internal *lbc_clk* frequency is determined by RCWL[LBCM]. See [Section 4.3.2.1, “Reset Configuration Word Low Register \(RCWLR\).”](#) Note that *lbc_clk* is not the external local bus or Secondary DDR frequency; *lbc_clk* passes through the LBC clock divider to create the external local bus clock outputs (LSYNC_OUT and LCLK[0:2]) also *lb_clk* passes through the Secondary DDR clock divider ($\div 2$) to create the differential Secondary DDR memory bus clock outputs (MCK and \overline{MCK}). The LBC clock divider ratio is controlled by LCCR[CLKDIV]. See [Section 10.1.2.1, “LBC Bus Clock and Clock Ratios,”](#) for more information.

In addition, some of the internal units may be required to be shut off or operate at lower frequency than the *csb_clk* frequency. These units have a default clock ratio that can be configured by a memory mapped register after the device comes out of reset. [Table 4-27](#) specifies which units have a configurable clock frequency. Refer to [Section 4.5.2.3, “System Clock Control Register \(SCCR\).”](#)

Table 4-27. Configurable Clock Units

Unit	Default Frequency	Options
Security core	<i>csb_clk</i> /3	Off, <i>csb_clk</i> , <i>csb_clk</i> /2, <i>csb_clk</i> /3
PCI and DMA complex	<i>csb_clk</i>	Off, <i>csb_clk</i>

NOTE

The clock ratios of these units must be set before they are accessed.

4.5 Memory Map/Register Definitions

This section presents the memory maps and register descriptions for both reset and clocking.

4.5.1 Reset Configuration Register Descriptions

The reset configuration and status registers are shown in [Table 4-28](#).

Table 4-28. Reset Configuration and Status Registers Memory Map

Address	Register	Access	Reset	Section/Page
0x0_0900	Reset configuration word low register (RCWLR)	R	All zeros	4.5.1.1/4-34
0x0_0904	Reset configuration word high register (RCWHR)	R	All zeros	4.5.1.2/4-34
0x0_0908	Reserved, should be cleared	—	—	—
0x0_090C	Reserved, should be cleared	—	—	—
0x0_0910	Reset status register (RSR)	R/W	All zeros	4.5.1.3/4-35
0x0_0914	Reset mode register (RMR)	R/W	All zeros	4.5.1.4/4-36
0x0_0918	Reset protection register (RPR)	R/W	All zeros	4.5.1.5/4-37
0x0_091C	Reset control register (RCR)	R/W	All zeros	4.5.1.6/4-37
0x0_0920	Reset control enable register (RCER)	R/W	All zeros	4.5.1.7/4-38
0x0_0924– 0x0_09FC	Reserved, should be cleared.	—	—	—

4.5.1.1 Reset Configuration Word Low Register (RCWLR)

The reset configuration word low register (RCWLR) is shown in [Figure 4-3](#) and described in [Section 4.3.2.1](#), “Reset Configuration Word Low Register (RCWLR).”

4.5.1.2 Reset Configuration Word High Register (RCWHR)

The reset configuration word high register (RCWHR) is shown in [Figure 4-4](#) and described in [Section 4.3.2.2](#), “Reset Configuration Word High Register (RCWHR).”

4.5.1.3 Reset Status Register (RSR)

RSR, shown in Figure 4-10, captures various reset events in the device. The RSR accumulates reset events. For example, because software watchdog expiration results in a hard reset, SWRS and HRS are all set after a software watchdog reset. This register returns to its reset value only when power-on reset occurs.

Address 0x0_0910

Access: User read/write

	0	2	3								14	15		
R	RSTSRC		—									BSF		
W	RSTSRC		—									BSF		
Reset	n1	0	0	0	0	0	0	0	0	0	0	0		
	16	17	18	19	20	22	23	24	26	27	28	29	30	31
R	—	SWSR	SWHR	—		JSRS	—		CSHR	SWRS	BMRS	SRS	HRS	
W	—	SWSR	SWHR	—		JSRS	—		CSHR	SWRS	BMRS	SRS	HRS	
Reset	All zeros													

¹ The reset value of this field is determined according to the reset configuration input signals CFG_RESET_SOURCE[0:2] sampled during the reset flow.

Figure 4-10. Reset Status Register (RSR)

Table 4-29 defines the reset status register bit fields.

Table 4-29. Reset Status Register Field Descriptions

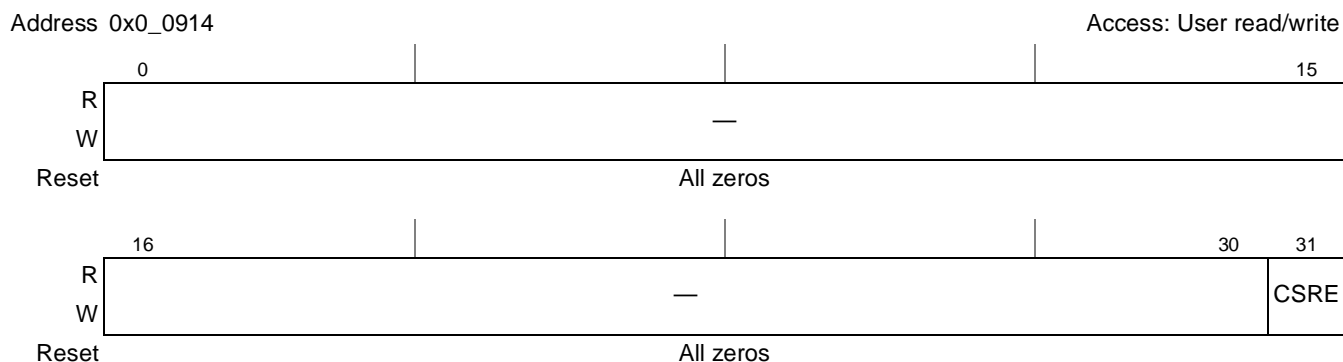
Bits	Name	Description
0–2	RSTSRC	Reset configuration word source. Reflects the value of CFG_RESET_SOURCE input signal during the reset flow. See Section 4.3.1.1, “Reset Configuration Word Source.” Changing this field has no effect.
3–14	—	Reserved, should be cleared.
15	BSF	Boot sequencer fail. If set, indicates that the I ² C boot sequencer has failed while loading the reset configuration words. Cleared by writing a 1 to it (writing zero has no effect).
16–17	—	Reserved, should be cleared.
18	SWSR	Software soft reset. If set, indicates that a software soft reset has occurred. Cleared by writing a 1 to it (writing zero has no effect).
19	SWHR	Software hard reset. If set, indicates a software hard reset. SWHR is cleared by writing a 1 to it (writing zero has no effect).
20–22	—	Reserved, should be cleared.
23	JSRS	JTAG soft reset status. Set when the JTAG reset request is set and remains set until software clears it. JSRS is cleared by writing a 1 to it (writing zero has no effect). 0 No JTAG reset event. 1 JTAG reset event.
24–26	—	Reserved, should be cleared.
27	CSHR	Check stop reset status. When the core enters a checkstop state and the checkstop reset is enabled by the RMR[CSRE], CSRS is set and it remains set until software clears it. CSRS is cleared by writing a 1 to it (writing zero has no effect). 0 No enabled check stop reset event. 1 Enabled check stop reset event.

Table 4-29. Reset Status Register Field Descriptions (continued)

Bits	Name	Description
28	SWRS	Software watchdog reset status. When a software watchdog expire event (which causes a reset) is detected, SWRS is set and remains that way until the software clears it. SWRS is cleared by writing a 1 to it (writing zero has no effect). 0 No software watchdog reset event. 1 Software watchdog reset event.
29	BMRS	Bus monitor reset status. When a bus monitor expire event (which causes a reset) is detected, BMRS is set and remains set until the software clears it. BMRS can be cleared by writing a 1 to it (writing zero has no effect). 0 No bus monitor reset event. 1 Bus monitor reset event.
30	SRS	Soft reset status. When an external or internal soft reset event is detected, SRS is set and remains set until software clears it. SRS is cleared by writing a 1 to it (writing zero has no effect). 0 No soft reset event. 1 Soft reset event.
31	HRS	Hard reset status. When an external or internal hard reset event is detected, HRS is set and remains set until software clears it. HRS is cleared by writing a 1 (writing zero has no effect). 0 No hard reset event. 1 Hard reset event.

4.5.1.4 Reset Mode Register (RMR)

RMR, shown in [Figure 4-11](#), enables a hard reset sequence on the device when the e300 core enters checkstop state.


Figure 4-11. Reset Mode Register (RMR)

[Table 4-30](#) describes the RMR fields.

Table 4-30. RMR Field Descriptions

Bits	Name	Function
0–30	—	Reserved, should be cleared.
31	CSRE	Checkstop reset enable. The core can enter checkstop mode as the result of several exception conditions. Setting CSRE configures the device to perform a hard reset sequence when the core enters checkstop state. 0 Reset not generated when core enters checkstop state. 1 Reset generated when core enters checkstop state.

4.5.1.5 Reset Protection Register (RPR)

RPR, shown in Figure 4-12, prevents unintended software reset requests caused by writes to the reset control register (RCR). To disable a write to the reset control register (RCR), the user should write a 1 to RCER[CRE].

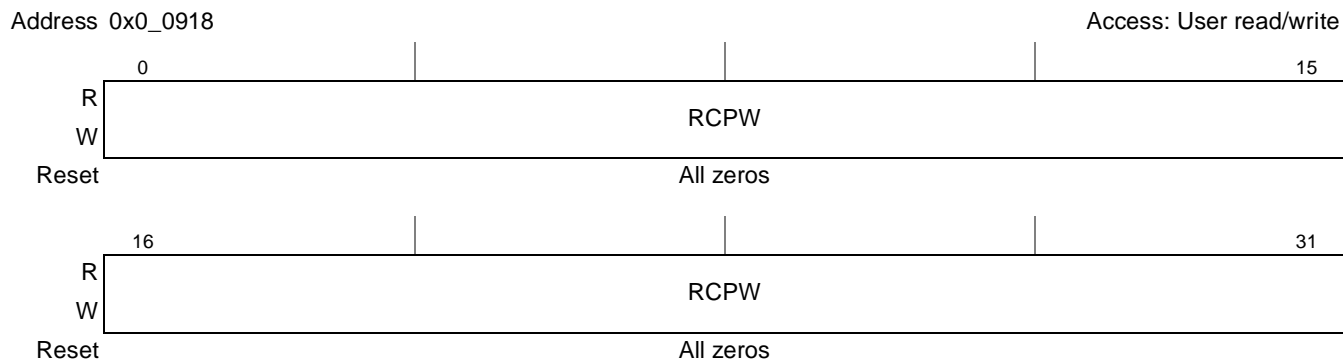


Figure 4-12. Reset Protection Register (RPR)

Table 4-31 defines the bit fields of RPR.

Table 4-31. RPR Bit Descriptions

Bits	Name	Description
0–31	RCPW	Reset control protection word. Prevents unintended software reset requests because of a write to the RCR. The user should write the value 0x5253_5445 (RSTE in ASCII) to enable. Enable indication appears in the reset control enable register (RCER[CRE]). Reading this register always returns all zeros.

4.5.1.6 Reset Control Register (RCR)

RCR, shown in Figure 4-13, can be used by software to initiate a hard reset sequence. To allow writing to this register, the user must enable it by writing the value 0x5253_5445 to the RPR.

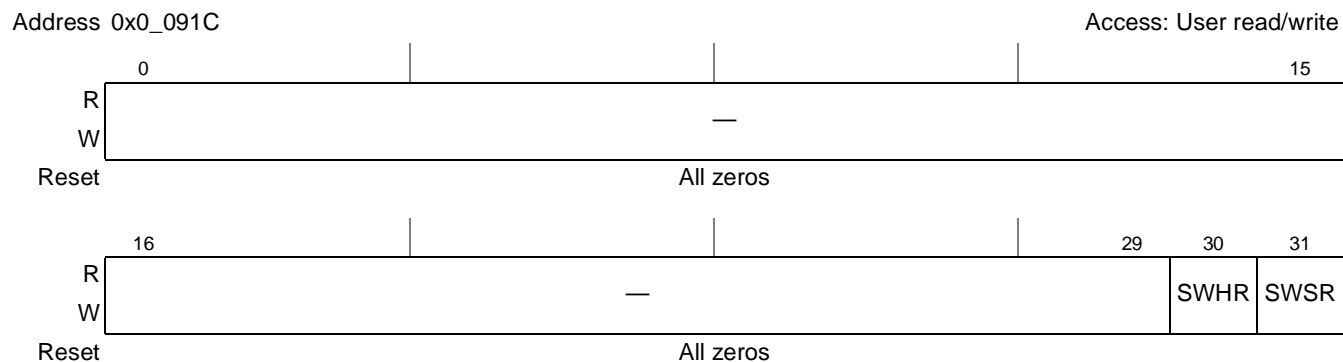


Figure 4-13. Reset Control Register (RCR)

Table 4-32 defines the bit fields of RCR.

Table 4-32. RCR Bit Settings

Bits	Name	Description
0–29	—	Reserved, should be cleared.
30	SWHR	Software hard reset. Setting this bit causes the device to begin a hard reset flow. This bit returns to its reset state during the reset sequence, so reading it always returns all zeros.
31	SWSR	Software soft reset. Setting this bit causes the device to begin a soft reset flow. This bit returns to its reset state during the reset sequence, so reading it always returns 0. H

4.5.1.7 Reset Control Enable Register (RCER)

RCER, shown in Figure 4-14, indicates by the CRE field that the RPR is accessed with a value that enables RCR.



Figure 4-14. Reset Control Enable Register (RCER)

Table 4-33 defines the bit fields of RCER.

Table 4-33. RCER Bit Settings

Bits	Name	Description
0–30	—	Reserved, should be cleared.
31	CRE	Control register enabled. When set, indicates that the RPR was accessed with a value that enables the RCR. Writing 1 to this bit disables the RCR and clears this bit. Writing zero has no effect.

4.5.2 Clock Configuration Registers

The clock configuration and status registers are shown in Table 4-34.

Table 4-34. Clock Configuration Registers Memory Map

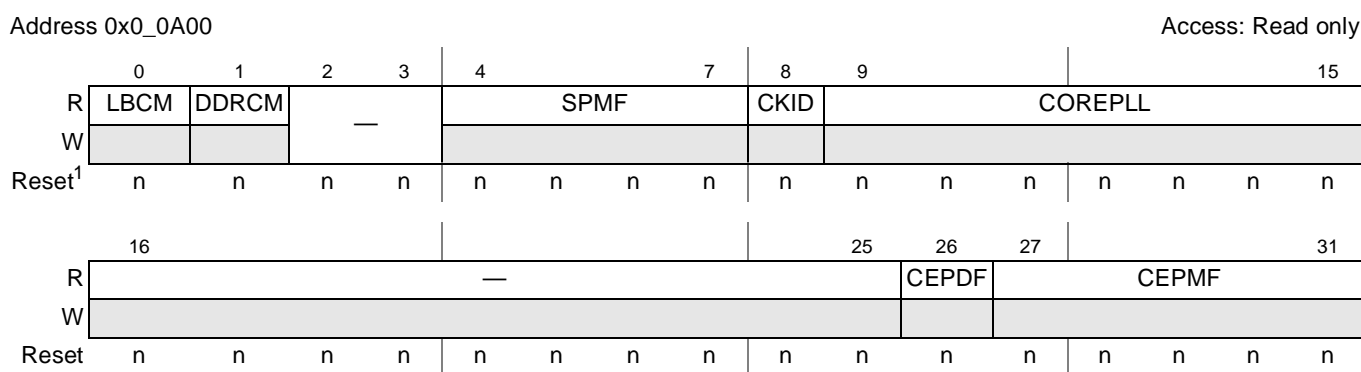
Address	Register	Access	Reset	Section/Page
0x0_0A00	System PLL mode register (SPMR)	R	0xn _{nnn} _n _{nnn}	4.5.2.1/4-39
0x0_0A04	Output clock control register (OCCR)	R/W	All zeros	4.5.2.2/4-40

Table 4-34. Clock Configuration Registers Memory Map (continued)

Address	Register	Access	Reset	Section/Page
0x0_0A08	System clock control register (SCCR)	R/W	0xFFFF_FFFF	4.5.2.3/4-41
0x0_0A0C– 0x0_0AFC	Reserved, should be cleared	—	—	—

4.5.2.1 System PLL Mode Register (SPMR)

SPMR is shown in [Figure 4-15](#), gets its values according to the CFG_CLKIN_DIV reset configuration input signal and the reset configuration word low loaded during the reset flow. Note that this register is updated only during a power-on reset sequence and not by a hard reset sequence. It may hold values different than those in the RCWLR after a hard reset sequence.


Figure 4-15. System PLL Mode Register

¹ See [Table 4-35](#) for reset values.

[Table 4-35](#) defines the system PLL mode register bit fields.

Table 4-35. System PLL Mode Register Bit Settings

Bits	Name	Meaning	Description
0	LBCM	Local bus memory controller and Secondary DDR SDRAM memory controller clock mode.	Section 4.3.2.1, “Reset Configuration Word Low Register (RCWLR)”
1	DDRCM	DDR SDRAM memory controller clock mode.	Section 4.3.2.1, “Reset Configuration Word Low Register (RCWLR)”
2–3	—	Reserved, should be cleared.	—
4–7	SPMF	System PLL multiplication factor	Section 4.3.2.1.2, “System PLL Configuration”
8	CKID	CLKIN division factor. Reflects the value of CFG_CLKIN_DIV input signal during the reset flow.	Section 4.3.1.2, “CLKIN Division”
9–15	COREPLL	Core PLL configuration.	See the hardware specifications for this device
16–25	—	Reserved, should be cleared.	
24–25	CEVCOD	QUICC Engine PLL VCO division	Section 4.3.2.1, “Reset Configuration Word Low Register (RCWLR).”

Table 4-35. System PLL Mode Register Bit Settings (continued)

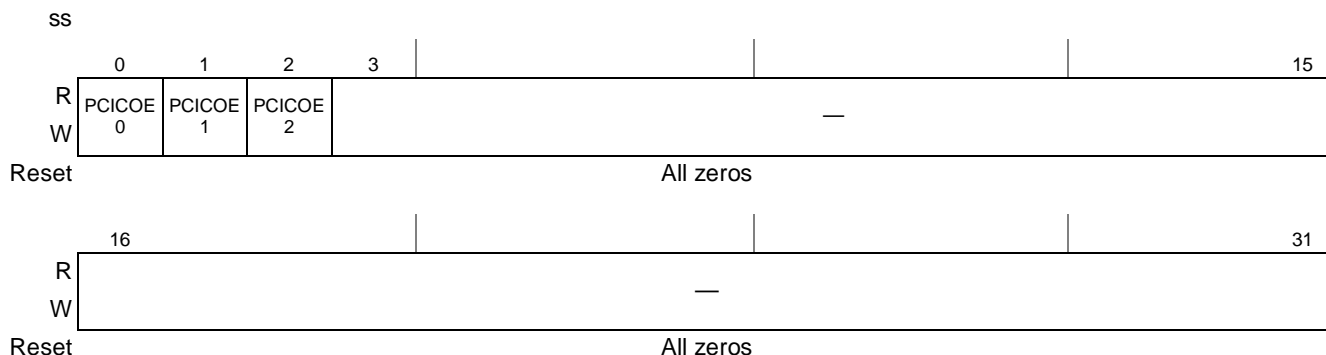
Bits	Name	Meaning	Description
26	CEPDF	QUICC Engine PLL division factor	Section 4.3.2.1, “Reset Configuration Word Low Register (RCWLR)”
27–31	CEPMF	QUICC Engine PLL multiplication factor	Section 4.3.2.1.3, “QUICC Engine PLL Multiplication Factor”

4.5.2.2 Output Clock Control Register (OCCR)

The OCCR shown in [Figure 4-16](#), controls the device output clocks. It is possible to control some output clock modes by writing to this memory mapped register as described below.

Addr: 0x0_0A04

Access: Read/Write


Figure 4-16. Output Clock Control Register (OCCR)

[Table 4-36](#) defines the bit fields of OCCR.

Table 4-36. OCCR Bit Settings

Bits	Name	Description
0	PCICOE0	PCI_CLK_OUT0 enable. 0 PCI_CLK_OUT0 signal is disabled (drive constant zero). 1 PCI_CLK_OUT0 signal is enabled to toggle.
1	PCICOE1	PCI_CLK_OUT1 enable. 0 PCI_CLK_OUT1 signal is disabled (drive constant zero). 1 PCI_CLK_OUT1 signal is enabled to toggle.
2	PCICOE2	PCI_CLK_OUT2 enable. 0 PCI_CLK_OUT2 signal is disabled (drive constant zero). 1 PCI_CLK_OUT2 signal is enabled to toggle.
3–31	—	Reserved, should be cleared.

4.5.2.3 System Clock Control Register (SCCR)

SCCR, shown in Figure 4-17, controls device units that have a configurable clock ratio.

Address 0x0_0A08

Access: Read/Write

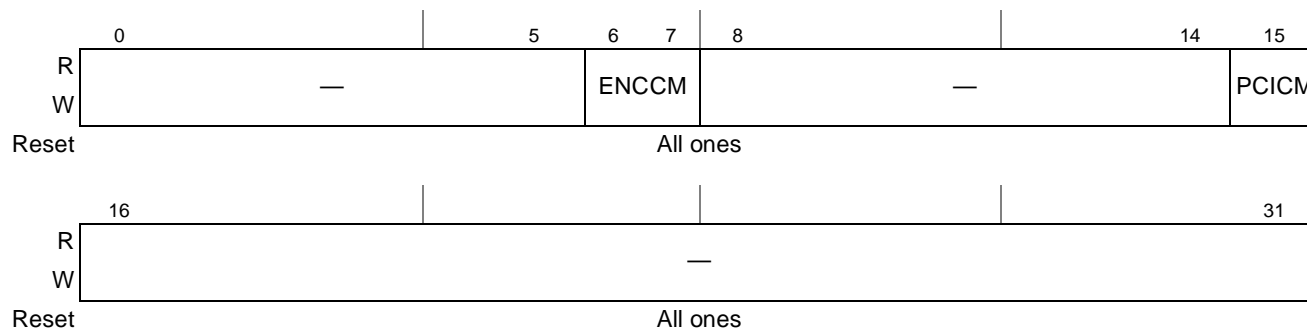


Figure 4-17. System Clock Control Register (SCCR)

Table 4-37 defines the bit fields of SCCR.

Table 4-37. SCCR Bit Descriptions

Bits	Name	Description
0–5	—	Reserved, should be cleared.
6–7	ENCCM	Encryption core and I ² C1 clock mode. 00 Encryption core clock is disabled. 01 Encryption core clock/ <i>csb_clk</i> ratio is 1:1 (for <i>csb_clk</i> < 166 MHz). 10 Encryption core clock/ <i>csb_clk</i> ratio is 1:2 (<i>csb_clk</i> has higher frequency than the encryption core). 11 Encryption core clock/ <i>csb_clk</i> ratio is 1:3 (<i>csb_clk</i> has higher frequency than the encryption core). Note: The encryption core must have the same clock ratio as the USB unit, unless one of them has its clock disabled.
8–14	—	Reserved, should be cleared.
15	PCICM	PCI clock mode. Define the clock mode for all of the PCI complex - PCI and DMA. 0 PCI complex clocks are disabled. 1 PCI complex clocks are enabled.
16–31	—	Reserved

4.5.3 Clock Control DDR Registers

NOTE

There is a DDR clock control register for each DDR controller. MCKENR1, for the primary controller, is at 0x00_1010 (offset 0x10 from the 0x0_1000); MCKENR2 (secondary controller) is at 0x00_0F10 (offset 0x10 from the 0x0_0F00).

The programmable clock control DDR register maps occupy 20 bytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All registers are 32 bits wide located on 32-bit address boundaries. All addresses used in this chapter are offsets from the clock control DDR starting addresses as defined in Chapter 2, “Memory Map.”

Table 4-38. Clock Control DDR Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00–0x0F	Reserved, should be cleared.	R	All zeros	—
0x10	MCK enable register (MCKENR _n)	R/W	0xFC00_0000	4.5.3.1/4-42
0x14–0xFF	Reserved, should be cleared.	—	—	—

4.5.3.1 MCK Enable Register (MCKENR_n)

MCKENR_n, shown in [Figure 4-18](#), enables or disables the DDR clock outputs.

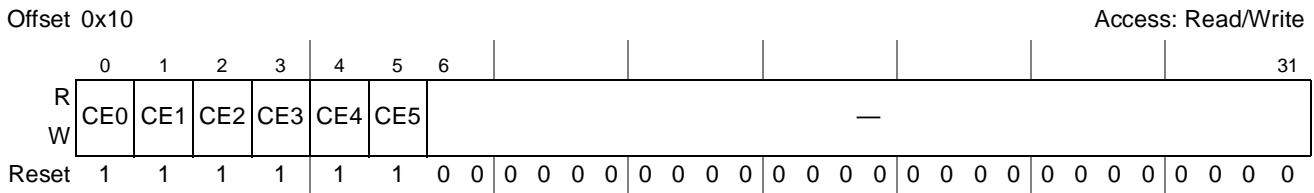


Figure 4-18. MCK Enable Register (MCKENR_n)

[Table 4-39](#) describes the MCKENR_n fields.

Table 4-39. MCKENR_n Field Descriptions

Bits	Name	Description
0	CE0	Enable/Disable MCK[0] clock output signals 0 Disable MCK[0] and $\overline{\text{MCK}}[0]$ 1 Enable MCK[0] and $\overline{\text{MCK}}[0]$
1	CE1	Enable/Disable MCK[1] clock output signals 0 Disable MCK[1] and $\overline{\text{MCK}}[1]$ 1 Enable MCK[1] and $\overline{\text{MCK}}[1]$
2	CE2	Enable/Disable MCK[2] clock output signals 0 Disable MCK[2] and $\overline{\text{MCK}}[2]$ 1 Enable MCK[2] and $\overline{\text{MCK}}[2]$
3	CE3	Enable/Disable MCK[3] clock output signals 0 Disable MCK[3] and $\overline{\text{MCK}}[3]$ 1 Enable MCK[3] and $\overline{\text{MCK}}[3]$
4	CE4	Enable/Disable MCK[4] clock output signals 0 Disable MCK[4] and $\overline{\text{MCK}}[4]$ 1 Enable MCK[4] and $\overline{\text{MCK}}[4]$
5	CE5	Enable/Disable MCK[5] clock output signals 0 Disable MCK[5] and $\overline{\text{MCK}}[5]$ 1 Enable MCK[5] and $\overline{\text{MCK}}[5]$
6–31	—	Reserved, should be cleared.

Chapter 5

System Configuration

5.1 Introduction

This chapter describes several functions that control the local access windows, system configuration, protection, and general utilities. These functions are discussed in the following sections:

- [Section 5.2, “Local Memory Map Overview and Example”](#)
- [Section 5.3, “QUICC Engine Secondary Bus Access Windows”](#)
- [Section 5.4, “System Configuration”](#)
- [Section 5.5, “Software Watchdog Timer \(WDT\)”](#)
- [Section 5.6, “Real Time Clock Module \(RTC\)”](#)
- [Section 5.7, “Periodic Interval Timer \(PIT\)”](#)
- [Section 5.8, “General-Purpose Timers \(GTMs\)”](#)
- [Section 5.9, “Power Management Control \(PMC\)”](#)

5.2 Local Memory Map Overview and Example

The device provides a flexible local memory map. The local memory map refers to the 32-bit address space seen by the processor as it accesses memory and I/O space. Internal DMA engines also see this same local memory map. All memory accessed by the DDR SDRAM, secondary DDR SDRAM, and local bus memory controllers exists in this memory map, as do all memory-mapped configuration, control, and status registers.

The local memory map is defined by a set of eleven local access windows. Each of these windows maps a region of memory to a particular target interface, such as the DDR SDRAM controller or the PCI controller. The DSP subsystem is not operational in the MSC7104. Note that the local access windows do not perform any address translation. The size of each window can be configured from 4 Kbytes to 2 Gbytes. Each local access window is assigned to a specific target interface as specified in [Table 5-1](#).

Table 5-1. Local Access Windows Target Interface

Window Number	Target Interface	Comments
0	Configuration registers (IMMR)	Fixed 2-Mbyte window size
1	Local bus	—
2	Local bus	—
3	Local bus	—
4	Local bus	—

Table 5-1. Local Access Windows Target Interface (continued)

Window Number	Target Interface	Comments
5	PCI	—
6	PCI	—
7	DDR SDRAM	—
8	DDR SDRAM	—
9	Secondary DDR SDRAM	—
10	Secondary DDR SDRAM	—

Figure 5-1 shows an example memory map.

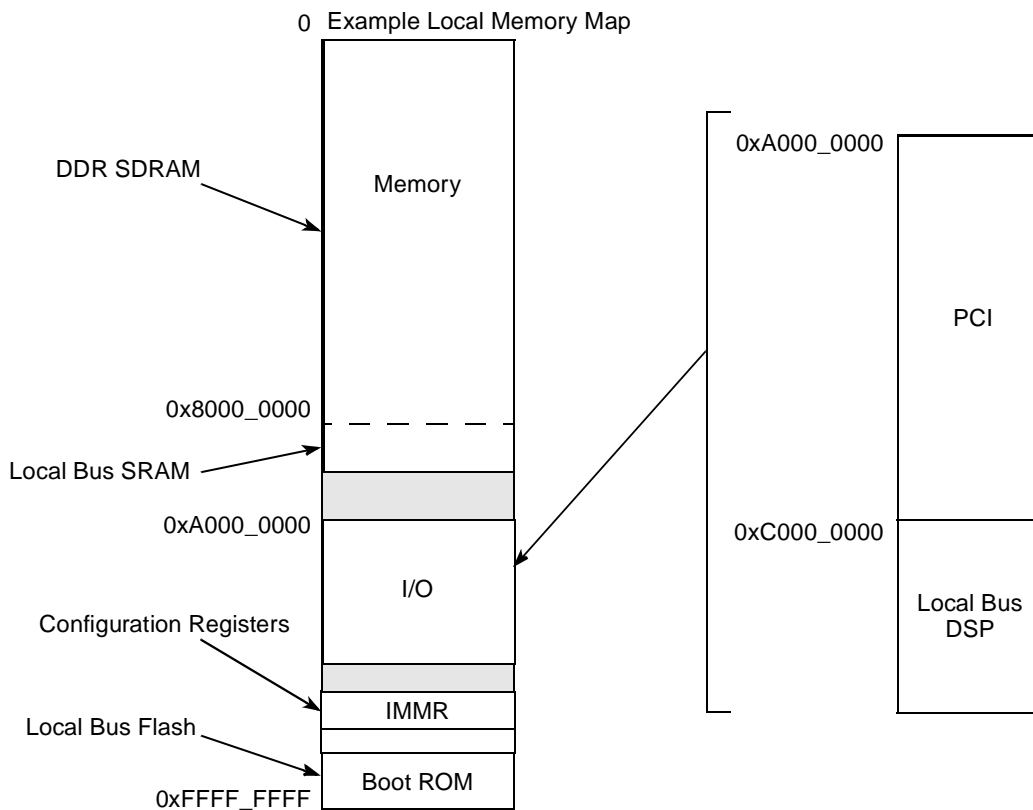


Figure 5-1. Local Memory Map Example

Table 5-2 shows one example of local access window settings.

Table 5-2. Local Access Windows Example

Window	Base Address	Size	Target Interface
7	All zeros	2 Gbytes	DDR SDRAM
2	0x8000_0000	1 Mbyte	Local bus
5	0xA000_0000	512 Mbytes	PCI
3	0xC000_0000	256 Mbytes	Local bus

Table 5-2. Local Access Windows Example (continued)

Window	Base Address	Size	Target Interface
0	0xFF40_0000	2 Mbyte	Configuration registers (IMMR)
1	0xFF80_0000	8 Mbytes	Local bus boot ROM Flash
4, 6, 8, 9, 10	Unused		

In this example, the local access window of the boot ROM is defined as window number 1, on a local bus device, in the highest 8 Mbytes of memory as set by the reset configuration word high during the reset sequence (see [Section 4.3.2.2.4, “Boot ROM Location”](#)) and [Section 5.2.4.3.1, “LBLAWBAR0\[BASE_ADDR\] Reset Value.”](#) The local access window, which describes the range of memory used for memory-mapped registers (IMMR), is a fixed 2-Mbyte space pointed to by the IMMRBAR register, using its default value (0xFF40_0000). See [Section 5.2.4.1, “Internal Memory Map Registers Base Address Register \(IMMRBAR\).”](#)

5.2.1 Address Translation and Mapping

In addition to any address translation performed by the e300c1 core MMU, three distinct types of translation and mapping operations are performed on transactions at the integrated device level. These are as follows:

- Mapping a local address to a target interface
- Translating the local 32-bit address to an external address space
- Translating external addresses to the local 32-bit address space

The local access windows perform target mapping for transactions within the local address space. The local access windows do not perform any address translation.

Outbound windows perform the mapping from the local 32-bit address space to the address space of PCI, which may be much larger than the local space.

Inbound windows perform address translation from the external address spaces of PCI to the local address space.

The target mappings created by an inbound window must be consistent with those of the local access windows. That is, if an inbound window maps a transaction to a given local address, a valid local access window for that address must be set independently.

All of the configuration registers that define mapping of local access windows follow the same register format. [Table 5-3](#) summarizes the general format of these window definitions.

Table 5-3. Format of Window Definitions

Register	Function
Base address	High-order address bits defining location of the window in the initial address space
Window size/attributes	Window enable, window size ¹

¹ An exception is the IMMR window, which is always enabled and has a fixed 2-Mbyte size.

Windows must be a power-of-two size. To perform a mapping function, the address of the transaction is compared with the base address register of each window. The number of bits used in the comparison is dictated by each window's size attribute. When an address hits within a window, the transaction is directed to the appropriate target.

5.2.2 Window into Configuration Space

The internal memory map registers' base address register (IMMRBAR) defines a window that is used to access all memory-mapped configuration, control, and status registers, referred to as internal memory map registers or IMMR. This window is always enabled with a fixed size of 2 Mbytes, and no other attributes are attached so there is no associated size/attribute register. This window always takes precedence over all local access windows. The IMMRBAR always come out of reset with a default base address value of 0xFF40_0000, and this base address value can be modified by writing to this register. For more information, see [Section 5.2.4.1, "Internal Memory Map Registers Base Address Register \(IMMRBAR\)."](#)

NOTE

Although it is legal to use the 2-Mbyte space consecutive to the 2 Mbytes of the IMMR (for example, if IMMRBAR is 0xFF40_0000, the 2-Mbyte address space consecutive to it is 0xFF60_0000–0xFF7F_FFFF), it is not recommended. This space may be used in future derivatives of the device that require a larger internal memory space.

5.2.3 Local Access Windows

As demonstrated in the address map overview in [Section 5.2, "Local Memory Map Overview and Example,"](#) local access windows associate a range of the local 32-bit address space with a particular target interface. This allows the internal interconnections of the device to route a transaction from its source to the proper target. No address translation is performed. The base address defines the high order address bits that give the location of the window in the local address space. The window attributes enable the window and define its size, while the window number specifies the target interface.

With the exception of configuration space (mapped by IMMRBAR), all addresses used by the system must be mapped by a local access window. This includes addresses that are mapped by PCI inbound windows.

The local access window registers exist as part of the local access block in the system configuration registers. See [Section 5.4.3, "System Configuration Registers."](#) A detailed description of the local access window registers is given in the following sections. Note that the minimum size of a window is 4 Kbytes, so the low order 12 bits of the base address cannot be specified.

5.2.3.1 Local Access Register Memory Map

Table 5-4 shows the memory map for the local access registers.

Table 5-4. Local Access Register Memory Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0000	Internal memory map base address register (IMMRBAR)	R/W	0xFF40_0000	5.2.4.1/5-6
0x0_0004	Reserved	—	—	—
0x0_0008	Alternate configuration base address register (ALTCBAR)	R/W	All zeros	5.2.4.2/5-7
0x0_000C– 0x0_001C	Reserved	—	—	—
0x0_0020	LBC local access window 0 base address register (LBLAWBAR0)	R/W	All zeros ¹	5.2.4.3/5-8
0x0_0024	LBC local access window 0 attribute register (LBLAWAR0)	R/W	All zeros ²	5.2.4.4/5-9
0x0_0028	LBC local access window 1 base address register (LBLAWBAR1)	R/W	All zeros	5.2.4.3/5-8
0x0_002C	LBC local access window 1 attribute register (LBLAWAR1)	R/W	All zeros	5.2.4.4/5-9
0x0_0030	LBC local access window 2 base address register (LBLAWBAR2)	R/W	All zeros	5.2.4.3/5-8
0x0_0034	LBC local access window 2 attribute register (LBLAWAR2)	R/W	All zeros	5.2.4.4/5-9
0x0_0038	LBC local access window 3 base address register (LBLAWBAR3)	R/W	All zeros	5.2.4.3/5-8
0x0_003C	LBC local access window 3 attribute register (LBLAWAR3)	R/W	All zeros	5.2.4.4/5-9
0x0_0040– 0x0_005C	Reserved	—	—	—
0x0_0060	PCI local access window 0 base address register (PCILAWBAR0)	R/W	All zeros ³	5.2.4.5/5-10
0x0_0064	PCI local access window 0 attribute register (PCILAWAR0)	R/W	All zeros ⁴	5.2.4.6/5-11
0x0_0068	PCI local access window 1 base address register (PCILAWBAR1)	R/W	All zeros ⁵	5.2.4.5/5-10
0x0_006C	PCI local access window 1 attribute register (PCILAWAR1)	R/W	All zeros	5.2.4.6/5-11
0x0_0070– 0x0_009C	Reserved	—	—	—
0x0_00A0	DDR local access window 0 base address register (DDRLAWBAR0)	R/W	All zeros ⁶	5.2.4.7/5-12
0x0_00A4	DDR local access window 0 attribute register (DDRLAWAR0)	R/W	All zeros ⁷	5.2.4.8/5-13
0x0_00A8	DDR local access window 1 base address register (DDRLAWBAR1)	R/W	All zeros	5.2.4.7/5-12
0x0_00AC	DDR local access window 1 attribute register (DDRLAWAR1)	R/W	All zeros	5.2.4.8/5-13
0x0_00B0– 0x0_00DC	Reserved	—	—	—
0x0_00E0	Secondary DDR local access window 0 base address register (SDDRLAWBAR0)	R/W	All zeros	5.2.4.9/5-14
0x0_00E4	Secondary DDR local access window 0 attribute register (SDDRLAWAR0)	R/W	All zeros	5.2.4.10/5-14
0x0_00E8	Secondary DDR local access window 1 base address register (SDDRLAWBAR1)	R/W	All zeros	5.2.4.9/5-14
0x0_00EC	Secondary DDR local access window 1 attribute register (SDDRLAWAR1)	R/W	All zeros	5.2.4.10/5-14

- ¹ Depends on reset configuration word high values. See [Section 5.2.4.3.1, “LBLAWBAR0\[BASE_ADDR\] Reset Value,”](#) for details.
- ² Depends on reset configuration word high values. See [Section 5.2.4.4.1, “LBLAWAR0\[EN\] and LBLAWAR0\[SIZE\] Reset Value,”](#) for details.
- ³ Depends on reset configuration word high values. See [Section 5.2.4.5.1, “PCILAWBAR0\[BASE_ADDR\] Reset Value,”](#) for details.
- ⁴ Depends on reset configuration word high values. See [Section 5.2.4.7.1, “DDRLAWBAR0\[BASE_ADDR\] Reset Value,”](#) for details.
- ⁵ Depends on reset configuration word high values. See [Section 5.2.4.6.1, “PCILAWAR0\[EN\] and PCILAWAR0\[SIZE\] Reset Value,”](#) for details.
- ⁶ Depends on reset configuration word high values. See [Section 5.2.4.7.1, “DDRLAWBAR0\[BASE_ADDR\] Reset Value,”](#) for details.
- ⁷ Depends on reset configuration word high values. See [Section 5.2.4.8.1, “DDRLAWAR0\[EN\] and DDRLAWAR0\[SIZE\] Reset Value,”](#) for details.

5.2.4 Local Access Register Descriptions

5.2.4.1 Internal Memory Map Registers Base Address Register (IMMRBAR)

The IMMR window contains configuration, control, and status registers, as well as internal device memory arrays. The internal memory map occupies a 2-Mbyte region of memory space. Its location is programmable using the internal memory map register (IMMR). The default base address for the internal memory map register is 0xFF40_0000. Because IMMRBAR is at offset 0x0 from the beginning of the local access registers, IMMRBAR always points to itself.

5.2.4.1.1 Updating IMMRBAR

Updates to IMMRBAR that relocate the entire 2-Mbyte region of the internal memory block require special treatment. The effect of the update must be guaranteed to be visible by the mapping logic before an access to the new location is seen. To make sure this happens, the following guidelines should be followed:

- IMMRBAR should be updated during initial configuration of the device when only one host or controller has access to the device as follows:
 - If an external host on PCI is configuring the device, it should set IMMRBAR to the desired final location before the e300c1 core is released to boot.
 - If the core is initializing the device, it should set IMMRBAR to the desired final location before enabling other I/O devices to access the device.
- When the e300 core is writing to IMMRBAR, it should use the following sequence:
 - Read the current value of IMMRBAR using a load word instruction followed by an **isync**. This forces all accesses to configuration space to complete.
 - Write the new value to IMMRBAR.
 - Perform a load of an address that does not access configuration space or the on-chip SRAM, but has an address mapping already in effect (for example, boot ROM). Follow this load with an **isync**.
 - Read the contents of IMMRBAR from its new location, followed by another **isync**.

The IMMRBAR is shown in [Figure 5-2](#).

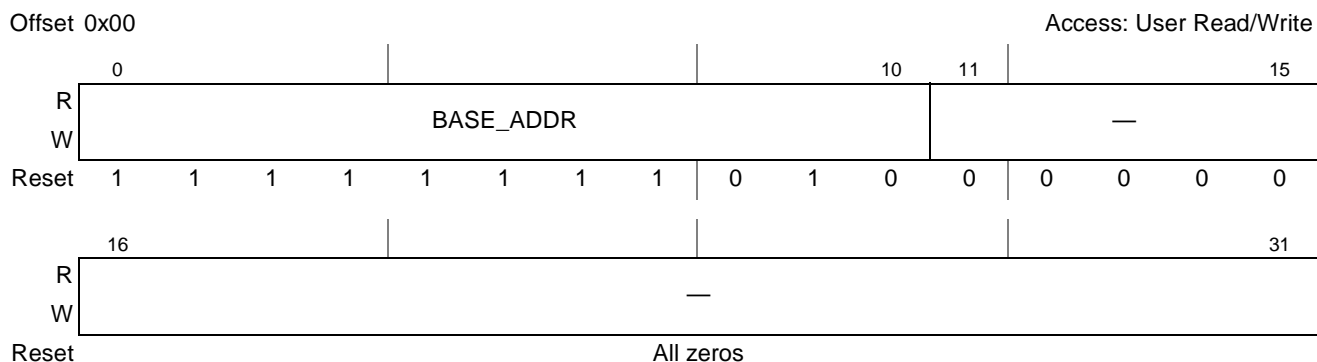


Figure 5-2. Internal Memory Map Registers' Base Address Register (IMMRBAR)

[Table 5-5](#) defines the bit fields of IMMRBAR.

Table 5-5. IMMRBAR Bit Settings

Bits	Name	Description
0–10	BASE_ADDR	Identifies the 11 most-significant address bits of the base of the 2-Mbyte internal memory window.
11–31	—	Reserved. Software must write all zeros.

5.2.4.2 Alternate Configuration Base Address Register (ALTCBAR)

The alternate configuration base address register (ALTCBAR) is used to define the base address for an alternate 1-Mbyte region of configuration space to be used by the boot sequencer. By loading the proper boot sequencer command in the serial ROM, the base address in the ALTCBAR can be combined with the 20 bits of address offset supplied from the serial ROM to generate a 32-bit address. Thus, by configuring this register, the boot sequencer has access to the entire memory map, one 1-Mbyte block at a time. See [Section 15.4.5, “Boot Sequencer Mode,”](#) for more information.

NOTE

ALTCBAR is not considered a local access window on its own, so the boot sequencer must configure one of the other local access windows properly to reach the desired target peripherals.

The alternate configuration base address register is shown in [Figure 5-3](#).



Figure 5-3. Alternate Configuration Base Address Register (ALTCBAR)

Table 5-6 defines the bit fields of ALTCBAR.

Table 5-6. ALTCBAR Bit Settings

Bits	Name	Description
0–11	BASE_ADDR	Identifies the 12 most-significant address bits of an alternate base address used for boot sequencer configuration accesses.
12–31	—	Reserved. Write has no effect, read returns 0.

5.2.4.3 LBC Local Access Window *n* Base Address Registers (LBLAWBAR0–LBLAWBAR3)

The LBC local access window *n* base address registers (LBLAWBAR0–LBLAWBAR3) are shown in Figure 5-4.



1. The LBLAWBAR0[BASE_ADDR] reset value depends on the reset configuration word high values. See Section 5.2.4.3.1, “LBLAWBAR0[BASE_ADDR] Reset Value,” for a detailed description.

Figure 5-4. LBC Local Access Window *n* Base Address Registers (LBLAWBAR0–LBLAWBAR3)

Table 5-7 defines the bit fields of LBLAWBAR0–LBLAWBAR3.

Table 5-7. LBLAWBAR0–LBLAWBAR3 Bit Settings

Bits	Name	Description
0–19	BASE_ADDR	Identifies the 20 most-significant address bits of the base of local access window <i>n</i> . The specified base address should be aligned to the window size, as defined by LBLAWARn[SIZE].
20–31	—	Reserved. Write has no effect, read returns 0.

5.2.4.3.1 LBLAWBAR0[BASE_ADDR] Reset Value

The core may also use a local bus peripheral device to fetch its boot vector. For this purpose, the LBLAWBAR0[BASE_ADDR] reset value is set according to the value set in the reset configuration word high BMS field.

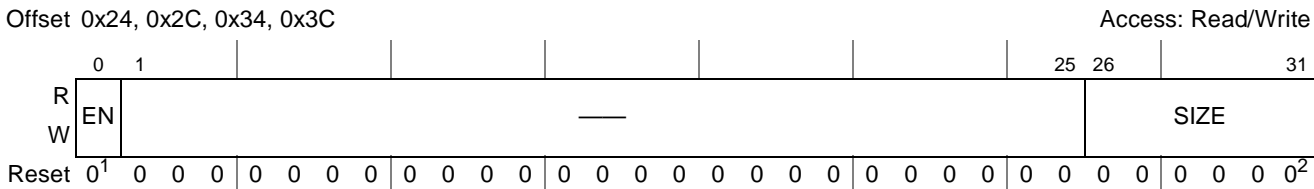
Table 5-8 defines the reset value of LBLAWBAR0[BASE_ADDR].

Table 5-8. LBLAWBAR0[BASE_ADDR] Reset Value

RCWHR[BMS]	BASE_ADDR Reset Value
0	0x00000
1	0xFF800

5.2.4.4 LBC Local Access Window *n* Attributes Registers (LBLAWAR0–LBLAWAR3)

The LBC local access window *n* attributes registers (LBLAWAR0–LBLAWAR3) are shown in [Figure 5-5](#).



- 1 The LBLAWAR0[EN] reset value depends on the reset configuration word high values. See [Section 5.2.4.4.1, “LBLAWAR0\[EN\] and LBLAWAR0\[SIZE\] Reset Value,”](#) for a detailed description.
- 2 The LBLAWAR0[SIZE] reset value is always 0b010110, meaning an 8-Mbyte local access window. See [Section 5.2.4.4.1, “LBLAWAR0\[EN\] and LBLAWAR0\[SIZE\] Reset Value,”](#) for a detailed description.

Figure 5-5. LBC Local Access Window *n* Attributes Registers (LBLAWAR0–LBLAWAR3)

[Table 5-9](#) defines the bit fields of LBLAWAR0–LBLAWAR3.

Table 5-9. LBLAWAR0–LBLAWAR3 Bit Settings

Bits	Name	Description
0	EN	0 Local bus local access window <i>n</i> is disabled. 1 Local bus local access window <i>n</i> is enabled and other LBLAWAR0 and LBLAWBAR0 fields combine to identify an address range for this window.
1–25	—	Reserved. Write has no effect, read returns 0.
26–31	SIZE	Identifies the size of the window from the starting address. Window size is $2^{(SIZE+1)}$ bytes. 000000–001010 Reserved. Window is undefined. 001011 4 Kbytes 001100 8 Kbytes 001101 16 Kbytes $2^{(SIZE+1)}$ bytes 011110 2 Gbytes 011111–111111 Reserved. Window is undefined.

5.2.4.4.1 LBLAWAR0[EN] and LBLAWAR0[SIZE] Reset Value

The core may use a local bus peripheral device to fetch its boot vector. For this purpose an 8-Mbyte ($2^{(22+1)}$) local access window is defined by the LBLAWBAR0[SIZE] reset value, and LBLAWAR0 is enabled according to the value set in the reset configuration word high ROMLOC field.

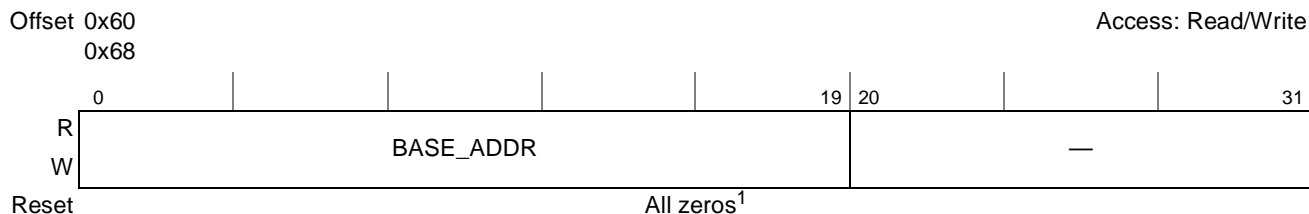
[Table 5-10](#) defines the reset value for LBLAWAR0[EN].

Table 5-10. LBLAWAR0[EN] Reset Value

RCWHR[ROMLOC]	LBLAWAR0[EN] Reset Value	Description
000–100	0	e300c1 core boot not performed from a local bus device.
101–111	1	e300c1 core boot performed from a local bus device. Local bus 8-Mbyte ($2^{(22+1)}$) local access window is enabled.

5.2.4.5 PCI Local Access Window *n* Base Address Register (PCILAWBAR0–PCILAWBAR1)

The PCI local access window *n* base address registers (PCILAWBAR0–PCILAWBAR1) are shown in Figure 5-6.



¹ The reset value of PCILAWBAR0[BASE_ADDR] depends on the reset configuration word high values. See Section 5.2.4.5.1, “PCILAWBAR0[BASE_ADDR] Reset Value,” for a detailed description.

Figure 5-6. PCI Local Access Window *n* Base Address Registers (PCILAWBAR0–PCILAWBAR1)

Table 5-11 defines the bit fields of PCILAWBAR0–PCILAWBAR1.

Table 5-11. PCILAWBAR0–PCILAWBAR1 Bit Settings

Bits	Name	Description
0–19	BASE_ADDR	Identifies the 20 most-significant address bits of the base of local access window <i>n</i> . The specified base address should be aligned to the window size, as defined by PCILAWAR n [SIZE].
20–31	—	Reserved. Write has no effect, read returns 0.

5.2.4.5.1 PCILAWBAR0[BASE_ADDR] Reset Value

The core may use a PCI peripheral device to fetch its boot vector. For this purpose, the PCILAWBAR0[BASE_ADDR] reset value is set according to the value set in the reset configuration word high BMS field.

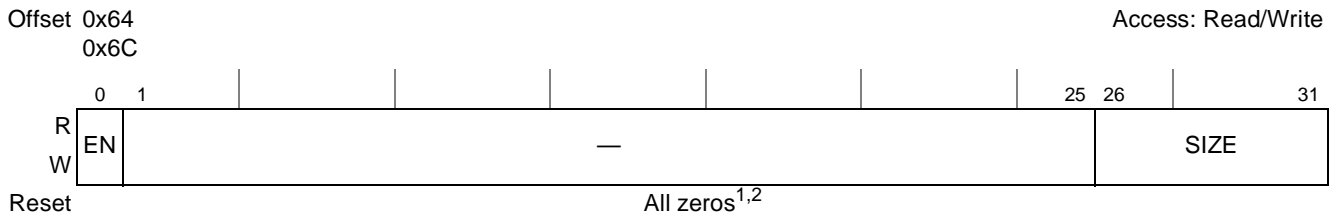
Table 5-12 defines the reset value of PCILAWBAR0[BASE_ADDR].

Table 5-12. PCILAWBAR0[BASE_ADDR] Reset Value

RCWHR[BMS]	PCILAWBAR0[BASE_ADDR] Reset Value
0	0x00000
1	0xFF800

5.2.4.6 PCI Local Access Window *n* Attributes Registers (PCILAWAR0–PCILAWAR1)

The PCI local access window *n* attributes registers (PCILAWAR0–PCILAWAR1) are shown in [Figure 5-7](#).



- ¹ The reset value of PCILAWAR0[EN] depends on the reset configuration word high values. See [Section 5.2.4.6.1, “PCILAWAR0\[EN\] and PCILAWAR0\[SIZE\] Reset Value,”](#) for a detailed description.
- ² The reset value of PCILAWAR0[SIZE] is always 0b010110, meaning an 8-Mbyte local access window. See [Section 5.2.4.6.1, “PCILAWAR0\[EN\] and PCILAWAR0\[SIZE\] Reset Value,”](#) for a detailed description

Figure 5-7. PCI Local Access Window *n* Attributes Registers (PCILAWAR0–PCILAWAR1)

[Table 5-13](#) defines the bit fields of PCILAWAR0–PCILAWAR1.

Table 5-13. PCILAWAR0–PCILAWAR1 Bit Settings

Bits	Name	Description
0	EN	0 The PCI local access window <i>n</i> is disabled. 1 The PCI local access window <i>n</i> is enabled and other PCILAWAR <i>n</i> and PCILAWBAR <i>n</i> fields combine to identify an address range for this window.
1–25	—	Reserved. Write has no effect, read returns 0.
26–31	SIZE	Identifies the size of the window from the starting address. Window size is $2^{(SIZE+1)}$ bytes. 000000–001010 Reserved. Window is undefined. 001011 4 Kbytes 001100 8 Kbytes 001101 16 Kbytes $2^{(SIZE+1)}$ bytes 011110 2 Gbytes 011111–111111 Reserved. Window is undefined.

5.2.4.6.1 PCILAWAR0[EN] and PCILAWAR0[SIZE] Reset Value

The core may use a PCI peripheral device to fetch its boot vector. For this purpose an 8-Mbyte ($2^{(22+1)}$) local access window is defined by the PCILAWBAR0[SIZE] reset value, and PCILAWAR0 is enabled according to the value set in the reset configuration word high ROMLOC field.

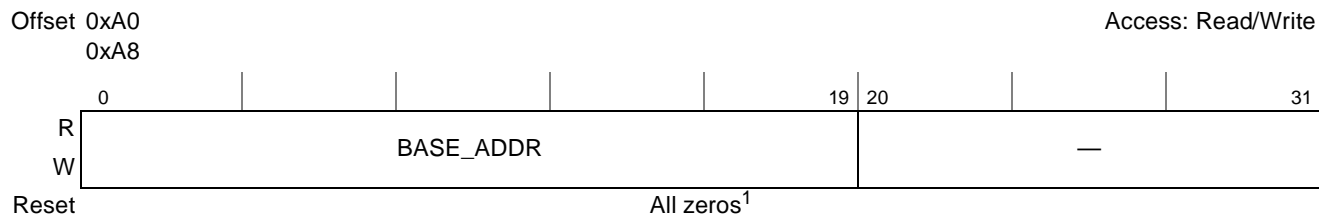
[Table 5-14](#) defines the reset value of PCILAWAR0[EN].

Table 5-14. PCILAWAR0[EN] Reset Value

RCWHR[ROMLOC]	PCILAWR0[EN] Reset Value	Description
000–001, 011–111	0	e300c1 core boot not performed from a PCI device.

5.2.4.7 DDR Local Access Window *n* Base Address Registers (DDRLAWBAR0–DDRLAWBAR1)

The DDR local access window *n* base address registers (DDRLAWBAR0–DDRLAWBAR1) are shown in Figure 5-8.



¹ The reset value of DDRLAWBAR0[BASE_ADDR] depends on the reset configuration word high values. See Section 5.2.4.7.1, “DDRLAWBAR0[BASE_ADDR] Reset Value,” for a detailed description

Figure 5-8. DDR Local Access Window *n* Base Address Registers (DDRLAWBAR0–DDRLAWBAR1)

Table 5-15 defines the bit fields of DDRLAWBAR0–DDRLAWBAR1.

Table 5-15. DDRLAWBAR0–DDRLAWBAR1 Bit Settings

Bits	Name	Description
0–19	BASE_ADDR	Identifies the 20 most-significant address bits of the base of local access window <i>n</i> . The specified base address should be aligned to the window size, as defined by DDRLAWAR n [SIZE].
20–31	—	Reserved. Write has no effect, read returns 0.

5.2.4.7.1 DDRLAWBAR0[BASE_ADDR] Reset Value

The core may use a DDR SDRAM device to fetch its boot vector. For this purpose, the DDRLAWBAR0[BASE_ADDR] reset value is set according to the value set in the reset configuration word high BMS field.

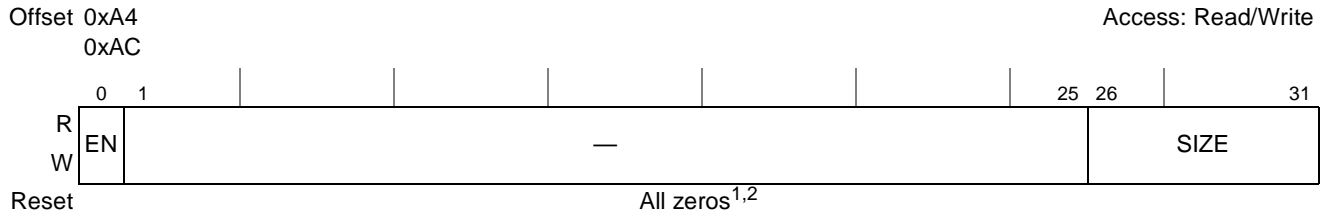
Table 5-16 defines the reset value DDRLAWBAR0.

Table 5-16. DDRLAWBAR0[BASE_ADDR] Reset Value

RCWHR[BMS]	DDRLAWBAR0[BASE_ADDR] Reset Value
0	0x00000
1	0xFF800

5.2.4.8 DDR Local Access Window n Attributes Registers (DDRLAWAR0–DDRLAWAR1)

The DDR local access window n attributes registers (DDRLAWAR0–DDRLAWAR1) are shown in Figure 5-9.



¹ The reset value of DDRLAWAR0[EN] depends on the reset configuration word high values. See Section 5.2.4.8.1, “DDRLAWAR0[EN] and DDRLAWAR0[SIZE] Reset Value,” for a detailed description.

² The reset value of DDRLAWAR0[SIZE] is always 0b010110, meaning an 8-Mbyte local access window. See Section 5.2.4.8.1, “DDRLAWAR0[EN] and DDRLAWAR0[SIZE] Reset Value,” for a detailed description.

Figure 5-9. DDR Local Access Window n Attributes Registers (DDRLAWAR0–DDRLAWAR1)

Table 5-17 defines the bit fields of DDRLAWAR0–DDRLAWAR1.

Table 5-17. DDRLAWAR0–DDRLAWAR1 Bit Settings

Bits	Name	Description
0	EN	0 The DDR local access window n is disabled. 1 The DDR local access window n is enabled and other DDRLAWAR n and DDRLAWBAR n fields combine to identify an address range for this window.
1–25	—	Reserved. Write has no effect, read returns 0.
26–31	SIZE	Identifies the size of the window from the starting address. Window size is $2^{(SIZE+1)}$ bytes. 000000–001010 Reserved. Window is undefined. 001011 4 Kbytes 001100 8 Kbytes 001101 16 Kbytes $2^{(SIZE+1)}$ bytes 011110 2 Gbytes 011111–111111 Reserved. Window is undefined.

5.2.4.8.1 DDRLAWAR0[EN] and DDRLAWAR0[SIZE] Reset Value

The core may use a DDR SDRAM device to fetch its boot vector. For this purpose an 8-Mbyte ($2^{(22+1)}$) local access window is defined by DDRLAWBAR0[SIZE] reset value, and DDRLAWAR0 is enabled according to the value set in the reset configuration word high ROMLOC field.

Table 5-18 defines the reset value DDRLAWAR0[EN] and DDRLAWAR0[SIZE].

Table 5-18. DDRLAWAR0[EN] Reset Value

RCWHR[ROMLOC]	DDRLAWAR0[EN] Reset Value	Description
000	1	e300c1 core boot performed from a DDR SDRAM device. DDR 8-Mbyte ($2^{(22+1)}$) local access window is enabled.
Else	0	e300c1 core boot not performed from a DDR SDRAM device.

5.2.4.9 Secondary DDR Local Access Window *n* Base Address Registers (SDDRLAWBAR0–SDDRLAWBAR1)

The secondary DDR local access window *n* base address registers (SDDRLAWBAR0–SDDRLAWBAR1) are shown in Figure 5-10.

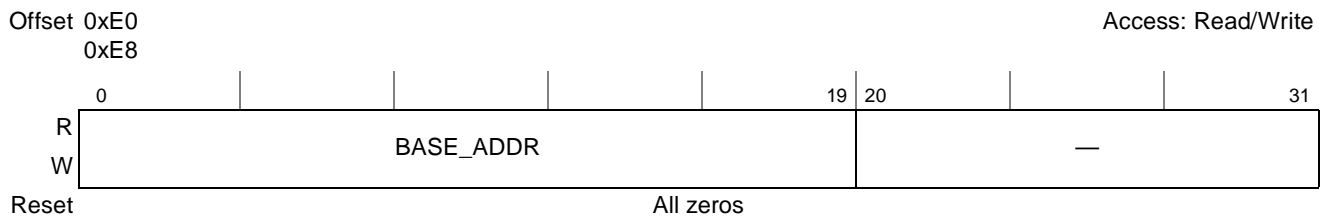


Figure 5-10. Secondary DDR Local Access Window *n* Base Address Registers (SDDRLAWBAR0–SDDRLAWBAR1)

Table 5-15 defines the bit fields of SDDRLAWBAR0-SDDRLAWBAR1.

Table 5-19. SDDRLAWBAR0–SDDRLAWBAR1 Bit Settings

Bits	Name	Description
0–19	BASE_ADDR	Identifies the 20 most-significant address bits of the base of local access window <i>n</i> . The specified base address should be aligned to the window size, as defined by SDDRLAWARn[SIZE].
20–31	—	Reserved. Write has no effect, read returns 0.

5.2.4.10 Secondary DDR Local Access Window *n* Attributes Registers (SDDRLAWAR0–SDDRLAWAR1)

The secondary DDR local access window *n* attributes registers (SDDRLAWAR0–SDDRLAWAR1) are shown in Figure 5-11.

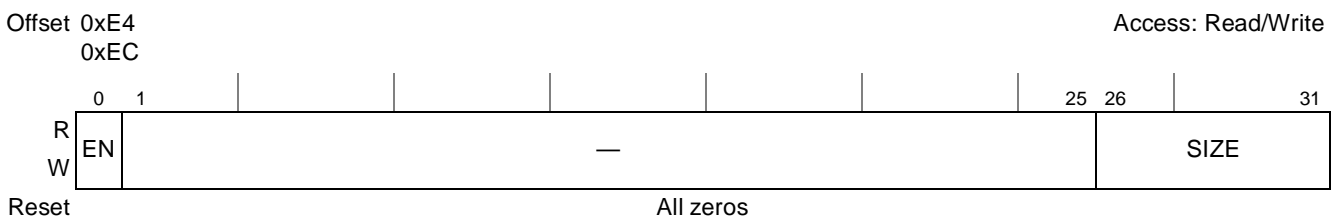


Figure 5-11. Secondary DDR Local Access Window *n* Attributes Registers (SDDRLAWAR0–SDDRLAWAR1)

Table 5-17 defines the bit fields of SDDRLAWAR0–SDDRLAWAR1.

Table 5-20. SDDRLAWAR0–SDDRLAWAR1 Bit Settings

Bits	Name	Description
0	EN	0 The secondary DDR local access window n is disabled. 1 The secondary DDR local access window n is enabled and other SDDRLAWARn and SDDRLAWBARn fields combine to identify an address range for this window.
1–25	—	Reserved. Write has no effect, read returns 0.
26–31	SIZE	Identifies the size of the window from the starting address. Window size is $2^{(SIZE+1)}$ bytes. 000000–001010 Reserved. Window is undefined. 001011 4 Kbytes 001100 8 Kbytes 001101 16 Kbytes $2^{(SIZE+1)}$ bytes 011110 2 Gbytes 011111–111111 Reserved. Window is undefined.

5.2.5 Precedence of Local Access Windows

If two local access windows overlap, the lower numbered window takes precedence (see Table 5-1 for window numbers). For instance, if two windows are set up as shown in Table 5-21, local access window 1 governs the mapping of the 1-Mbyte region from 0x7FF0_0000 to 0x7FFF_FFF, even though the window described in local access window 7 also encompasses that memory region.

Table 5-21. Overlapping Local Access Windows

Window	Base Address	Size	Target Interface
1	0x7FF0_0000	1 Mbyte	Local bus
7	All zeros	2 Gbytes	DDR SDRAM

5.2.6 Configuring Local Access Windows

After a local access window is enabled, it should not be modified while any device in the system may be using the window. Accordingly, a new window should not be used until the effect of the write to the window is visible to all blocks that use the window. This can be guaranteed by completing a read of the last local access window configuration register before enabling any other devices to use the window. For instance, if local bus local access windows 1–3 are being configured in order during the initialization process, the last write (to LBLAWAR3) should be followed by a read of LBLAWAR3 before any devices try to use any of these windows. If the configuration is being done by the local e300c1 core, the read of LBLAWAR3 should be followed by an **isync** instruction.

5.2.7 Distinguishing Local Access Windows from Other Mapping Functions

It is important to distinguish between the mapping function performed by the local access windows and the additional mapping functions that happen at the target interface. The local access windows define how a transaction is routed through the device internal interconnects from the transaction's source to its target.

Once the transaction has arrived at its target interface, that interface controller may perform additional mapping. For instance, the DDR SDRAM controller has chip select registers that map a memory request to a particular external device. The local bus controller has base registers that perform a similar function. The PCI interface has outbound address translation units that map the local address into an external address space.

These other mapping functions are configured by programming the configuration, control, and status registers of the individual interfaces. Note that there is no need to have a one-to-one correspondence between local access windows and chip select regions or outbound windows. A single local access window can be further decoded to any number of chip selects or to any number of outbound windows at the target interface.

5.2.8 Outbound Address Translation and Mapping Windows

Outbound address translation and mapping refers to the translation of addresses from the local 32-bit address space to the external address space and attributes of a particular I/O interface. On this device, the PCI block has an outbound address translation unit.

The PCI controller has six outbound windows plus a default window. See [Section 4.5, “Memory Map/Register Definitions,”](#) for a detailed description of the PCI outbound windows.

5.2.9 Inbound Address Translation and Mapping Windows

Inbound address translation and mapping refers to the translation of an address from the external address space of an I/O interface (such as PCI address space) to the local address space understood by the internal interfaces of this processor. It also refers to the mapping of transactions to a particular target interface and the assignment of transaction attributes. The PCI controller has inbound address translation unit.

5.2.9.1 PCI Inbound Windows

The PCI controller has three general inbound windows plus a dedicated window for memory mapped configuration accesses (PIMMR). These windows have a one-to-one correspondence with the base address registers in the PCI programming model. Updating one automatically updates the other. There is no default inbound window; if a PCI address does not match one of the inbound windows, this processor does not respond with an assertion of `PCI_DEVSEL`. See [Section 13.4.6, “PCI Inbound Address Translation,”](#) for a detailed description of the PCI inbound windows.

5.2.10 Internal Memory Map

All of the memory mapped configuration, control, and status registers in the device are contained within a 2-Mbyte address region, referred as the IMMR. To allow for flexibility, the internal memory map block can be relocated in the local address space. The local address map location of this register block is controlled by the internal memory map registers’ base address register (IMMRBAR); see [Section 5.2.4.1, “Internal Memory Map Registers Base Address Register \(IMMRBAR\).”](#) The default value for the IMMRBAR is `0xFF40_0000`.

NOTE

The internal memory map window is always the highest priority local access window.

5.2.11 Accessing Internal Memory from External Masters

In addition to being accessible by the e300 processor, the IMMR memory window is accessible from external interfaces. This allows external masters on the I/O ports to configure the device.

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

The PCI base address for accessing the local IMMR memory is selectable through the PCI internal memory map register (PIMMR), at offset 0x10, described in [Section 13.3.2.12, "PCI Inbound Base Address Registers \(PIBARn\)." When the device is a PCI agent, an external PCI master sets this register by running a PCI configuration cycle. Subsequent memory accesses by a PCI master to the PCI address range indicated by PIMMR are translated to the local address indicated by the current setting of IMMRBAR.](#)

5.3 QUICC Engine Secondary Bus Access Windows

The QUICC Engine block can access all the memory space defined by the local access windows like any other bus master of the coherent system bus (for example, processor, DMA). However the QUICC Engine block has also a dedicated local bus through which it can access devices connected to the local bus memory controller and DDR SDRAM devices connected to the secondary DDR memory controller. The accesses which are directed to the QUICC Engine secondary bus are not passing through the coherent system bus thus by using this dedicated local bus it is possible to get a better bus utilization on both buses. See TBD for details on usage of the dedicated local bus by the QUICC Engine block. The QUICC Engine secondary bus access windows define the actual interface (local bus memory controller or secondary DDR memory controller) that will handle bus transaction sent to the QUICC Engine dedicated local bus.

5.3.1 QUICC Engine Secondary Bus Access Windows Register Memory Map

[Table 5-22](#) shows the memory map for the QUICC Engine secondary bus access windows registers.

NOTE

It is not allowed to write to any of these registers while there are opened transactions on QUICC Engine block's local bus.

Table 5-22. QUICC Engine Secondary Bus Access Windows Register Memory Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x00000	Local bus memory controller start address register (LBMCSAR)	R/W	All zeros	5.3.2.1/55-18
0x00004	Secondary DDR memory controller start address register (SDMCSAR)	R/W	All zeros	5.3.2.2/55-19
0x00008–0x0003C	Reserved			
0x00040	Local bus memory controller end address register (LBMCEAR)	R/W	All zeros	5.3.2.3/55-19
0x00044	Secondary DDR memory controller end address register (SDMCEAR)	R/W	All zeros	5.3.2.4/55-20
0x00048–0x0007C	Reserved			
0x00080	Local bus memory controller attributes register (LBMCAR)	R/W	All zeros	5.3.2.5/55-20
0x00084	Secondary DDR memory controller attributes register (SDMCAR)	R/W	All zeros	5.3.2.6/55-21
0x00088–0x0007FC	Reserved			

5.3.2 QUICC Engine Secondary Bus Access Windows Registers

5.3.2.1 Local Bus Memory Controller Start Address Register (LBMCSAR)

The local bus memory controller start address register, shown in [Figure 5-12](#), defines the start address of the memory region which is assigned to the local bus memory controller in the QUICC Engine block’s local bus. To ensure proper operation do not modify this register while the specific window is enabled. (first clear WEN bit in the LBMCAR register).

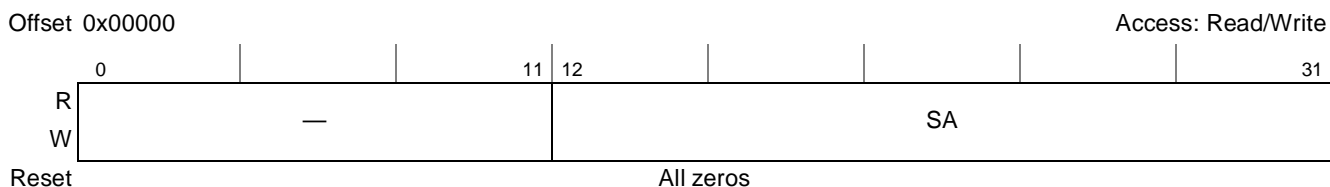


Figure 5-12. Local Bus Memory Controller Start Address Register (LBMCSAR)

[Table 5-34](#) defines the bit fields of LBMCSAR.

Table 5-23. LBMCSAR Bit Settings

Bits	Name	Description
0–11	Reserved	Reserved should be cleared.
12–31	SA	This field contains the 20 most-significant bits of the start address of the local bus memory controller window. The 12 least-significant bits are all zeros.

5.3.2.2 Secondary DDR Memory Controller Start Address Register (SDMCSAR)

The secondary DDR memory controller start address register, shown in [Figure 5-13](#), defines the start address of the memory region which is assigned to the secondary DDR memory controller in QUICC Engine block's local bus. To ensure proper operation do not modify this register while the specific window is enabled. (First clear SDMCAR[WEN].)

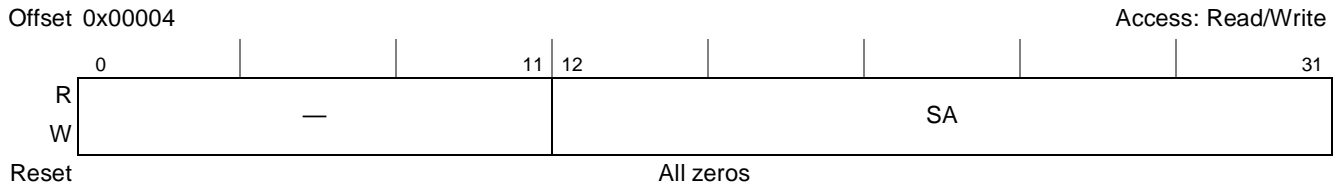


Figure 5-13. Secondary DDR Memory Controller Start Address Register (SDMCSAR)

[Table 5-34](#) defines the bit fields of SDMCSAR.

Table 5-24. SDMCSAR Bit Settings

Bits	Name	Description
0–11	Reserved	Reserved should be cleared.
12–31	SA	This field contains the 20 most-significant bits of the start address of secondary DDR memory controller window. The 12 least-significant bits are all zeros.

5.3.2.3 Local Bus Memory Controller End Address Register (LBMCEAR)

The local bus memory controller end address register, shown in [Figure 5-14](#), defines the end address of the memory region which is assigned to the local bus memory controller in the QUICC Engine block's local bus. To ensure proper operation do not modify this register while the specific window is enabled. (First clear LBMCAR[WEN].)

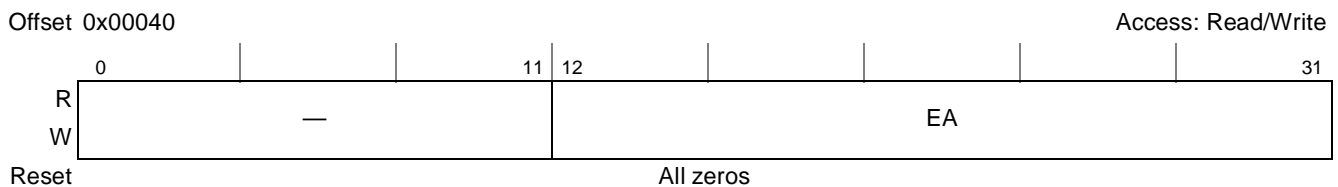


Figure 5-14. Local Bus Memory Controller End Address Register (LBMCEAR)

[Table 5-34](#) defines the bit fields of LBMCEAR.

Table 5-25. LBMCEAR Bit Settings

Bits	Name	Description
0–11	Reserved	Reserved should be cleared.
12–31	EA	This field contains the 20 most-significant bits of the end address of local bus memory controller window. The 12 least-significant bits are all zeros. When programming the end address, the user is required to make sure it is greater than or equal to the start address of the same window. If the end address is equal to the start address of this window, then the address window is of 4 Kbytes in size.

5.3.2.4 Secondary DDR Memory Controller End Address Register (SDMCEAR)

The secondary DDR memory controller end address register, shown in [Figure 5-15](#), defines the end address of the memory region which is assigned to the secondary DDR memory controller in the QUICC Engine block’s local bus. To ensure proper operation do not modify this register while the specific window is enabled. (First clear SDMCAR[WEN].)



Figure 5-15. Secondary DDR Memory Controller End Address Register (SDMCEAR)

[Table 5-34](#) defines the bit fields of SDMCEAR.

Table 5-26. SDMCEAR Bit Settings

Bits	Name	Description
0–11	Reserved	Reserved should be cleared.
12–31	EA	This field contains the 20 most-significant bits of the end address of secondary DDR memory controller window. The 12 least-significant bits are all zeros. When programming the end address, the user is required to make sure it is greater than or equal to the start address of the same window. If the end address is equal to the start address of this window, then the address window is 4 Kbytes in size.

5.3.2.5 Local Bus Memory Controller Attributes Register (LBMCAR)

The local bus memory controller attributes register, shown in [Figure 5-16](#), defines the attributes of the memory region which is assigned to the local bus memory controller in the QUICC Engine block’s local bus.



Figure 5-16. Local Bus Memory Controller Attributes Register (LBMCAR)

[Table 5-34](#) defines the bit fields of LBMCAR.

Table 5-27. LBMCAR Bit Settings

Bits	Name	Description
0–30	Reserved	Reserved should be cleared.
31	WEN	0 Transaction on the QUICC Engine block’s local bus will not be forwarded to the local bus memory controller. 1 Transaction on the QUICC Engine block’s local bus which hits in the memory region defined by LBMCSAR and LBMCEAR is forwarded to the local bus memory controller.

5.3.2.6 Secondary DDR Memory Controller Attributes Register (SDMCAR)

The secondary DDR memory controller attributes register shown in [Figure 5-17](#) defines the attributes of the memory region which is assigned to the secondary DDR memory controller in QUICC Engine block's local bus.

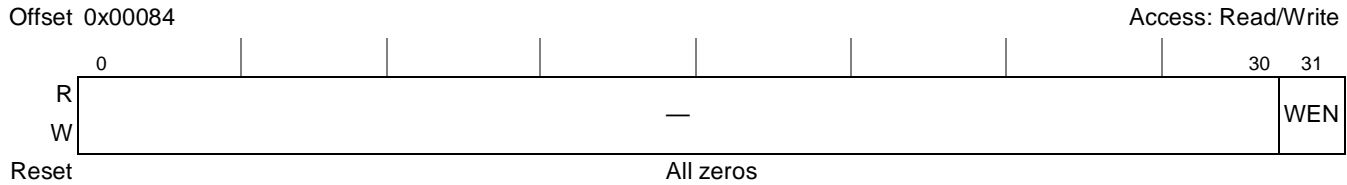


Figure 5-17. Secondary DDR Memory Controller Attributes Register (SDMCAR)

[Table 5-34](#) defines the bit fields of SDMCAR.

Table 5-28. SDMCAR Bit Settings

Bits	Name	Description
0–30	Reserved	Reserved should be cleared.
31	WEN	0 Transaction on QUICC Engine block's local bus will not be forwarded to the secondary DDR memory controller. 1 Transaction on QUICC Engine block's local bus which hits in the memory region defined by SDMC SAR and SDMC EAR is forwarded to the secondary DDR memory controller.

5.3.3 QUICC Engine Secondary Bus Windows Operation Description

The QUICC Engine secondary bus windows manage the forwarding of transactions from the QUICC Engine block's local bus to a specific interface connected to this bus. Two specific interfaces are connected to the QUICC Engine block's local bus: the local bus memory controller and the secondary DDR SDRAM memory controller. Note that these two interfaces are accessible from the coherent system bus too through the local access windows (see [Section 5.2, “Local Memory Map Overview and Example”](#)).

Two QUICC Engine secondary bus windows are available. Each of these windows is associated to one of the interfaces and defines a certain memory region. The address of each transaction on the QUICC Engine block's local bus is compared to the memory regions defined by the QUICC Engine secondary bus windows and forwarded to the region which contains this address. In case there is no matching region a transfer error is reported to the QUICC Engine block. In case there is an overlap between the two memory regions, transactions hitting in the overlapping region is forwarded to the secondary DDR memory controller. Thus the secondary DDR memory controller has higher priority. This feature can be used to generate a secondary DDR memory controller region in the middle of a large local bus memory controller region as described in [Section 5.3.4, “QUICC Engine Secondary Bus Windows Example.”](#)

The local bus memory controller and the secondary DDR memory controller are both accessible from the coherent system bus (through local access windows) and from the QUICC Engine block's local bus (through QUICC Engine secondary bus windows). It is theoretically possible to allow access to one of these interfaces only from a certain bus. It is the users responsibility to correctly configure the two window mechanisms according to his system's needs.

5.3.4 QUICC Engine Secondary Bus Windows Example

Figure 5-18 shows an example memory map and the usage of QUICC Engine secondary bus windows.

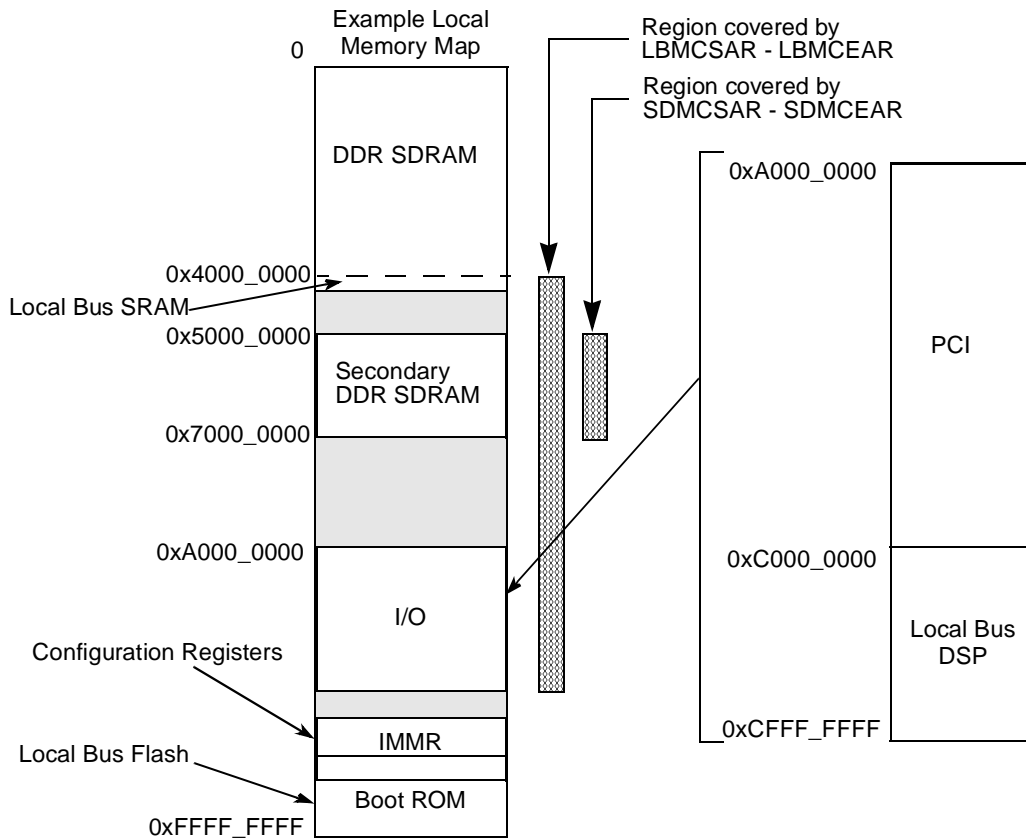


Figure 5-18. Local Memory Map Example

Table 5-30 shows one corresponding set of local access window settings.

Table 5-29. Local Access Windows Example

Window	Base Address	Size	Target Interface
7	All zeros	1 Gbyte	DDR SDRAM
2	0x4000_0000	1 Mbyte	Local bus (SRAM)
9	0x5000_0000	256 Mbyte	Secondary DDR SDRAM
5	0xA000_0000	512 Mbytes	PCI
3	0xC000_0000	256 Mbytes	Local bus (DSP)
0	0xFF40_0000	2 Mbytes	Configuration registers (IMMR)
1	0xFF80_0000	8 Mbytes	Local bus boot ROM Flash
4,6,8,10	Unused		

Table 5-30 shows the settings of QUICC Engine secondary bus windows.

Table 5-30. QUICC Engine Secondary Bus Windows Settings

Window	Value in Start Address Register	Value in End Address Register
Local bus memory controller	0x4000_0000	0xCFFF_F000
Secondary DDR memory controller	0x5000_0000	0x5FFF_F000

In this example the system includes several interfaces/devices: DDR SDRAM on the coherent system bus, DDR SDRAM on local bus, SRAM, PCI, DSP device, Boot ROM Flash and the configuration registers (IMMR). All these devices are accessible through the coherent system bus. A subset of all these devices is accessible by the QUICC Engine block's local bus: DDR SDRAM on local bus, SRAM and the DSP device. Note that the PCI is not accessible by the QUICC Engine secondary bus although it is within the region of local bus memory controller window. This is because the PCI is connected to the coherent system bus and not as one of the banks of the local bus memory controller. Thus, to access the PCI interface the QUICC Engine block should initiate the transaction on its coherent system bus. Note that transactions hitting in the region 0x5000_0000–0x5FFF_FFFF is directed to the secondary DDR memory controller although the window of the local bus memory controller is overlapping on this region. This is due to the precedence of secondary DDR memory controller window which is used intentionally in this example.

5.4 System Configuration

The following sections describe some general information and configuration options that affect system behavior and performance.

5.4.1 External Signal Description

Section 5.4.1.1, “[PCI_MODE Signal](#),” describes the MPC8360E external signal that affects the system configuration.

5.4.1.1 $\overline{\text{PCI_MODE}}$ Signal

One signal affects the system configuration of the MPC8360E. [Table 5-31](#) lists this signal.

Table 5-31. System Configuration Signal Properties

Name	Port	Function	I/O	Reset	Pull Up
PCI_MODE	$\overline{\text{PCI_MODE}}$	Global enable for PCI external I/O signals	I	N/A	—

Table 5-32 provides a detailed description of the external MPC8360E system configuration signal.

Table 5-32. System Configuration Signal—Detailed Description

Signal	I/O	Description
PCI_MODE	I	A group of QUICC Engine parallel I/O ports (CE_PF[3:29] & CE_PG[0:31]) may function as PCI signals. The PCI_MODE signal enables PCI mode for these signals. If PCI is not used, the device clock source must be connected to PCI_CLK/PCI_SYNC_IN, not CLKIN.
		State Meaning <ul style="list-style-type: none"> Asserted—QUICC Engine parallel I/O ports CE_PF[3:29] & CE_PG[0:31] function as PCI signals. For CE_PF[26:28] which may function as PCI_CLK_OUT[0:2], the PCI function is selected with the reset configuration bit RCWH[PCICKDRV] (in addition to assertion of the PCI_MODE signal). See Section 4.3.2.2, “Reset Configuration Word High Register (RCWHR).” Assertion of PCI_MODE during PORESET flow affects the reset value of QUICC Engine parallel I/O ports configuration registers and selects the PCI function. In addition, the PCI function is selected during normal operation regardless of the values in these configuration registers. Thus the user may change the values in the configuration registers, but the PCI function will still be selected as long as PCI_MODE is asserted. Negated—QUICC Engine parallel I/O ports CE_PF[3:29] & CE_PG[0:31] function according to the values in their configuration registers (this may be PCI or non-PCI functionality.)
		Timing <p>This input signal is sampled during the assertion of PORESET, after a stable clock is supplied (PORESET flow). During normal operation, the state of this signal affects the QUICC Engine ports functionality asynchronously.</p>

5.4.2 System Configuration Register Memory Map

Table 5-33 shows the memory map for the system configuration registers.

Table 5-33. System Configuration Register Memory Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x00100	System general purpose register low (SGPRL)	R/W	All zeros	5.4.3.1/5-25
0x00104	System general purpose register high (SGPRH)	R/W	All zeros	5.4.3.2/5-25
0x00108	System part and revision ID register (SPRIDR)	R		5.4.3.3/5-26
0x0010C	Reserved	—	—	—
0x00110	System priority configuration register (SPCR)	R/W	All zeros	5.4.3.4/5-27
0x00114	System I/O configuration register low (SICRL)	R/W	0x4000_0000 ¹	5.4.3.5/5-28
0x00118	System I/O configuration register high (SICRH)	R/W	All zeros	5.4.3.6/5-30
0x0011C–0x00124	Reserved	—	—	—
0x00128	DDR control driver register (DDRCDR)	R/W	0x7304_00001	5.4.3.8/5-33
0x0012C	DDR debug status register (DDRDSR)	R	All zeros	5.4.3.9/5-35

¹ Depends on the reset configuration word high configuration values.

5.4.3 System Configuration Registers

5.4.3.1 System General Purpose Register Low (SGPRL)

The system general purpose register low (SGPRL), shown in [Figure 5-19](#), can be used by software for any purpose. The values set in this register have no effect on the internal hardware.

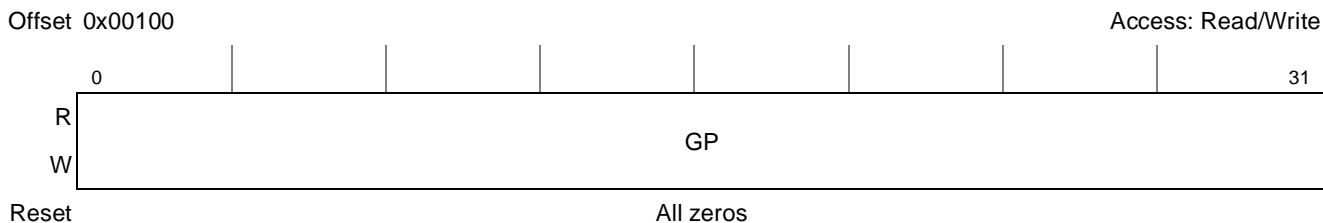


Figure 5-19. System General Purpose Register Low (SGPRL)

[Table 5-34](#) defines the bit fields of SGPRL.

Table 5-34. SGPRL Bit Settings

Bits	Name	Description
0–31	GP	General purpose

5.4.3.2 System General Purpose Register High (SGPRH)

The system general purpose register high (SGPRH), shown in [Figure 5-20](#), can be used by software for any purpose. The values set in this register have no effect on the internal hardware.

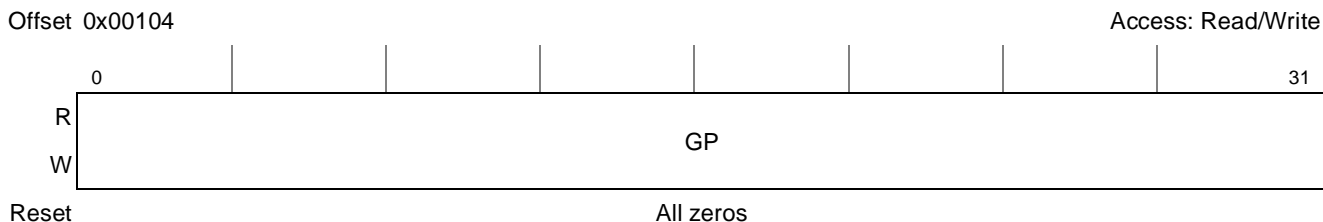


Figure 5-20. System General Purpose Register High (SGPRH)

[Table 5-35](#) defines the bit fields of SGPRH.

Table 5-35. SGPRH Bit Settings

Bits	Name	Description
0–31	GP	General purpose

5.4.3.3 System Part and Revision ID Register (SPRIDR)

SPRIDR, shown in [Figure 5-21](#), provides information about the device and revision numbers.

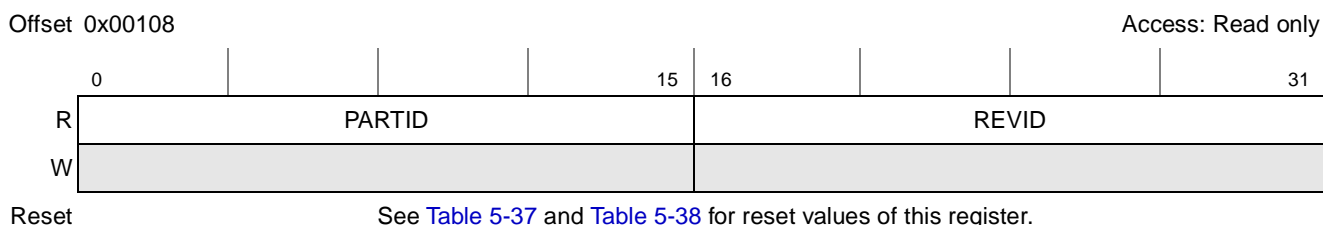


Figure 5-21. System Part and Revision ID Register (SPRIDR)

[Table 5-36](#) defines the bit fields of SPRIDR.

Table 5-36. SPRIDR Bit Settings

Bits	Name	Description
0–15	PARTID	Part identification. This read-only field is mask-programmed with a code corresponding to the device number. It is intended to help factory test and user code that is sensitive to device changes. The device number changes according to manufacturing considerations. See Table 5-37 for values of this field.
16–31	REVID	Revision identification. This read-only field is mask-programmed with a code corresponding to the revision number of the part defined in PARTID field. It is intended to help factory test and user code that is sensitive to device changes. The mask number is programmed in a commonly changed layer, and changes with each mask set change. See Table 5-38 for values of this field.

5.4.3.3.1 SPRIDR[PARTID] Coding

[Table 5-37](#) defines the reset values of SPRIDR[PARTID].

Table 5-37. PARTID Coding

PARTID	Device Name	Package Type
0x8048	MPC8360E	TBGA
0x8049	MPC8360	TBGA
0x804A	MPC8358E	TBGA
0x804B	MPC8358	TBGA
0x804E	MPC8358E	PBGA
0x804F	MPC8358	PBGA

[Table 5-38](#) defines the reset values of SPRIDR[REVID].

Table 5-38. REVID Coding

REVID	Device Revision
0x0010	1.0
0x0011	1.1

Table 5-38. REVID Coding (continued)

REVID	Device Revision
0x0012	1.2
0x0020	2.0

5.4.3.4 System Priority and Configuration Register (SPCR)

The system priority and configuration register (SPCR), shown in [Figure 5-22](#), controls the priority of requests for transactions on the internal system bus. This priority is considered by the system arbiter whenever an internal unit requests mastership of the coherent system bus (CSB). The SPCR also includes some other control functions.

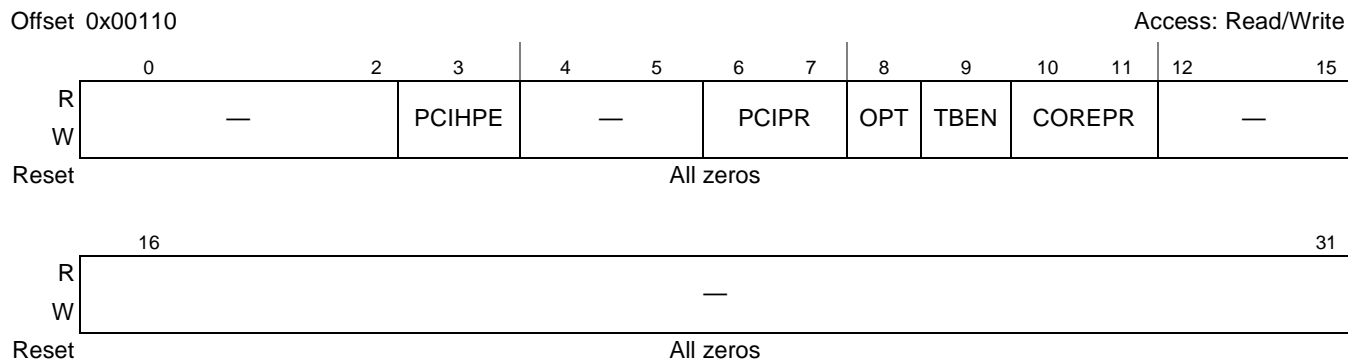


Figure 5-22. System Priority Configuration Register (SPCR)

[Table 5-39](#) defines the bit fields of SPCR.

Table 5-39. SPCR Bit Settings

Bits	Name	Description
0–2	—	Reserved. Should be cleared.
3	PCIHPE	PCI highest priority enable. If this bit is set, the PCI bridge is permitted to request the coherent system bus (CSB) with highest priority, regardless of SPCR[PCIPR] value, when it needs to complete a posted write transaction from an external PCI master. To follow PCI ordering rules specifications, the PCI bridge must flush any outstanding write transactions before it can start a new read transaction. Setting this bit allows faster flushing of the outstanding write transactions coming from the PCI bus onto the CSB and to the device targets, such as DDR SDRAM and local bus memories.
4–5	—	Reserved. Should be cleared.
6–7	PCIPR	PCI bridge CSB request priority. The level of priority can be chosen from 4 possible levels. 00 Level 0 (lowest priority) 01 Level 1 10 Level 2 11 Level 3 (highest priority) Note: DMA has the same priority as PCI.

Table 5-39. SPCR Bit Settings (continued)

Bits	Name	Description
8	OPT	Optimize. Setting this bit may enhance the performance of transactions issued to the internal coherent system bus (CSB) by the security engine (SEC) and the QUICC Engine block. Performance is enhanced by reading more bytes on the bus than actually needed by the master in the case that this is more efficient. The user may set this bit only if it is known that QUICC /Engine SEC transactions sent to the internal CSB are not accessing devices in which speculative reads may change the state of the device (for example, FIFOs in which reading a byte may advance some internal counter). 0 No performance enhancement. 1 Performance enhancement by speculative reading is enabled.
9	TBEN	e300c1 core time base unit enable 0 Time base unit is disabled. 1 Time base unit is enabled.
10–11	COREPR	e300c1 core CSB request priority. The priority level for the core in accessing the CSB can be chosen from 4 possible levels. 00 Level 0 (lowest priority) 01 Level 1 10 Level 2 11 Level 3 (highest priority)
12–31	—	Reserved. Should be cleared.

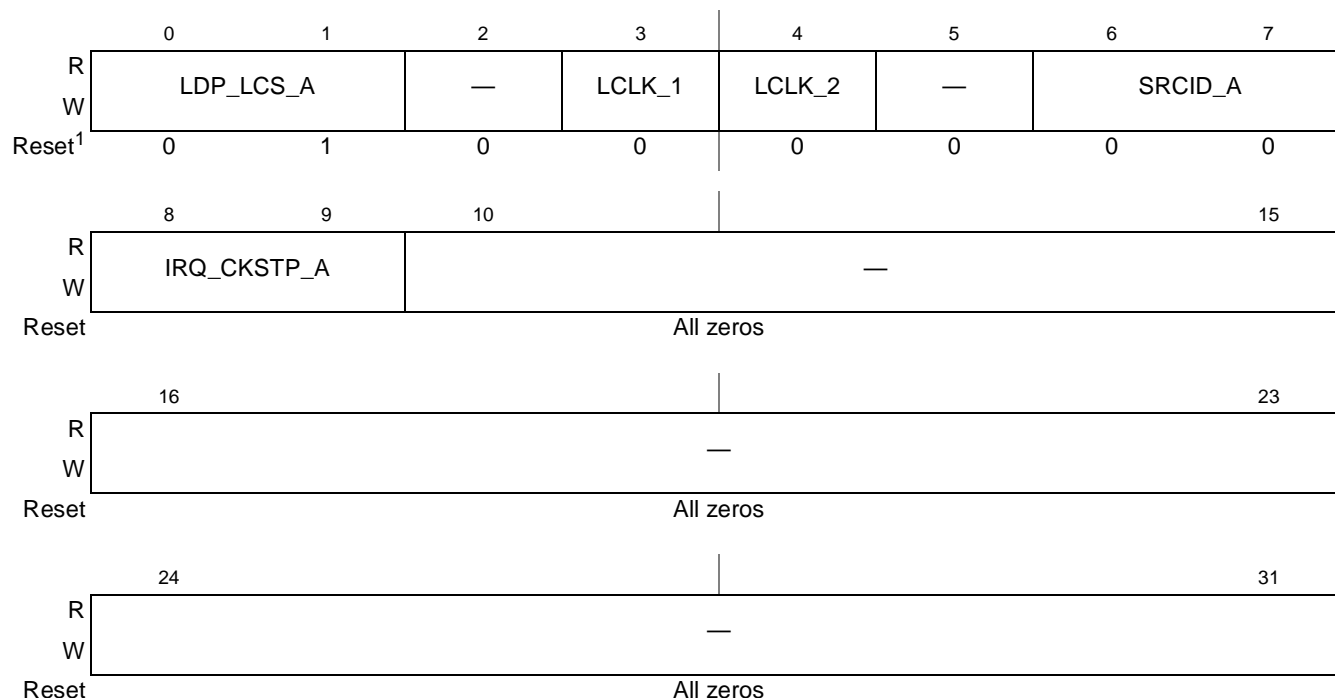
5.4.3.5 System I/O Configuration Register Low (SICRL)

The system I/O configuration register low (SICRL) controls the multiplexing of some of the device I/O pins. Each bit or set of bits in the SICRL selects which function is used by a certain group of the device pins.

Figure 5-23 shows SICRL.

Offset 0x00114

Access: Read/Write



¹ The reset value of SICRL[0] depends on the value of RCWH[30] which is loaded into the reset configuration word.

Figure 5-23. System I/O Configuration Register Low (SICRL)

Table 5-40 defines the bit fields of SICRL. Each Pin Function column lists the name of the multi-function pin used in this option. Some groups have only two options (shown as Pin Function 0 and Pin Function 1) and therefore, only one control bit. In this case they can only have a value of 0b0 or 0b1. Other groups may have four options (shown as Pin Function 0, Pin Function 1, Pin Function 2, and Pin Function 3) and therefore, two control bits. In this case they can have a value of 0b00, 0b01, 0b10, or 0b11. Use the notations ‘0bN’ or ‘0bNN’ according to whether a group has one or two control bits, respectively.

Table 5-40. SICRL Bit Settings

SICRL[Bits] Value		0b0/0b00	0b1/0b01	0b10	0b11
Bits	Group	Pin Function 0	Pin Function 1	Pin Function 2	Pin Function 3
0-1	LDP_LCS_A	$\overline{\text{CKSTOP_OUT}}$	LDP[0]	N/A	N/A
		$\overline{\text{CKSTOP_IN}}$	LDP[1]	N/A	N/A
		$\overline{\text{LCS}}[6]$	LDP[2]	$\overline{\text{LCS}}[6]$	N/A
		$\overline{\text{LCS}}[7]$	LDP[3]	$\overline{\text{LCS}}[7]$	N/A
2	Reserved	—	—	—	—
3	LCLK_1	LCLK1	$\overline{\text{LCS}}[6]$	N/A	N/A
4	LCLK_2	LCLK2	$\overline{\text{LCS}}[7]$	N/A	N/A

Table 5-40. SICRL Bit Settings (continued)

SICRL[Bits] Value		0b0/0b00	0b1/0b01	0b10	0b11
Bits	Group	Pin Function 0	Pin Function 1	Pin Function 2	Pin Function 3
5	Reserved	—	—	—	—
6–7	SRCID_A	UART1_SOUT	M1SRCID0	M2SRCID0	LSRCID0
		UART1_SIN	M1SRCID1	M2SRCID1	LSRCID1
		$\overline{\text{UART1_CTS}}$	M1SRCID2	M2SRCID2	LSRCID2
		$\overline{\text{UART1_RTS}}$	M1SRCID3	M2SRCID3	LSRCID3
		$\overline{\text{IRQ1}}$	M1SRCID4	M2SRCID4	LSRCID4
		$\overline{\text{IRQ2}}$	M1DVAL	M2DVAL	LDVAL
8–9	IRQ_CHKSTP_A	$\overline{\text{IRQ6}}$	N/A	$\overline{\text{LCS}}[6]$	$\overline{\text{CKSTOP_OUT}}$
		$\overline{\text{IRQ7}}$	N/A	$\overline{\text{LCS}}[7]$	$\overline{\text{CKSTOP_IN}}$
10–31	Reserved	—	—	—	—

5.4.3.6 System I/O Configuration Register High (SICRH)

The system I/O configuration register high, shown in [Figure 5-24](#), controls the multiplexing of the rest of the device I/O pins. Each bit or set of bits in this register select which function is used by a certain group of the device pins. The reset value of bit 2 in this register (SDDRIOE group) depends on the SDDRIOE field setting in the reset configuration word high.

Offset 0x00118

Access: Read/Write

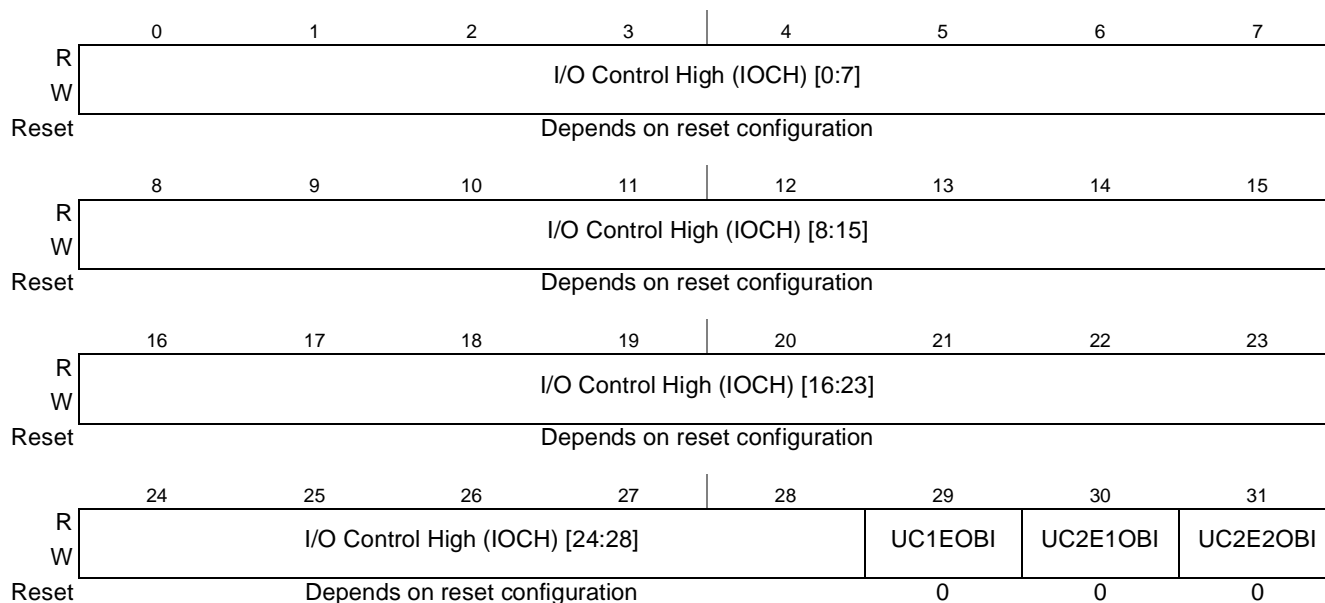


Figure 5-24. System I/O Configuration Register High (SICRH)

Table 5-41 defines the bit fields of SICRH. Each Pin Function column lists the name of the multi-function pin used in this option. Some groups have only two options (shown as Pin Function 0 and Pin Function 1) and therefore, only one control bit. In this case they can have the value of 0b0 or 0b1 only. Other groups may have three options (shown as Pin Function 0, Pin Function 1, and Pin Function 2) and therefore, two control bits. In this case they can have the value of 0b00, 0b01, or 0b10. A value of 0b11 selects GPIO mode of the appropriate pin. Use the 0bN or 0bNN notations according to a group having one or two control bits, respectively.

Table 5-41 defines the bit fields of SICRH. Each Pin Function column lists the name of the multi-function pin used in this option. Each group has two options (shown as Pin Function 0 and Pin Function 1) and therefore one control bit, that can have the value of 0 or 1 only.

Note that bits 29, 30 and 31 (UC1EOBI, UC2E1OBI and UC2E2OBI), which control UCC Ethernet output buffer impedance, are described in Table 5-42.

Table 5-41. SICRH Bit Settings

SICRH[Bits] Value:		0b0	0b1	Reset Value
Bits	Group	Pin Function 0	Pin Function 1	
0	DDR	MEMC1_MECC[0]	MEMC1_MSRCID[0]	0
		MEMC1_MECC[1]	MEMC1_MSRCID[1]	
		MEMC1_MECC[2]	MEMC1_MSRCID[2]	
		MEMC1_MECC[3]	MEMC1_MSRCID[3]	
		MEMC1_MECC[4]	MEMC1_MSRCID[4]	
		MEMC1_MECC[5]	MEMC1_MDVAL	
1	Secondary DDR	MEMC2_MECC[0]	MEMC2_MSRCID[0]	0
		MEMC2_MECC[1]	MEMC2_MSRCID[1]	
		MEMC2_MECC[2]	MEMC2_MSRCID[2]	
		MEMC2_MECC[3]	MEMC2_MSRCID[3]	
		MEMC2_MECC[4]	MEMC2_MSRCID[4]	
		MEMC2_MECC[5]	MEMC2_MDVAL	
2	SDDRIOE	MEMC1_MDQ[32:63]	MEMC2_MDQ[0:31]	SDDRIOE bit from reset configuration word high.
3	IRQ3	$\overline{\text{IRQ}}[3]$	$\overline{\text{CORE_SRESET}}$	0
4–29	Reserved	—	—	—

Table 5-42. SICRH[29–31] Bit Settings

Bits	Name	Description	Reset Value
29	UC1EOBI	UCC1 Ethernet output buffer impedance. This bit controls the output buffer impedance of UCC1 output signals, used for reduced pin mode interface (RTBI/RGMII). The output buffer impedance should be correlated to the voltage supplied to the UCC1 I/O pins (LVDD0). 0 Output buffer is set for 40 ohm, 3.3 volts. 1 Output buffer is set for 40 ohm, 2.5 volts.	0
30	UC2E1OBI	UCC2 Ethernet pin option 1 output buffer impedance. This bit controls the output buffer impedance of 'UCC2 pin option 1' output signals, used for reduced pin mode interface (RTBI/RGMII). The output buffer impedance should be correlated to the voltage supplied to the UCC2 I/O pins (LVDD1). 0 Output buffer is set for 40 ohm, 3.3 volts. 1 Output buffer is set for 40 ohm, 2.5 volts.	0
31	UC2E2OBI	UCC2 Ethernet pin option 2 output buffer impedance. This bit controls the output buffer impedance of 'UCC2 pin option 2' output signals, used for reduced pin mode interface (RTBI/RGMII). The output buffer impedance should be correlated to the voltage supplied to the UCC2 I/O pins (LVDD2). 0 Output buffer is set for 40 ohm, 3.3 volts. 1 Output buffer is set for 40 ohm, 2.5 volts.	0

5.4.3.7 Debug Configuration

Debug information may be driven on the device pins. This information can identify the internal source of a transaction that reached the (system) DDR SDRAM, secondary DDR SDRAM or local bus interfaces. The device can be configured to drive the MEMC1_MSRCID[0:4], MEMC1_MDVAL, MEMC2_MSRCID[0:4], MEMC2_MDVAL and LSRCID[0:4], and LDVAL signals, respectively on other device pins. The coding of the source ID debug information is the same as the coding of the MSTR_ID field in the AEATR register of the arbiter (See [Section 6.2.6, “Arbiter Event Attributes Register \(AEATR\)”](#)).

5.4.3.7.1 DDR Debug Configuration

The DDR debug configuration enables a DDR memory controller to enter debug mode in which the DDR SDRAM source ID field and data valid strobe are driven onto two optional set of pins:

- ECC pins. ECC checking and generation are disabled in this case. ECC signals driven from the SDRAMs must be electrically disconnected from the ECC I/O pins of the processor in this mode. Set SICRH[0] to select this mode.
- UART pins. UART operation is disabled, and any signals driven by UART devices must be electrically disconnected from the UART I/O pins. Set SICRL[6–7] to 0b01 to select this mode.

5.4.3.7.2 Secondary DDR Debug Configuration

The secondary DDR debug configuration enables a secondary DDR memory controller debug mode in which the secondary DDR SDRAM source ID field and data valid strobe are driven onto one of two optional set of pins:

1. ECC pins. ECC checking and generation are disabled in this case. ECC signals driven from the secondary SDRAMs must be electrically disconnected from the ECC I/O pins of the MPC8360E in this mode. Set SICRH[1] to select this mode.
2. UART pins. UART operation is disabled, and any signals driven by UART devices must be electrically disconnected from the UART I/O pins. Set SICRL[6–7] to 0b10 to select this mode.

5.4.3.7.3 Local Bus Debug Configuration

The local bus debug configuration enables a LBC debug mode in which the SDRAM source ID field and data valid strobe for LBC memory accesses are driven onto UART pins. UART operation must be disabled, and any signals driven by UART devices must be electrically disconnected from the UART I/O pins in this case. Set SICRL[6–7] to 0b11 to select this mode.

5.4.3.8 DDR Control Driver Register (DDRCDR)

The DDR control driver register (DDRCDR) contains bits that allow control over the driver of the DDR SDRAM controller. Note that the MDIC signals require the use of precision 18.2- Ω resistors; MDIC0 should be pulled to GND, while MDIC1 should be pulled to GV_{DD} .

The fields in DDRCDR (other than ODT) are used to enable driver calibration with the MDIC[0:1] signals. This can be used to calibrate the DDR drivers to 18.2 Ω . However, this should only be used for full-strength driver applications. If half strength is desired, then this calibration should remain disabled.

Hardware DDR driver calibration is enabled using DDRCDR[DHC_EN]. If hardware calibration is used, then it should be set before DDR_SDRAM_CFG[MEM_EN] is set.

Software can be used to calibrate the drivers instead of the automatic hardware calibration. If software calibration is used, the following steps should be taken:

1. Set DDRCDR[DSO_EN] and ensure that DDRCDR[DHC_EN] is cleared.
2. Set the highest impedance (value 0000) for DDRCDR[DSO_PZ].
3. Set DDRCDR[MDIC0_OE] to enable the output enable for MDIC[0].
4. After at least 4 cycles, read DDRDSR[0]. If the value is 0, then use the next lower impedance, and read DDRDSR[0] again. Once a value of 1 is detected, then leave DDRCDR[DSO_PZ] at the calibrated value.
5. Clear DDRCDR[MDIC0_OE].
6. After DDRCDR[DSO_PZ] is calibrated, set a value of 0000 for DDRCDR[DSO_NZ].
7. Set DDRCDR[MDIC1_OE] to enable the output enable for MDIC[1].
8. After at least 4 cycles, read DDRDSR[1]. If the value is 1, then use the next lower impedance, and read DDRDSR[1] again. Once a value of 0 is detected, then leave DDRCDR[DSO_NZ] at the calibrated value.

9. Clear DDRCDR[MDIC1_OE]

Note that the legal impedance values, from highest impedance to lowest impedance, are as follows:

- 0000
- 1000
- 1100
- 1110
- 1111

DDRCDR is shown in [Figure 5-25](#).

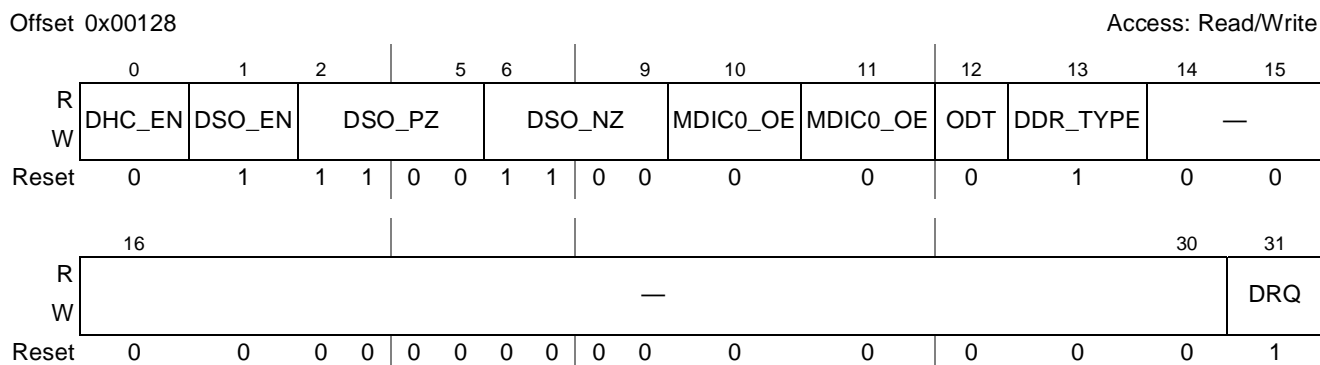


Figure 5-25. DDR Control Driver Register (DDRCDR)

[Table 5-43](#) shows the bit definition of the DDRCDR.

Table 5-43. DDRCDR Field Descriptions

Bits	Name	Description
0	DHC_EN	DDR driver hardware compensation enable
1	DSO_EN	0 DDR driver software override disable 1 DDR driver software override enable
2–5	DSO_PZ	DDR driver software p-impedance override 0000 Half strength—Highest Z 1000 Much higher Z than nominal 1100 Higher Z than nominal 1110 Nominal impedance setting 1111 Lower Z than nominal
6–9	DSO_NZ	DDR driver software n-impedance override 0000 Half strength—Highest Z 1000 Much higher Z than nominal 1100 Higher Z than nominal 1110 Nominal impedance setting 1111 Lower Z than nominal
10	MDIC0_OE	Output enable control for MDIC0 pin when software impedance calibration is performed. 0 - MDIC0 is disabled 1 - MDIC0 is enabled
11	MDIC1_OE	Output enable control for MDIC1 pin when software impedance calibration is performed. 0 - MDIC1 is disabled 1 - MDIC1 is enabled

5.5 Software Watchdog Timer (WDT)

The following sections describe the theory of operation of the software watchdog timer (WDT) in the device, including a definition of the external signals and the functions they serve. Additionally, the configuration, control, and status registers are also described. Note that individual chapters in this book describe specific initialization aspects for each individual block.

5.5.1 WDT Overview

The device provides a software watchdog timer (WDT) feature to prevent system lock in case the software becomes trapped in loops with no controlled exit. Watchdog timer operations are configured in the system watchdog control register (SWCRR).

The watchdog counter 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. 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).

Figure 5-27 shows a high-level block diagram of the WDT.

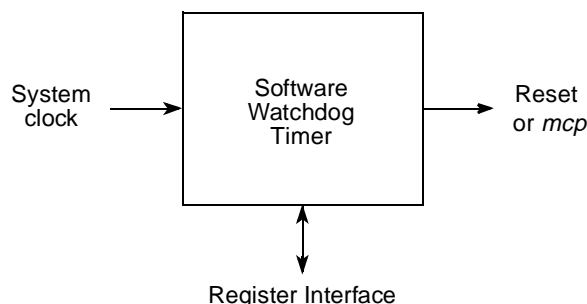


Figure 5-27. Software Watchdog Timer High-Level Block Diagram

The software watchdog timer is enabled after reset to cause a hardware reset if it times out. The user has the option of disabling the software watchdog if it is not needed. If used, the software watchdog timer requires a special service sequence to be executed periodically. Without this periodic servicing, the software watchdog timer times out and issues a reset or a non-maskable interrupt.

5.5.2 WDT Features

The WDT includes the following key features:

- Based on 16-bit prescaler and 16-bit down-counter
- Provides a selectable range for the time-out period
- Provides ~12.8-sec maximum software time-out delay for 333-MHz input clock
- Functional and programming compatibility with MPC8260 watchdog timer

5.5.3 WDT Modes of Operation

The WDT unit can operate in the following modes:

- WDT enable/disable mode:
If the software watchdog timer is not needed, the user can disable it with software after a system reset. When the watchdog timer is disabled, the watchdog counter and prescaler counter are held in a stopped state.
- WDT output reset/interrupt mode:
Without software periodic servicing, the software watchdog timer times out and issues a reset or a nonmaskable interrupt (*mcp*)
- WDT prescaled/non-prescaled clock mode:
The WDT counter clock can be prescaled by programming the SWCRR[SWPR] bit, which controls the divide-by-65,536 of the WDT counter.

5.5.4 WDT Memory Map/Register Definition

The WDT programmable register map occupies 16 bytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All WDT registers are 16- or 32-bits wide, located on 16-bit address boundaries, and should be accessed as 16- or 32-bit quantities. All addresses used in this chapter are offsets from the WDT base, as defined in Chapter 2, “Memory Map.”

Table 5-45 shows the WDT memory map.

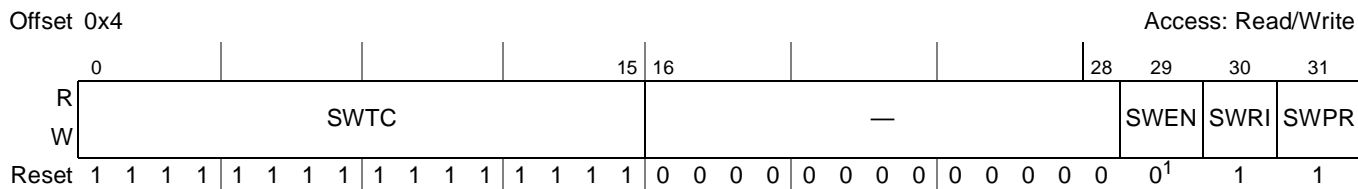
Table 5-45. WDT Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00–0x03	Reserved	—	—	—
0x04	System watchdog control register (SWCRR)	R/W	0xFFFF_0003 or 0xFFFF_0007 ¹	5.5.4.1/5-38
0x08	System watchdog count register (SWCNR)	R	0x0000_FFFF	5.5.4.2/5-39
0x0C–0x0D	Reserved	—	—	—
0x0E	System watchdog service register (SWSRR)	R/W	0x0000	5.5.4.3/5-39

¹ SWCRR[SWEN] reset value directly depends on RCWHR[SWEN] (reset configuration word high).

5.5.4.1 System Watchdog Control Register (SWCRR)

The system watchdog control register (SWCRR), shown in [Figure 5-28](#), controls the software watchdog period and configures watchdog timer operation. SWCRR can be read at any time but can be written only once after system reset.



¹ SWCRR[SWEN] reset value directly depends on RCWHR[SWEN] (reset configuration word high).

Figure 5-28. System Watchdog Control Register (SWCRR)

[Table 5-46](#) defines the bit fields of SWCRR.

Table 5-46. SWCRR Bit Settings

Bits	Name	Description
0–15	SWTC	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 0xFFFF. Note: The prescaler counter is reset any time a new value is loaded into the watchdog counter and also during reset.
16–28	—	Write reserved, read = 0
29	SWEN	Watchdog enable bit Enables the watchdog timer. The reset value directly depends on the value of the RCWHR[SWEN] bit. 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 Note: After software writes the SWRI bit, the state of SWEN cannot be changed.
30	SWRI	Software watchdog reset/interrupt select bit A WDT time out causes either 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
31	SWPR	Software watchdog counter prescale bit Controls the divide-by-65,536 WDT counter prescaler 0 The WDT counter is not prescaled. 1 The WDT counter clock is prescaled.

5.5.4.2 System Watchdog Count Register (SWCNR)

The system watchdog count register (SWCNR), shown in [Figure 5-29](#), provides visibility to the watchdog counter value. SWCNR is a read-only register. Writes to SWCNR have no effect and terminate without transfer error exception.

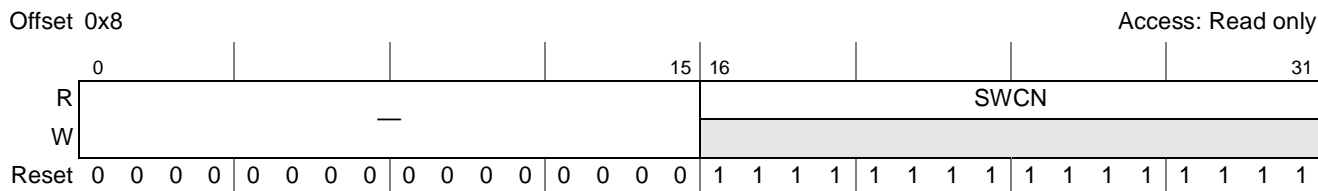


Figure 5-29. System Watchdog Count Register (SWCNR)

Table 5-47 defines the bit fields of SWCNR.

Table 5-47. SWCNR Bit Settings

Bits	Name	Description
0–15	—	Write reserved, read = 0
16–31	SWCN	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 0xFFFF. Note: Reading the 16 least-significant bits of 32-bit SWCNR register with two 8-bit reads is not guaranteed to return a coherent value.

5.5.4.3 System Watchdog Service Register (SWSRR)

The system watchdog service register (SWSRR) is shown in [Figure 5-30](#). When the watchdog timer is enabled, a write of 0x556C followed by a write 0xAA39 to the SWSRR register before the watchdog counter times out prevents a device reset. If the SWSRR register is not serviced before the timeout, a signal from the watchdog timer to the reset or interrupt controller module asserts a system reset or interrupt (depending on the setting of SWCRR[SWRI]).

Both writes must occur before the timeout in the order listed, but any number of instructions can be executed between the two writes. However, writing any value other than 0x556C or 0xAA39 to the SWSRR register resets the servicing sequence, requiring both values to be written to keep the watchdog timer from causing a reset. Reset initializes the SWSRR[WS] field to 0x0000. SWSRR can be written at any time, but returns all zeros when read.



Figure 5-30. System Watchdog Service Register (SWSRR)

Table 5-48 defines the bit fields of SWCNR.

Table 5-48. SWSRR Bit Settings

Bits	Name	Description
0–15	WS	Software watchdog service field. The user should periodically write 0x556C followed by 0xAA39 to this register to prevent a software watchdog timer timeout. SWSRR[WS] can be written at any time, but returns all zeros when read.

5.5.5 Functional Description

5.5.5.1 Software Watchdog Timer Unit

The device provides a software watchdog timer (WDT) feature to prevent system lock in case the software becomes trapped in loops with no controlled exit. Watchdog timer operations are configured in the system watchdog control register (SWCRR).

The software watchdog timer is enabled after reset to cause a soft reset or non-maskable interrupt (MCP) if it times out. If the software watchdog timer is not needed, the user must clear SWCRR[SWEN] to disable it. If used, the software watchdog timer requires a special service sequence to be executed periodically. Without this periodic servicing, the software watchdog timer times out and issues a reset or a nonmaskable interrupt, as programmed in SWCRR[SWRI]. Once software writes SWRI, the state of SWEN cannot be changed.

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 5-31 shows a state diagram for the watchdog timer.

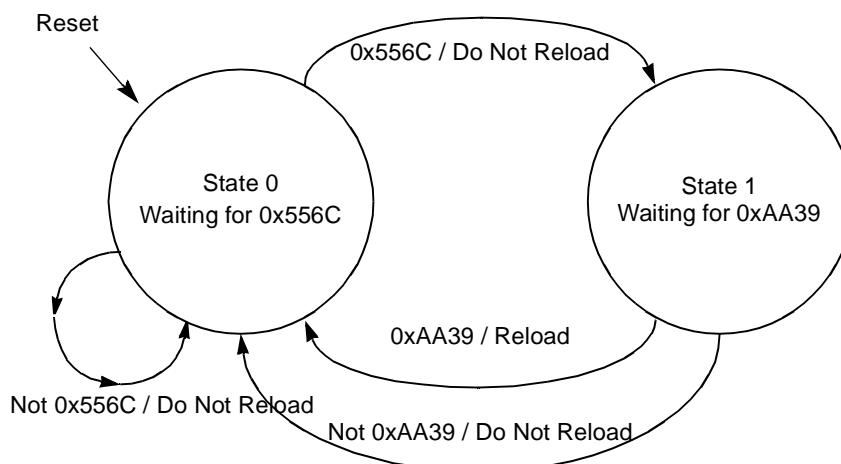


Figure 5-31. Software Watchdog Timer Service State Diagram

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 5-32 shows how to handle this need.

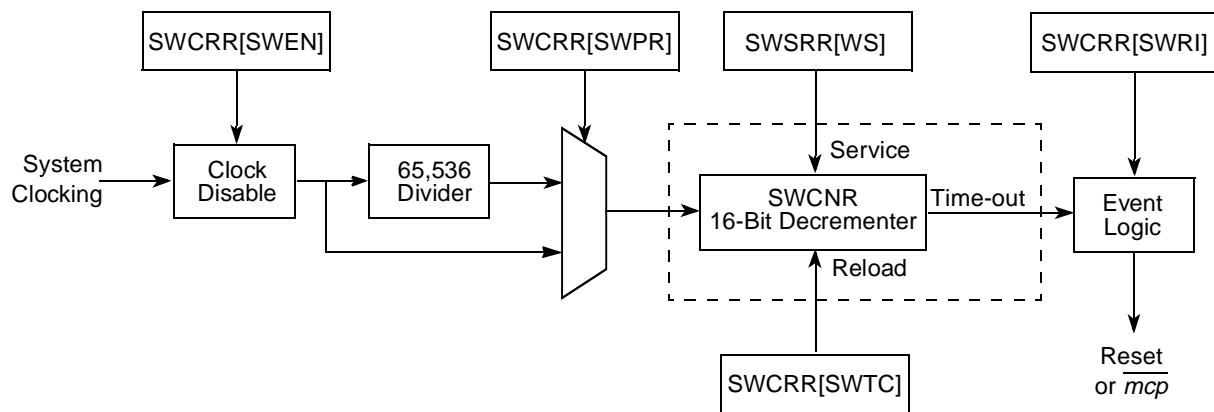


Figure 5-32. Software Watchdog Timer Functional Block Diagram

Figure 5-32 shows that the range is determined by SWCRR[SWTC]. The value in SWTC is then loaded into a 16-bit decremter clocked by the system clock. An additional divide-by-65,536 prescaler value is used when needed.

The decremter 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) 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.

5.5.5.2 Modes of Operation

The WDT unit can operate in the following modes:

- WDT enable/disable mode:

If the software watchdog timer is not needed, the user can disable it. The 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)
- WDT disable mode (SWCRR[SWEN] = 0)

- 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 the value of SWCRR[SWRI], the 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 hard 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 the SWCRR[SWPR] bit that controls the divide-by-65,536 of the WDT counter.

- Prescale mode (SWCRR[SWPR] = 1)

The WDT clock is prescaled.

- Non-prescale mode (SWCRR[SWPR] = 0)

The WDT clock is not prescaled.

5.5.6 Initialization/Application Information

5.5.6.1 WDT Programming Guidelines

The software watchdog timer is enabled (by the default value of SWCRR[SWEN]) after reset. The following initialization sequence of WDT is required:

- WDT disabling

If the software watchdog timer is not needed, the user must clear SWCRR[SWEN] bit to disable the WDT not later than its timer times out (~12.8 sec. for a 333-MHz system clock).

- WDT initial servicing

If the software watchdog timer is to be used, the special service sequence, described in [Section 5.5.5.1, “Software Watchdog Timer Unit,”](#) must be executed after system reset and not later than the first WDT time-out (~12.8 sec. for a 333-MHz system clock).

Subsequently, periodical WDT servicing should be performed according to the programming guidelines given in [Section 5.5.5.1, “Software Watchdog Timer Unit.”](#)

5.6 Real Time Clock Module (RTC)

The following sections describe the theory of operation of the real time clock module (RTC) including a definition of the external signals and the functions it serves. Additionally, the configuration, control, and status registers are described. Note that individual chapters in this reference manual describe additional specific initialization aspects for each individual block.

5.6.1 RTC Overview

The device platform provides a real time clock (RTC) timer suitable for timestamping or time and calendar generation. It can maintain a one-second count which is unique over a period of approximately 136 years.

The RTC can be initialized by software with an initial count value using the real time counter load register (RTLDR). It can also be programmed to generate an interrupt every second. The real time counter control register (RTCTR) is used to enable or disable the various timer functions. The real time counter event

register (RTEVR) is used to report the interrupt source. The RTC counter is initialized by software and can be disabled if needed.

Figure 5-33 shows the high level RTC block diagram.

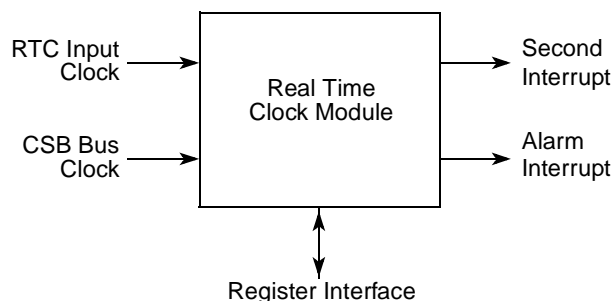


Figure 5-33. Real Time Clock Module High Level Block Diagram

5.6.2 RTC Features

The key features of the RTC include the following:

- Maintains a one-second count, unique over a period of thousand of years
- 32-bit RTC counter can be initialized by software to specific initial count value
- Provides an alarm function with programmable and maskable alarm interrupt
- Provides programmable and maskable every second interrupt
- Uses two possible clock sources: the CSB bus clock or an external RTC clock
- RTC function can be disabled if needed

5.6.3 RTC Modes of Operation

The RTC unit can operate in the following modes:

- RTC enable/disable mode
- RTC every-second interrupt enable/disable mode
- RTC alarm interrupt enable/disable mode
- RTC internal/external input clock mode

5.6.4 RTC External Signal Description

This section provides an overview and detailed descriptions of the RTC signals.

There is one distinct external RTC clock input signal, defined in [Table 5-49](#).

Table 5-49. RTC Signal Properties

Name	Port	Function	I/O	Reset	Pull Up
RTC_CLK	RTC_CLK	Real time clock input. In the MPC8360E, RTC_CLK is one of the possible functions of PC[26]. See Table 3-18 in Chapter 3 , “ Signal Descriptions .”	I	N/A	—

Table 5-50 provides a detailed description of the external RTC signal.

Table 5-50. RTC External Signal—Detailed Signal Description

Signal	I/O	Description
RTC_CLK	I	This signal is used as the timebase for the real time clock module.
		State Meaning —
		Timing —

5.6.5 RTC Memory Map/Register Definition

The RTC programmable register map occupies 32 bytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All RTC registers are 32 bits wide that are located on 32-bit address boundaries and should only be accessed as a 32-bit quantities.

All addresses used in this section are offsets from the RTC base, as defined in Chapter 2, “Memory Map.”

Table 5-45 shows the memory map of the RTC.

Table 5-51. RTC Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00	Real time counter control register (RTCNR)	R/W	All zeros	5.6.5.1/5-44
0x04	Real time counter load register (RTLDR)	R/W	All zeros	5.6.5.2/5-45
0x08	Real time counter prescale register (RTPSR)	R/W	All zeros	5.6.5.3/5-46
0x0C	Real time counter register (RTCTR)	R	All zeros	5.6.5.4/5-46
0x10	Real time counter event register (RTEVR)	w1c	All zeros	5.6.5.5/5-47
0x14	Real time counter alarm register (RTALR)	R/W	0xFFFF_FFFF	5.6.5.6/5-47
0x18–0x1F	Reserved	—	—	—

5.6.5.1 Real Time Counter Control Register (RTCNR)

The real time counter control register (RTCNR), shown in Figure 5-34, is used to enable RTC functions. The register can be read at any time.

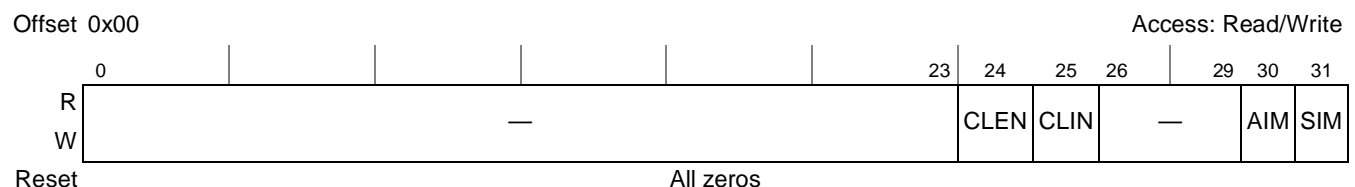


Figure 5-34. Real Time Counter Control Register (RTCNR)

Table 5-52 defines the bit fields of RTCNR.

Table 5-52. RTCNR Bit Settings

Bits	Name	Description
0–23	—	Write reserved, read = 0
24	CLEN	Clock enable control bit. This bit controls the counting of the RTC. When the RTC's clock is disabled, the counter maintains its old value. When the counter's clock is enabled, it continues counting using the previous value. 0 Disable counter. 1 Enable counter.
25	CLIN	Input clock control bit. The input clock to the RTC may be either the CSB clock or an external RTC clock. 0 The input clock to the periodic interrupt timer is CSB input clock. 1 The input clock to the periodic interrupt timer is the external RTC clock.
26–29	—	Write reserved, read = 0
30	AIM	Alarm interrupt mask bit. Used to enable or disable (mask) the RTC alarm interrupt when the RTC's 32-bit counter reaches RTALR[ALRM] value. 0 Alarm interrupt generation disabled. 1 Alarm interrupt generation enabled.
31	SIM	Second interrupt mask bit. Used to enable or disable (mask) the RTC periodic interrupt. 0 Periodic interrupt generation disabled. 1 Periodic interrupt generation enabled.

5.6.5.2 Real Time Counter Load Register (RTLDR)

The real time counter load register (RTLDR), shown in Figure 5-35, contains the 32-bit value to be loaded in the 32-bit RTC counter.

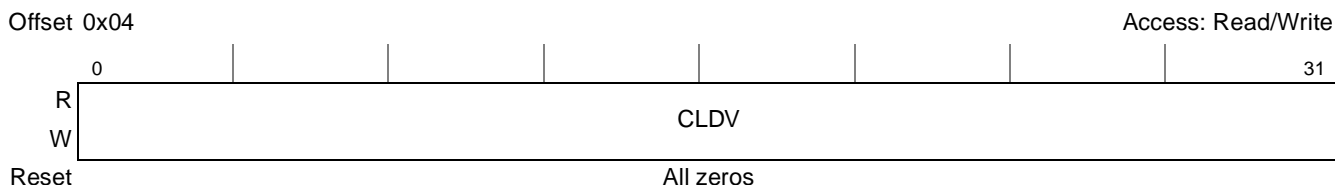


Figure 5-35. Real Time Counter Load Register (RTLDR)

Table 5-53 defines the bit fields of RTLDR.

Table 5-53. RTLDR Bit Settings

Bits	Name	Description
0–31	CLDV	Contains the 32-bit value to be loaded in the 32-bit RTC counter.

5.6.5.3 Real Time Counter Prescale Register (RTPSR)

The real time counter prescale register (RTPSR), shown in [Figure 5-36](#), is a read/write register used to configure the RTC prescaler's value.

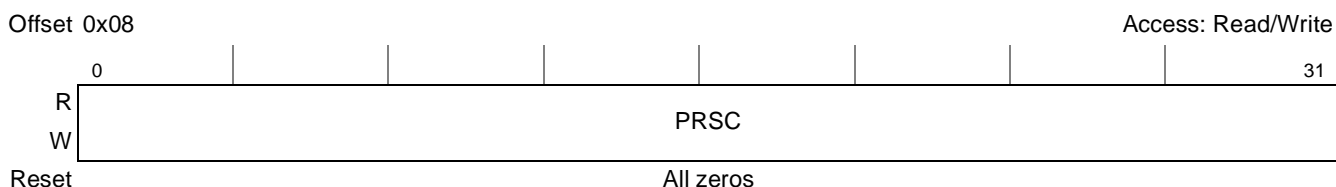


Figure 5-36. Real Time Counter Prescale Register (RTPSR)

[Table 5-54](#) defines the bit fields of RTPSR.

Table 5-54. RTPSR Bit Settings

Bits	Name	Description
0–31	PRSC	RTC prescaler bits. Select the input clock divider for the RTC counter clock. The prescaler is programmed to divide the RTC clock input by values from 1 to 4,294,967,296. The value 0x0000 divides the clock by 1 and 0xFFFF_FFFF divides the clock by 4,294,967,296. To accurately predict the timing of the next count, change the RTPSR[PRSC] field only when the enable bit RTCNR[CLE] is clear. Changing the RTPSR[PRSC] bits resets the prescaler counter. System reset and the loading of a new value into the counter both reset the prescaler counter. Clearing RTCNR[CLE] stops the prescaler counter.

5.6.5.4 Real Time Counter Register (RTCTR)

The real time counter register (RTCTR), shown in [Figure 5-37](#), is a read-only register that shows the current value in the RTC counter.

The CNTV value is not affected by reads or writes to RTCTR.

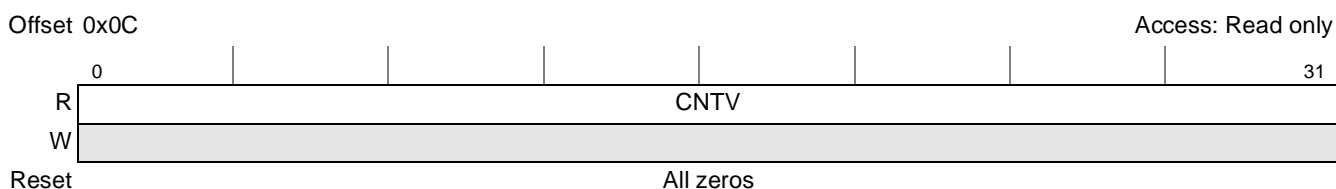


Figure 5-37. Real Time Counter Register (RTCTR)

[Table 5-55](#) defines the bit fields of RTCTR.

Table 5-55. RTCTR Bit Settings

Bits	Name	Description
0–31	CNTV	RTC counter value field. RTCTR[CNTV] contains the current value of the time counter. This is a read-only field. Writes have no effect on RTCTR[CNTV].

5.6.5.5 Real Time Counter Event Register (RTEVR)

The real time counter event register (RTEVR), shown in [Figure 5-38](#), is used to report the source of the interrupts. The register can be read at any time.

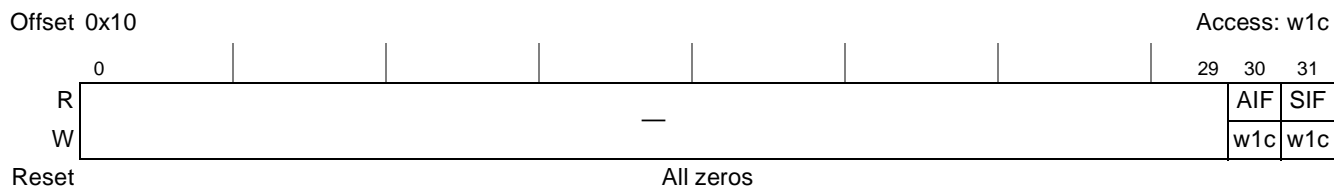


Figure 5-38. Real Time Counter Event Register (RTEVR)

RTEVR bits are cleared by writing ones. Writing zeros does not affect the value of the status bits.

[Table 5-56](#) defines the bit fields of RTEVR.

Table 5-56. RTEVR Bit Settings

Bits	Name	Description
0–29	—	Write reserved, read = 0
30	AIF	Alarm interrupt flag bit. Used to indicate the alarm interrupt. The bit is set if the RTC issues an interrupt when the RTC counter value equals RTALR[ALRM].
31	SIF	Second interrupt flag bit. Used to indicate the every-second interrupt. This status bit is set each time that the prescaler count reaches zero and should be cleared by software.

5.6.5.6 Real Time Counter Alarm Register (RTALR)

The real time counter alarm register (RTALR), shown in [Figure 5-39](#), contains the 32-bit alarm (ALRM) value. When the value of the RTC counter equals the RTALR[ALRM] value, a maskable interrupt is generated.

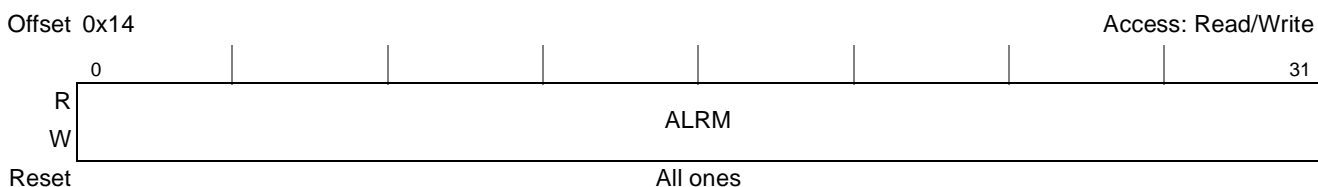


Figure 5-39. Real Time Counter Alarm Register (RTALR)

[Table 5-57](#) defines the bit fields of RTALR.

Table 5-57. RTALR Bit Settings

Bits	Name	Description
0–31	ALRM	RTC alarm value. The alarm interrupt is generated when the value of the RTC counter equals RTALR[ALRM].

5.6.6 Functional Description

5.6.6.1 Real Time Counter Unit

The real time clock (RTC) timer is suitable for time stamping or time and calendar generation. It can maintain a one-second count which is unique over a period of approximately 136 years. Software can convert this count into time-of-day or calendar information if required. An alarm function is also provided. The RTC can be clocked by the internal system bus clock or by an external clock source. The RTC consists of 32-bit up-counter which is incremented by an one-second count clock derived from the RTC input clock. The RTC can be programmed to generate a maskable interrupt when the time value matches the value in its associated alarm register.

The RTC can be initialized by software with an initial count value in the real time counter load register (RTLDR). It can also be programmed to generate an interrupt every second. The real time counter control register (RTCTR) is used to enable or disable the various timer functions. The real time counter event register (RTEVR) is used to report the interrupt source. The RTC counter is reset to zero on hard reset but is not affected by soft reset. It is initialized by the software. The RTC function can be disabled.

Figure 5-40 shows the functional RTC block diagram.

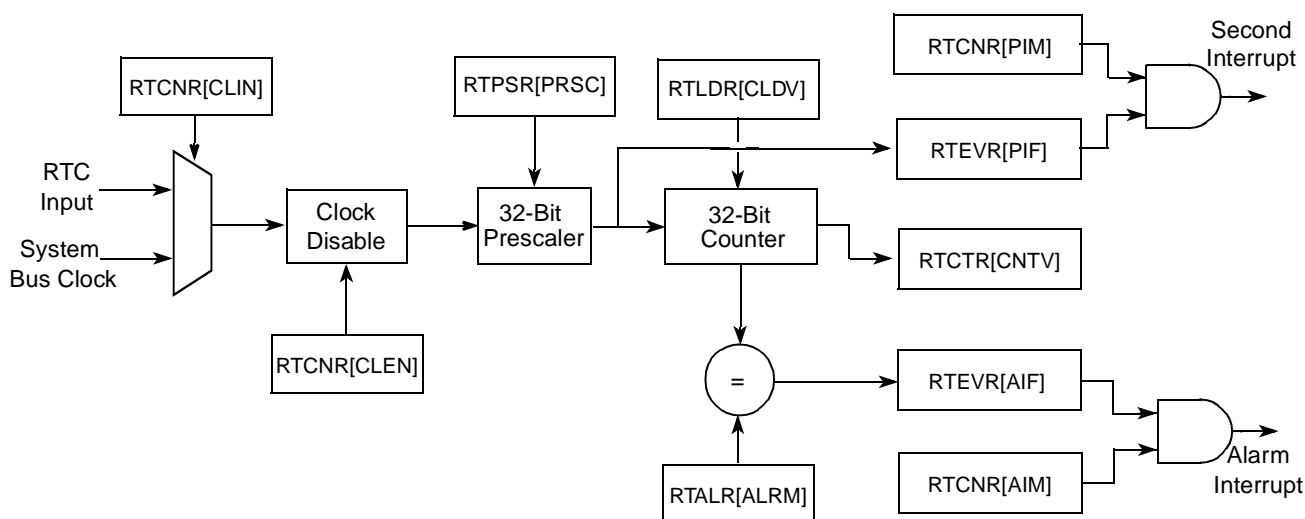


Figure 5-40. Real Time Clock Module Functional Block Diagram

5.6.6.2 RTC Operational Modes

The RTC unit can operate in the following modes:

- RTC enable/disable mode:

RTCNR[CLEN] enables the RTC timer. It should be set by software after a system reset to enable the RTC timer.

— RTC disable mode (RTCNR[CLEN] = 0)

When the RTC's clock is disabled, counter maintains its old value (default).

— RTC enable mode (RTCNR[CLEN] = 1)

When the counter's clock is enabled, it continues counting using the previous value.

- RTC every-second interrupt enable/disable mode:
 - RTC every-second interrupt enable mode (RTCNR[SIM] = 1)
In this mode the RTC set the RTEVR[SIF] flag and generate an interrupt after the RTC's 32-bit counter reaches zero.
 - RTC every-second interrupt disable mode (RTCNR[SIM] = 0)
In this mode the RTC sets the RTEVR[SIF] flag but does not generate an interrupt after the RTC's 32-bit counter reaches zero.
- RTC alarm interrupt enable/disable mode:
 - RTC alarm interrupt enable mode (RTCNR[AIM] = 1)
In this mode, the RTC sets the RTEVR[AIF] flag and generates an interrupt each time when the RTC's 32-bit counter reaches the RTALR[ALRM] value.
 - RTC alarm interrupt disable mode (RTCNR[AIM] = 0)
In this mode the RTC sets the RTEVR[AIF] flag but does not generate an interrupt when the RTC's 32-bit counter reaches the RTALR[ALRM] value.
- RTC internal/external input clock mode:
The input clock to the RTC may be the CSB clock or an external 32.768-kHz crystal.
 - RTC uses the internal input clock mode (RTCNR[CLIN] = 0)
 - RTC uses the external 32.768-kHz crystal clock (RTCNR[CLIN] = 1)

5.6.7 RTC Programming Guidelines

The following initialization sequence for the RTC is recommended:

1. Write to RTPSR to set the RTC prescaler to the desired value
2. Write to RTLDR to initialize the RTC initial value
3. Write to RTALR to program the RTC alarm value, if needed
4. Write to RTCNR to configure and start the RTC operation: RTC input clock source, second/alarm interrupt mask, RTC clock enable.

5.7 Periodic Interval Timer (PIT)

The following sections describe theory of operation of the periodic interval timer (PIT) including a definition of the external signals and the functions it serves. Additionally, the configuration, control, and status registers are described. Note that individual chapters in this reference manual describe additional specific initialization aspects for each individual block.

5.7.1 PIT Overview

The periodic interval timer (PIT) that generates periodic interrupts for a real-time operating system or an application software.

The PIT consists of a 32-bit down-counter which is decremented by a clock derived from a CSB clock or from an external 32.768-kHz crystal. The 32-bit counter decrements to zero when loaded with a initial value from the periodic interval timer load register (PTLDR). The periodic interval timer control register (PTCTR) is used to enable or disable the various timer functions. The periodic interval timer event register (PTEVR) is used to report the interrupt source. The PIT function can be disabled if needed.

Figure 5-41 shows the functional PIT block diagram.

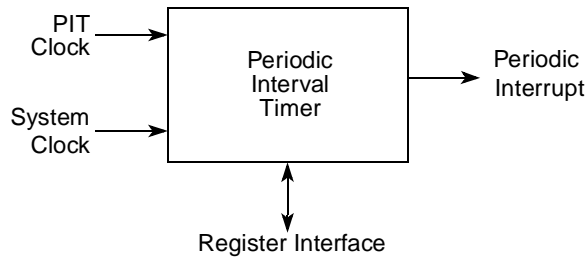


Figure 5-41. Periodic Interval Timer High Level Block Diagram

5.7.2 PIT Features

The key features of the PIT include the following:

- Maintains a 32-bit down-counter, clocked by a 32-bit prescaled input clock
- 32-bit PIT counter can be initialized by software to specific initial count value
- Provides programmable and maskable periodic interrupt
- Uses two possible clock sources: the CSB clock or an external PIT clock
- PIT function can be disabled

5.7.3 PIT Modes of Operation

The PIT unit can operate in the following modes:

- PIT enable/disable mode
- PIT periodic interrupt enable/disable mode
- PIT internal/external input clock mode

5.7.4 PIT External Signal Description

This section provides an overview and detailed descriptions of the PIT signals.

There is one distinct external input signal (PIT clock), defined in [Table 5-58](#).

Table 5-58. PIT Signal Properties

Name	Port	Function	I/O	Reset	Pull Up
PIT_CLK	PIT_CLK	Periodic interval timer. In MPC8360E, PIT_CLK is one of the possible functions of PC[26]. See Table 3-18 in Chapter 3 , “Signal Descriptions .	I	N/A	—

Table 5-61 defines the bit fields of PTCNR.

Table 5-61. PTCNR Bit Settings

Bits	Name	Description
0–23	—	Write reserved, read = 0
24	CLEN	Clock enable control bit. Controls the counting of the PIT. When the PIT's clock is disabled, the counter maintains its old value. When the counter's clock is enabled, it continues counting using the previous value. 0 Disable counter. 1 Enable counter.
25	CLIN	Input clock control bit. The input clock to the PIT can be either an internal system clock or an external PIT clock. 0 The input clock to the periodic interrupt timer is internal system clock. 1 The input clock to the periodic interrupt timer is external PIT clock.
26–30	—	Write reserved, read = 0
31	PIM	Periodic interrupt mask bit. Used to enable or disable (mask) the PIT periodic interrupt. 0 Periodic interrupt generation disabled. 1 Periodic interrupt generation enabled.

5.7.5.2 Periodic Interval Timer Load Register (PTLDR)

The periodic interval timer load register (PTLDR), shown in Figure 5-43, contains the 32-bit value to be loaded in a 32-bit PIT counter.

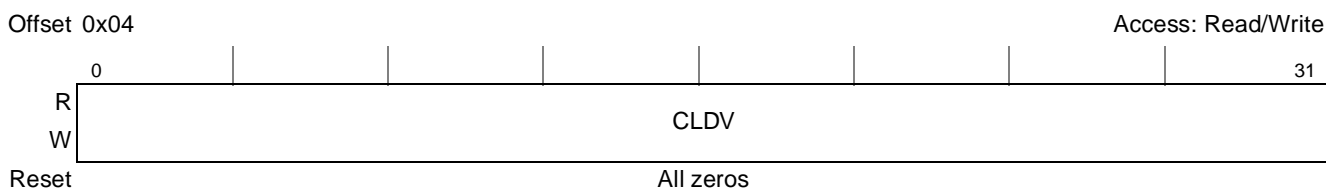


Figure 5-43. Periodic Interval Timer Load Register (PTLDR)

Table 5-62 defines the bit fields of PTLDR.

Table 5-62. PTLDR Bit Settings

Bits	Name	Description
0–31	CLDV	Contains the 32-bit value to be loaded in a 32-bit PIT counter.

5.7.5.3 Periodic Interval Timer Prescale Register (PTPSR)

The periodic interval timer prescale register (PTPSR), shown in [Figure 5-44](#), is a read/write register that used to configure the PIT prescaler's value.

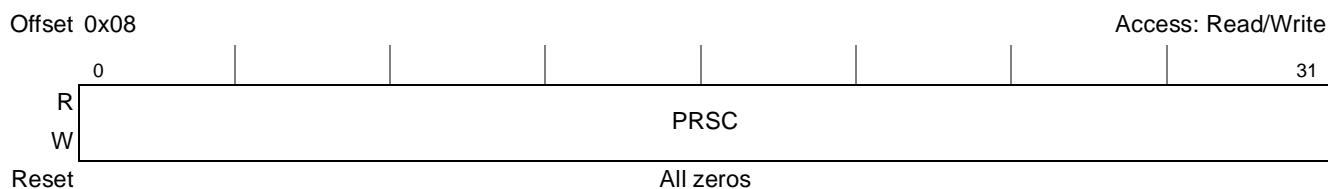


Figure 5-44. Periodic Interval Timer Prescale Register (PTPSR)

[Table 5-63](#) defines the bit fields of PTPSR.

Table 5-63. PTPSR Bit Settings

Bits	Name	Description
0–31	PRSC	PIT prescaler bits. Selects the input clock divider to generate the PIT counter clock. The prescaler is programmed to divide the PIT clock input by values from 1 to 4,294,967,296. The value 0x0000 divides the clock by 1 and 0xFFFF_FFFF divides the clock by 4,294,967,296. To accurately predict the timing of the next count, change the PRSC bit only when the enable bit PTCNR[CLE] is clear. Changing PRSC resets the prescaler counter. System reset and the loading of a new value into the counter also reset the prescaler counter. Clearing the PTCNR[CLE] bit stops the prescaler counter.

5.7.5.4 Periodic Interval Timer Counter Register (PTCTR)

The periodic interval timer counter register (PTCTR), shown in [Figure 5-45](#), is a read-only register that shows the current value in the PIT counter. The PTCTR counter is not affected by reads or writes.

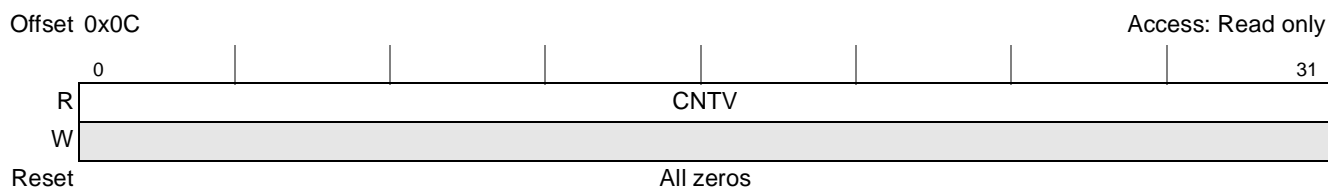


Figure 5-45. Periodic Interval Timer Counter Register (PTCTR)

[Table 5-64](#) defines the bit fields of PTCTR.

Table 5-64. PTCTR Bit Settings

Bits	Name	Description
0–31	CNTV	PIT counter value field. Contains the current value of the time counter. This is a read-only field. Writes have no effect on PTCTR[CNTV].

5.7.5.5 Periodic Interval Timer Event Register (PTEVR)

The periodic interval timer event register (PTEVR), shown in Figure 5-46, is used to report the source of the interrupts. The register can be read at any time.

PTEVR bits are cleared by writing ones. Writing zeros does not affect the value of the status bits.

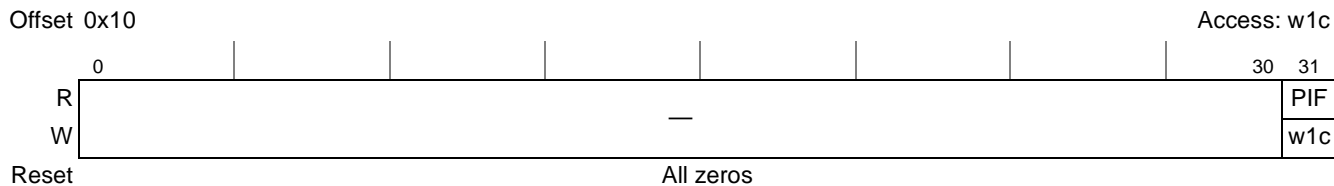


Figure 5-46. Periodic Interval Timer Event Register (PTEVR)

Table 5-65 defines the bit fields of PTEVR.

Table 5-65. PTEVR Bit Settings

Bits	Name	Description
0–30	—	Write reserved, read = 0
31	PIF	Periodic interrupt flag bit. It is asserted after the SPMPIT counter counts to zero. This status bit should be cleared by software.

5.7.6 Functional Description

5.7.6.1 Periodic Interval Timer Unit

The PIT generates periodic interrupts for use with a real-time operating system or the application software. It consists of a 32-bit down-counter which is decremented by a clock derived from the CSB clock or from the PIT clock. The 32-bit counter decrements to zero when loaded with a initial value from the periodic interval timer load register (PTLDR). After the timer reaches zero, PTEVR[PIF] is set and an interrupt is generated if PTCNR[PIM] = 1. At the next count cycle, the value in the PTLDR[CLDV] is loaded into the counter and the process repeats. When a new value is loaded into the PTLDR[CLDV], the PIT is updated, the prescaler counter is reset, and the counter begins counting. Setting of PTEVR[PIF] generates an interrupt, that remains pending until PTEVR[PIF] is cleared. If PTEVR[PIF] is set again before being cleared, the interrupt remains pending until PTEVR[PIF] is cleared. Any write to the PTLDR[CLDV] stops the current countdown and the count resumes with the new value in PTLDR[CLDV]. If PTCNR[CLEN] = 0, the PIT cannot count and retains the old count value. PTCTR contain the PIT current value. The PIT function can be disabled if needed.

Figure 5-47 shows the functional PIT block diagram.

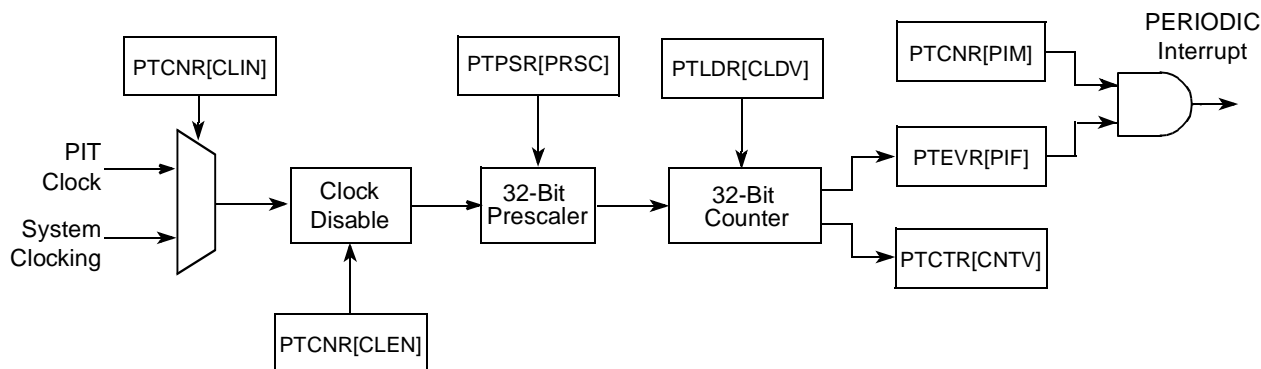


Figure 5-47. Periodic Interval Timer Functional Block Diagram

5.7.6.2 PIT Operational Modes

The PIT unit can operate in the following modes:

- PIT enable/disable mode:

The PTCNR[CLEN] bit enables the PIT timer. It should be set by software after a system reset to enable the PIT timer.

 - PIT disable mode (PTCNR[CLEN] = 0). When the PIT's clock is disabled, counter maintains its old value.
 - PIT enable mode (PTCNR[CLEN] = 1). When the counter's clock is enabled, it continues counting using the previous value.
- PIT periodic interrupt enable/disable mode:
 - PIT periodic interrupt enable mode (PTCNR[PIM] = 1). After the PIT's 32-bit counter reaches zero, the PIT sets the PTEVR[PIF] flag and generates an interrupt.
 - PIT periodic interrupt disable mode (PTCNR[PIM] = 0). After the PIT's 32-bit counter reaches zero, the PIT sets the PTEVR[PIF] flag but does not generate an interrupt.
- PIT internal/external input clock mode:

The input clock to the PIT may be an internal system clock or the PIT clock.

 - PIT use the internal input clock mode (PTCNR[CLIN] = 0)
 - PIT use the PIT clock (PTCNR[CLIN] = 1)

5.7.7 PIT Programming Guidelines

The following initialization sequence of PIT is recommended:

1. Write to PTPSR to set the PIT prescaler to the desired value
2. Write to PTLDR to initialize the PIT initial value
3. Write to PTCNR to configure and start the PIT operation: PIT input clock source, periodic interrupt mask, PIT clock enable.

For real-time clock programming guidelines, see [Section 5.6.7, "RTC Programming Guidelines."](#)

5.8 General-Purpose Timers (GTM)

The following sections describe theory of operation of the two general purpose (global) timer modules, including a definition of the external signals and the functionality. Additionally, the configuration, control, and status registers are described. Note that individual chapters in this book describe additional specific initialization aspects for each individual block.

5.8.1 GTM Overview

Each global timer module (GTM) includes four identical 16-bit general-purpose timers, two 32-bit timers or one 64-bit timer. Each GTM timer consists of a timer prescale register (GTPSR), a timer mode register (GTMDR) a timer capture register (GTCPR), a timer counter register (GTCNR), a timer reference register (GTRFR), a timer event register (GTEVR), and a timer global configuration register (GTCFR). The GTPSRs and the GTMDRs contain the primary and secondary prescalers, programmed by the user.

Note that while the MPC8360E has two global timer modules, GTM2_TOUT2 and GTM2_TOUT4 signals are not available externally.

Figure 5-48 shows the functional GTM block diagram.

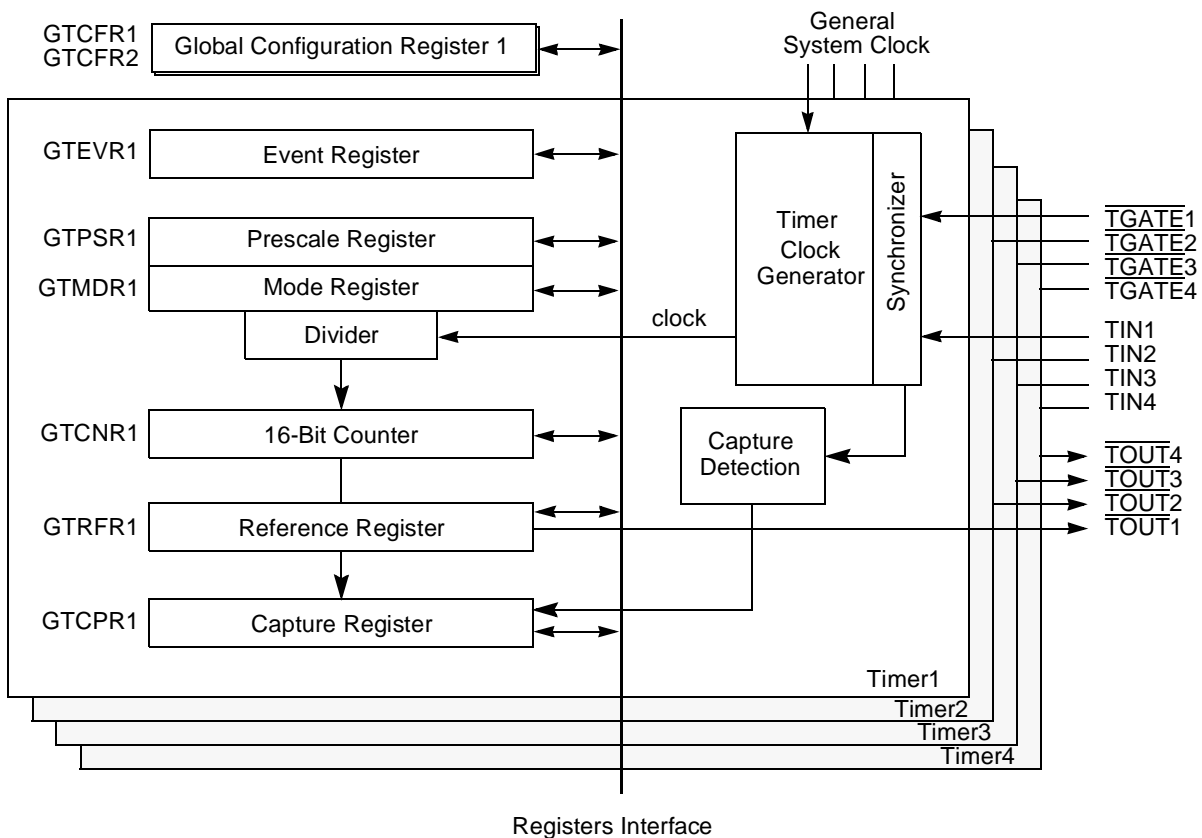


Figure 5-48. Global Timers Block Diagram

5.8.2 GTM Features

The key features of the timer include the following:

- The maximum input clock is the system bus clock
- Four 16-bit programmable timers
- Two timers cascaded internally or externally to form a 32-bit timer
- One timer cascaded internally or externally to form a 64-bit timer
- Maximum period of ~206 seconds (at 333-MHz bus clock in slow go mode, primary and secondary prescaler = 256) for 16-bit timer
- Maximum period of ~3298 seconds (at 333-MHz bus clock and prescaler = 256) for 32-bit timer
- Maximum period of thousands of years (at 333-MHz bus clock and prescaler = 256) for 64-bit timer
- 3-nanosecond timer resolution (at 333-MHz bus clock and no prescaler)
- Resolution and maximum period can be traded off by selecting prescaler divisor
- Three programmable input clock sources for the timer prescalers
- Input capture capability
- Output compare with programmable mode for the output pin
- Free run and restart modes
- Functional and programming compatibility with MPC8260 timers

5.8.3 GTM Modes of Operation

The GTM unit can operate in the following modes:

- Cascaded modes
- Clock source modes
- Reference modes
- Capture modes

5.8.3.1 Cascaded Modes

$GTCFR_n[PCAS]$ and $GTCFR_2[SCAS]$ are used to put the timers into different cascaded modes:

- Non-cascaded mode: Each timer (timer 1, timer 2, timer 3, and timer 4), function as a independent 16-bit timer with a 16-bit GTRFR, GTCPR, GTMDR, and GTCNR. In this mode, the non-cascaded GTRFR, GTCPR, and GTCNR should be referenced with corresponding 16-bit bus cycles.
- Pair-cascaded mode: In this mode, two 16-bit timers can be internally cascaded to form a 32-bit counter: timer 1 can be internally cascaded to timer 2 and timer 3 may be internally cascaded to timer 4. Because the decision to cascade timers is made independently, the user has the option of selecting two 16-bit timers and one 32-bit timer, or two 32-bit timers. When working in the pair-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with 32-bit bus cycles.

- Super-cascaded mode: In this mode, all four 16-bit timers can be internally cascaded to form a 64-bit counter. When working in the super-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with two 32-bit bus cycles.

5.8.3.2 Clock Source Modes

The clock input to the timer's prescaler can be selected from three sources:

- The system clock
- The system slow go clock (system bus clock internally divided by 16)
- The corresponding TINx pin

5.8.3.3 Reference Modes

Each timer can be configured to count until a reference is reached and then either begin a new time count immediately or continue to run. The FRR bit of the corresponding GTMRR selects each mode.

- Free run reference mode. The corresponding timer count continues to increment after the reference value is reached.
- Reset reference mode. The corresponding timer count is reset immediately after the reference value is reached.

5.8.3.4 Capture Modes

Each timer has a 16-bit field in GTCPR, used to latch the value of the counter when a defined transition of TINx is sensed by the corresponding input capture edge detector.

- Normal gate mode enables the count on a falling edge of the $\overline{\text{TGATE}}$ pin and disables the count on the rising edge of $\overline{\text{TGATE}}$. This mode allows the timer to count conditionally, based on the state of $\overline{\text{TGATE}}$.
- The restart gate mode performs the same function as normal mode, except it also resets the counter on the falling edge of the $\overline{\text{TGATE}}$ pin. This mode has applications in pulse interval measurement and bus monitoring.

5.8.4 GTM External Signal Description

This section provides an overview and detailed descriptions of the GTM signals.

There are four distinct external input timer capture signals (TIN1, TIN2, TIN3, and TIN4), four distinct external input timer gate signals ($\overline{\text{TGATE1}}$, $\overline{\text{TGATE2}}$, $\overline{\text{TGATE3}}$, and $\overline{\text{TGATE4}}$), and four distinct external timer output signals ($\overline{\text{TOUT1}}$, $\overline{\text{TOUT2}}$, $\overline{\text{TOUT3}}$, and $\overline{\text{TOUT4}}$). The GTM interface signals are defined in [Table 5-66](#).

Table 5-66. GTM Signal Properties

Name	Port	Function	I/O	Reset	Require Pull Up
TIN1	TIN1	Global timer 1 capture control signal	I	0	No
TIN2	TIN2	Global timer 2 capture control signal	I	0	No
TIN3	TIN3	Global timer 3 capture control signal	I	0	No
TIN4	TIN4	Global timer 4 capture control signal	I	0	No
$\overline{\text{TGATE1}}$	$\overline{\text{TGATE1}}$	Global timer 1 counter gate control signal	I	0	No
$\overline{\text{TGATE2}}$	$\overline{\text{TGATE2}}$	Global timer 2 counter gate control signal	I	0	No
$\overline{\text{TGATE3}}$	$\overline{\text{TGATE3}}$	Global timer 3 counter gate control signal	I	0	No
$\overline{\text{TGATE4}}$	$\overline{\text{TGATE4}}$	Global timer 4 counter gate control signal	I	0	No
$\overline{\text{TOUT1}}$	$\overline{\text{TOUT1}}$	Global timer 1 counter output signal	O	1	No
$\overline{\text{TOUT2}}$	$\overline{\text{TOUT2}}$	Global timer 2 counter output signal	O	1	No
$\overline{\text{TOUT3}}$	$\overline{\text{TOUT3}}$	Global timer 3 counter output signal	O	1	No
$\overline{\text{TOUT4}}$	$\overline{\text{TOUT4}}$	Global timer 4 counter output signal	O	1	No

Table 5-67 provides detailed descriptions of the external GTM signals.

Table 5-67. GTM External Signals—Detailed Signal Descriptions

Signal	I/O	Description
TIN n	I	Global timer capture control signal. Used to latch the value of the counter when a defined transition of TIN n is sensed by the corresponding input capture edge detector.
		State Meaning Asserted/Negated—According to the programmed polarity by the corresponding GTMDR n [CE]. Each timer has a 16-bit GTCPR used to latch the value of the counter when a defined transition of TIN n is sensed by the corresponding input capture edge detector. Upon a capture or reference event, the corresponding GTEVR bit is set and a maskable interrupt request is issued to the interrupt controller.
		Timing Assertion/Negation—Asynchronous to internal bus clock. TIN n is internally synchronized to the system bus clock. If TIN n meets the asynchronous input setup time, the value of counter is captured after one system bus clock when working with the internal clock.
$\overline{\text{TGATE}}_n$	I	Global timer counter gate control signal. Used to gate/restart the counter when a defined transition of $\overline{\text{TGATE}}_n$ is sensed by the corresponding input capture edge detector.
		State Meaning Asserted/Negated—According to the programmed polarity by the corresponding GTCFR[GM x] bits. In a restart gate mode (GTCFR[GM n] = 0), the $\overline{\text{TGATE}}_n$ pin is used to enable/disable count. A falling $\overline{\text{TGATE}}_n$ pin enables and restarts the count and a rising edge of $\overline{\text{TGATE}}_n$ disables the count. In a normal gate mode (GTCFR[GM n] = 1), the $\overline{\text{TGATE}}_n$ have similar functionality, except the falling edge of $\overline{\text{TGATE}}_n$ does not restart the appropriate count value in GTCNR n [CNV n].
		Timing Assertion/Negation—Asynchronous to internal bus clock. $\overline{\text{TGATE}}_n$ is internally synchronized to the system bus clock. If $\overline{\text{TGATE}}_n$ meets the asynchronous input setup time, the counter begins counting or stops counting (depending on the signal state and the configured mode) after one system bus clock when working with the internal clock.

Table 5-67. GTM External Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
$\overline{\text{TOUT}}_n$	O	Global timer counter output signal. The GTM output a signal on the timer output pin $\overline{\text{TOUT}}_n$ when the reference value is reached.
		State Meaning Asserted/Negated—According to the programmed polarity by the corresponding $\text{GTMDR}_n[\text{OM}_n]$. 1. Active-low pulse on $\overline{\text{TOUT}}_n$ for one timer input clock cycle as defined by the $\text{GTMDR}_n[\text{ICLK}_n]$ bits ($\text{GTMDR}_n[\text{OM}_n] = 1$). Thus, $\overline{\text{TOUT}}_n$ may be low for one general system clock period, one general system slow go clock period, or one TIN_n pin clock cycle period. 2. Toggle the $\overline{\text{TOUT}}_n$ pin ($\text{GTMDR}_n[\text{OM}_n] = 0$). $\overline{\text{TOUT}}_n$ begins or stops counting, depending on the signal state and the configured mode.
		Timing Assertion/Negation— $\overline{\text{TOUT}}_n$ changes occur on the rising edge of the timer input clock.

5.8.5 GTM Memory Map/Register Definition

The GTM programmable register map occupies 64 bytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All GTM registers are 8 or 16 bits wide, located on 8-bit or 16-bit address boundaries, and should only be accessed as 8-bit or 16-bit quantities. All addresses used in this chapter are offsets from GPT Base, as defined in [Chapter 2, “Memory Map.”](#)

[Table 5-68](#) shows the memory map of the GTM.

Table 5-68. GTM Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
General Purpose (Global) Timer Module 1—Block Base Address 0x0_0500				
0x00	Timer 1 and 2 global timers configuration register (GTCFR1)	R/W	0x00	5.8.5.1/5-61
0x01–0x03	Reserved	—	—	—
0x04	Timer 3 and 4 global timers configuration register (GTCFR2)	R/W	0x00	5.8.5.1/5-61
0x05–0x0F	Reserved	—	—	—
0x10	Timer 1 global timers mode register (GTMDR1)	R/W	0x0000	5.8.5.2/5-65
0x12	Timer 2 global timers mode register (GTMDR2)			
0x14	Timer 1 global timers reference register (GTRFR1)	R/W	0xFFFF	5.8.5.3/5-66
0x16	Timer 2 global timers reference register (GTRFR2)			
0x18	Timer 1 global timers capture register (GTCPR1)	R/W	0x0000	5.8.5.4/5-66
0x1A	Timer 2 global timers capture register (GTCPR2)			
0x1C	Timer 1 global timers counter register (GTCNR1)	R/W	0x0000	5.8.5.5/5-67
0x1E	Timer 2 global timers counter register (GTCNR2)			

Table 5-68. GTM Register Address Map (continued)

Offset	Register	Access	Reset Value	Section/ Page
0x20	Timer 3 global timers mode register (GTMDR3)	R/W	0x0000	5.8.5.2/5-65
0x22	Timer 4 global timers mode register (GTMDR4)			
0x24	Timer 3 global timers reference register (GTRFR3)	R/W	0xFFFF	5.8.5.3/5-66
0x26	Timer 4 global timers reference register (GTRFR4)			
0x28	Timer 3 global timers capture register (GTCPR3)	R	0x0000	5.8.5.4/5-66
0x2A	Timer 4 global timers capture register (GTCPR4)			
0x2C	Timer 3 global timers counter register (GTCNR3)	R/W	0x0000	5.8.5.5/5-67
0x2E	Timer 4 global timers counter register (GTCNR4)			
0x30	Timer 1 global timers event register (GTEVR1)	w1c	0x0000	5.8.5.6/5-67
0x32	Timer 2 global timers event register (GTEVR2)			
0x34	Timer 3 global timers event register (GTEVR3)			
0x36	Timer 4 global timers event register (GTEVR4)			
0x38	Timer 1 global timers prescale register (GTPSR1)	R/W	0x0003	5.8.5.7/5-68
0x3A	Timer 2 global timers prescale register (GTPSR2)			
0x3C	Timer 3 global timers prescale register (GTPSR3)			
0x3E	Timer 4 global timers prescale register (GTPSR4)			
General Purpose (Global) Timer Module 2: All registers defined for GTM1 are also defined for GTM2; the base address of GTM2 registers is 0x0_06nn.				

5.8.5.1 Global Timers Configuration Registers (GTCFR n)

The global timers configuration registers (GTCFR1 and GTCFR2), shown in [Figure 5-49](#) and [Figure 5-50](#), contain configuration parameters used by the timers. These registers allow simultaneous starting, stopping

and resetting of a pair of timers (1 and 2 or 3 and 4) or of a groups of timers (1, 2, 3, and 4) if one bus cycle is used. GTCFR is cleared by reset.

NOTE

For proper operation of the timers, do not change the modes of operation and enable the timer in the same register write operation. The modes can be changed when $GTCFR_n[RST_n]$ is cleared. However, when $GTCFR_n[RST_n]$ are set, they are the only bits that can be changed.

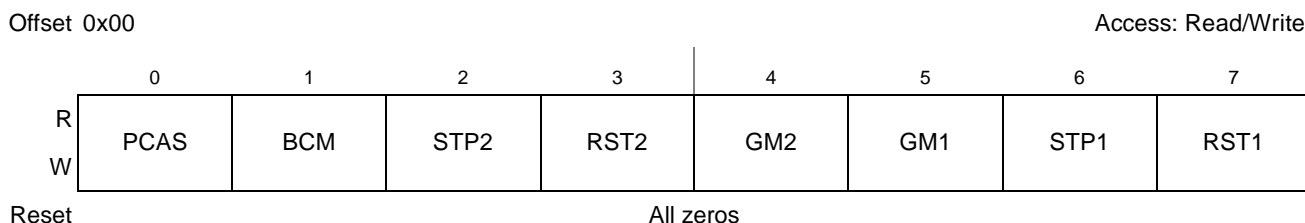


Figure 5-49. Global Timers Configuration Register 1 (GTCFR1)

Table 5-69 defines the bit fields of GTCFR1.

Table 5-69. GTCFR1 Bit Settings

Bits	Name	Description
0	PCAS	Pair-cascade mode 0 Normal operation 1 Timers 1 and 2 cascade to form a 32-bit timer. Note: This bit is ignored in super-cascade mode ($GTCFR2[SCAS] = 1$). Note: It is allowed to change the value of this bit only when the corresponding timers are in reset mode. Thus, the user should first clear the RST1 and RST2 bits (without changing PCAS) and then, in a separate write to the register, change the value of PCAS.
1	BCM	Backward compatible mode 0 Provide backward compatibility to PowerQUICC II family timers. In this mode $GTCFR1[GM2]$ bit will control the gate mode for timers 1 and 2 and $GTCFR2[GM4]$ bit will control the gate mode for timers 3 and 4. $GTCFR1[GM1]$ and $GTCFR2[GM3]$ bits are ignored. 1 Normal operational mode
2	STP2	Stop timer 2 0 Normal operation 1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 2, except the Register Interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
3	RST2	Reset timer 2 0 Reset the timer 2, including GTMDR2, GTRFR2, GTCNR2, GTCPR2, and GTEVR2 (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP2 bit is cleared.
4	GM2	Gate mode for $\overline{TGATE2}$ 0 Restart gate mode. The $\overline{TGATE2}$ pin is used to enable/disable count. A low level of $\overline{TGATE2}$ enables and a falling edge of $\overline{TGATE2}$ restarts the count (reset the dynamic counter's count value to 0) and a high level of $\overline{TGATE2}$ disables the count. 1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{TGATE2}$ does not restart the appropriate count value in $GTCNR2[CNV2]$.

Table 5-69. GTCFR1 Bit Settings (continued)

Bits	Name	Description
5	GM1	<p>Gate mode for $\overline{\text{TGATE1}}$</p> <p>0 Restart gate mode. The $\overline{\text{TGATE1}}$ is used to enable/disable count. A low level of $\overline{\text{TGATE1}}$ enables and a falling edge of $\overline{\text{TGATE1}}$ restarts the count (reset the dynamic counter's count value to 0) and a high level of $\overline{\text{TGATE1}}$ disables the count.</p> <p>1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE1}}$ does not restart the appropriate count value in GTCNR1[CNV1].</p> <p>Note: In backward compatible mode (GTCFR1[BCM] = 0) this bit is ignored. GTCFR1[GM2] bit will control the gate mode for timers 1 and 2.</p>
6	STP1	<p>Stop timer 1</p> <p>0 Normal operation</p> <p>1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 1, except the register interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.</p>
7	RST1	<p>Reset timer 1</p> <p>0 Reset the timer 1, including GTMDR1, GTRFR1, GTCNR1, GTCPR1, and GTEVR1 (a software reset is identical to an external reset).</p> <p>1 Enable the corresponding timer if the STP1 bit is cleared.</p>

The GTCFR2 register is shown in [Figure 5-50](#).

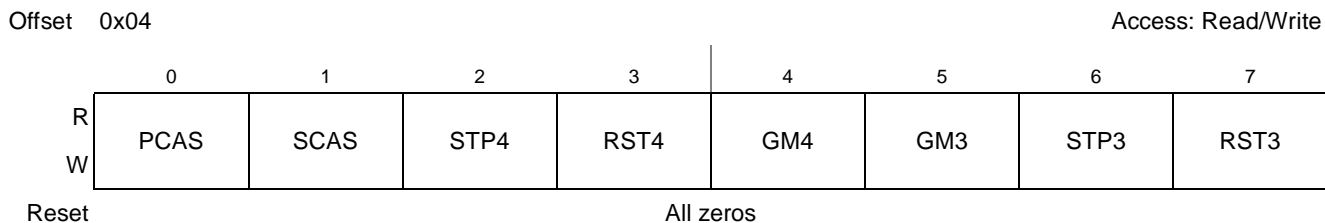


Figure 5-50. Global Timers Configuration Register 2 (GTCFR2)

Table 5-70 defines the bit fields of GTCFR2.

Table 5-70. GTCFR2 Bit Settings

Bits	Name	Description
0	PCAS	<p>Pair-cascade mode</p> <p>0 Normal operation.</p> <p>1 Timers 3 and 4 cascade to form a 32-bit timer.</p> <p>Note: This bit is ignored in super-cascade mode (GTCFR2[SCAS] = 1).</p> <p>Note: It is allowed to change the value of this bit only when the corresponding timers are in reset mode. Thus, the user should first clear the RST3 and RST4 bits (without changing PCAS) and then, in a separate write to the register, change the value of PCAS.</p>
1	SCAS	<p>Super cascade mode</p> <p>0 Normal operation</p> <p>1 Timers 1, 2, 3 and 4 cascade to form a 64-bit timer.</p> <p>Note: In super-cascade mode (GTCFR2[SCAS] = 1) the pair-cascade mode bits are ignored, (GTCFR1/2[PCAS] = Don't Care).</p> <p>Note: It is allowed to change the value of this bit only when the corresponding timers are in reset mode. Thus, the user should first clear the RST1, RST2, RST3, and RST4 bits (without changing SCAS) and then, in a separate write to the register, change the value of SCAS.</p>
2	STP4	<p>Stop timer 4</p> <p>0 Normal operation</p> <p>1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 4, except the register interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.</p>
3	RST4	<p>Reset timer 4</p> <p>0 Reset the timer 4, including GTMDR4, GTRFR4, GTCNR4, GTCPR4, and GTEVR4 (a software reset is identical to an external reset).</p> <p>1 Enable the corresponding timer if the STP4 bit is cleared.</p>
4	GM4	<p>Gate mode for $\overline{\text{TGATE4}}$</p> <p>0 Restart gate mode. The $\overline{\text{TGATE4}}$ is used to enable/disable count. A low level of $\overline{\text{TGATE4}}$ enables and a falling edge of $\overline{\text{TGATE4}}$ restarts the count (reset the dynamic counter's count value to 0) and a high level of $\overline{\text{TGATE4}}$ disables the count.</p> <p>1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE4}}$ does not restart the appropriate count value in GTCNR4[CNV4].</p>
5	GM3	<p>Gate mode for $\overline{\text{TGATE3}}$</p> <p>0 Restart gate mode. The $\overline{\text{TGATE3}}$ is used to enable/disable count. A low level of $\overline{\text{TGATE3}}$ enables and a falling edge of $\overline{\text{TGATE3}}$ restarts the count (reset the dynamic counter's count value to 0) and a high level of $\overline{\text{TGATE3}}$ disables the count.</p> <p>1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE3}}$ does not restart the appropriate count value in GTCNR3[CNV3].</p> <p>Note: In backward compatible mode (GTCFR1[BCM] = 0) this bit is ignored. The GTCFR2[GM4] bit controls the gate mode for timers 3 and 4.</p>
6	STP3	<p>Stop timer 3</p> <p>0 Normal operation</p> <p>1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 3, except the register interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.</p>
7	RST3	<p>Reset timer 3</p> <p>0 Reset the timer 3, including GTMDR3, GTRFR3, GTCNR3, GTCPR3, and GTEVR3 (a software reset is identical to an external reset).</p> <p>1 Enable the corresponding timer if the STP3 bit is cleared.</p>

5.8.5.2 Global Timers Mode Registers (GTMDR1–GTMDR4)

The global timers mode registers (GTMDR1, GTMDR2, GTMDR3, and GTMDR4) are shown in Figure 5-51.

Erratic behavior may occur if GTCFR1 and GTCFR2 are not initialized before the GTMDR n . Only GTCFR n [RST n] and GTCFR n [STP n] can be modified at any time.

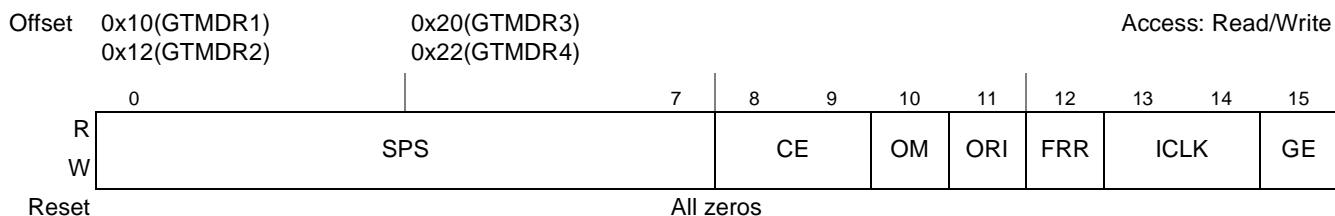


Figure 5-51. Global Timers Mode Registers (GTMDR1–GTMDR4)

Table 5-71 defines the bit fields of GTMDR.

Table 5-71. GTMDR Bit Settings

Bits	Name	Description
0–7	SPS	Secondary prescaler value The secondary prescaler is programmed to divide the clock input to corresponding timer by values from 1 to 256. The value 0x00 divides the clock by 1 and 0xFF divides the clock by 256.
8–9	CE	Capture edge and enable interrupt 00 Disable interrupt on capture event; capture function is disabled 01 Capture on rising TIN n edge only and enable interrupt on capture event. 10 Capture on falling TIN n edge only and enable interrupt on capture event. 11 Capture on any TIN n edge and enable interrupt on capture event. Note: The frequency of TIN n should be slower than system clock (TIN n is sampled internally by system clock to detect TIN n 's rising/falling edge before updating the counter)
10	OM	Output mode 0 Toggle $\overline{\text{TOUT}}_n$ every time when the corresponding timer matches its reference value. 1 Active-low pulse on $\overline{\text{TOUT}}_n$ for one timer input clock cycle (4 input clock cycles for the system clock) as defined by the ICLK n bits. Thus, $\overline{\text{TOUT}}_n$ may be low for four general system clocks, one general system slow go clock period, or one TIN n pin clock cycle period. Note: $\overline{\text{TOUT}}_n$ changes are internally synchronized to the rising edge of the system clock
11	ORI	Output reference interrupt enable 0 Disable interrupt for reference reached (does not affect interrupt on capture function). 1 Enable interrupt on reaching the reference value.
12	FRR	Free run/restart mode 0 Free run. The timer count continues to increment after the reference value is reached. 1 Restart. The timer count is reset immediately after the reference value is reached.

Table 5-71. GTMDR Bit Settings (continued)

Bits	Name	Description
13–14	ICLK	Input clock source for the timer. 00 Internally cascaded input. This selection means: For ICLK1, the timer 1 input is the output of timer 2. For ICLK2, the timer 1 input is the output of timer 2, the timer 2 input is the output of timer 3, the timer 3 input is the output of timer 4. For ICLK3, the timer 3 input is the output of timer 4. For ICLK4 this selection means no input clock is provided to the timer. 01 Internal general system bus clock. 10 Internal slow go clock (divided by 16 system bus clock). 11 TIN n : corresponding TIN1, TIN2, TIN3, or TIN4 pin (falling edge).
15	GE	Gate enable 0 The $\overline{\text{TGATE}}_n$ signal is ignored. 1 The $\overline{\text{TGATE}}_n$ signal is used to control the timer.

5.8.5.3 Global Timers Reference Registers (GTRFR1–GTRFR4)

Global timers reference registers, shown in [Figure 5-52](#), are 16-bit memory-mapped, read/write registers containing the 16-bit reference values for each timer’s timeout. The reference value is not reached until $\text{GTCNR}_n[\text{CNV}]$ increments to the value in $\text{GTRFR}_n[\text{TRV}]$.

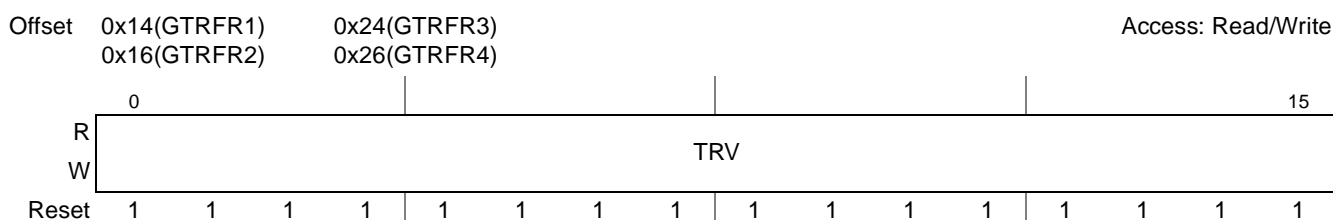


Figure 5-52. Global Timers Reference Registers (GTRFR1–GTRFR4)

[Table 5-72](#) defines the bit fields of GTRFR.

Table 5-72. GTRFR Bit Settings

Bits	Name	Description
0–15	TRV	Timeout reference value. 16-bit timeout reference value for the corresponding timer. Set to all ones by reset.

5.8.5.4 Global Timers Capture Registers (GTCPR1–GTCPR4)

Global timers capture registers (GTCPR_1 , GTCPR_2 , GTCPR_3 , and GTCPR_4), shown in [Figure 5-53](#), are used to latch the value of the counters according to $\text{GTMDR}_n[\text{CE}]$.

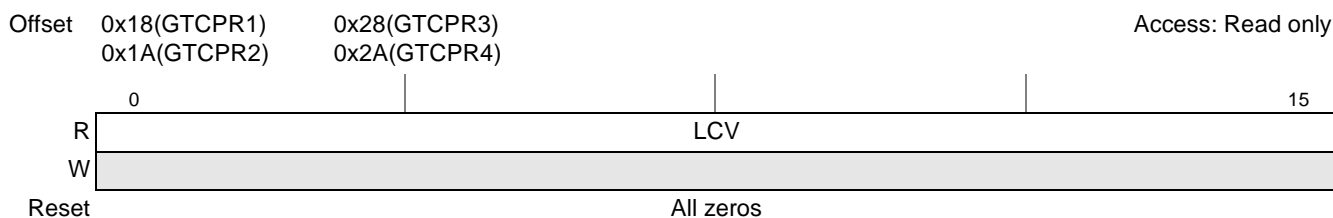


Figure 5-53. Global Timers Capture Registers (GTCPR1–GTCPR4)

Table 5-73 defines the bit fields of $GTCPR_n$.

Table 5-73. GTCPR_n Bit Settings

Bits	Name	Description
0–15	LCV	Latched counter value. Corresponding timer's 16-bit latched value.

5.8.5.5 Global Timers Counter Registers (GTCNR1–GTCNR4)

Global timers counter registers (GTCNR1, GTCNR2, GTCNR3, and GTCNR4), shown in Figure 5-54, are four 16-bit, memory-mapped, read/write up-counters. A read cycle to a $GTCNR_n[CNV]$ fields yields the current value of the appropriate timer but does not affect the counting operation. A write cycle to a $GTCNR_n[CNV]$ field sets the register to the written value, causing its corresponding primary and secondary prescaler counters to be reset.

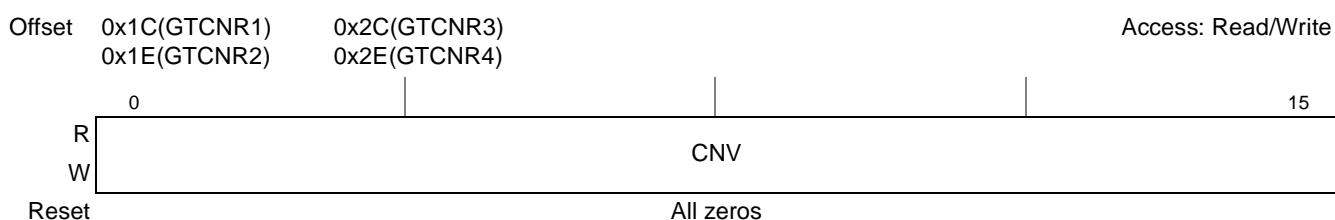


Figure 5-54. Global Timers Counter Registers (GTCNR1—GTCNR4)

Table 5-74 defines the bit fields of GTCNR.

Table 5-74. GTCNR Bit Settings

Bits	Name	Description
0–15	CNV	Counter value. Corresponding timer's 16-bit read/write up-counter value.

5.8.5.6 Global Timers Event Registers (GTEVR1–GTEVR4)

Global timers event registers (GTEVR1, GTEVR2, GTEVR3, and GTEVR4), shown in Figure 5-55, are used to report events recognized by any of the timers. On recognition of an output reference event, the appropriate timer sets $GTEVR_n[REF]$, regardless of the corresponding $GTMDR_n[ORI]$. The capture event is only set if it is enabled by $GTMDR_n[CE]$. GTEVRs appear as memory-mapped registers to users, which can be read at any time.

$GTEVR_n$ bits are cleared by writing ones to them (writing zeros does not affect bit values). Both bits must be reset before the timer negates the interrupt to the interrupt controller.

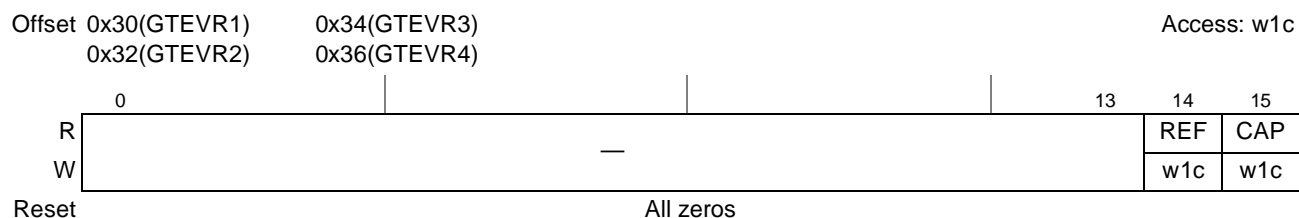


Figure 5-55. Global Timers Event Registers (GTEVR1—GTEVR4)

Table 5-75 defines the bit fields of GTEVR_n.

Table 5-75. GTEVR_n Bit Settings

Bits	Name	Description
0–13	—	Reserved, should be cleared.
14	REF	Output reference event 0 No event 1 The counter reached the GTRFR _n [TRV] value. GTMDR _n [ORI] is used to enable the interrupt request caused by this event.
15	CAP	Counter capture event Corresponding timer's 16-bit read/write up-counter value. 0 No event 1 The counter value has been latched into the GTCPR _n [LCV]. GTMDR _n [CE] is used to enable generation of this event.

5.8.5.7 Global Timers Prescale Registers (GTPSR1–GTPSR4)

The global timers prescale registers (GTPSR1, GTPSR2, GTPSR3, and GTPSR4) are shown in Figure 5-56.

Erratic behavior may occur if GTPSR_n is not initialized before the corresponding GTMDR_n.

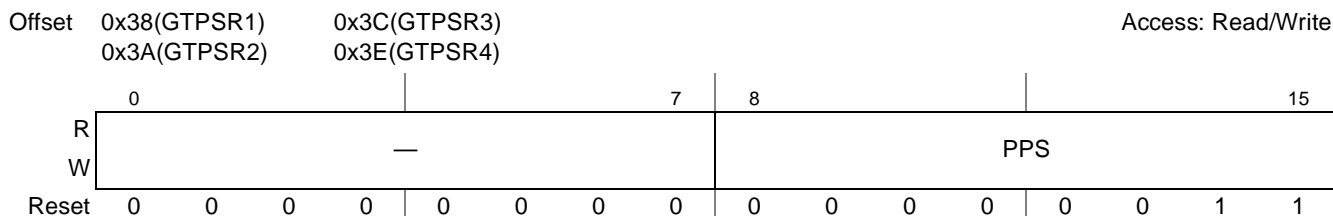


Figure 5-56. Global Timers Prescale Registers (GTPSR1–GTPSR4)

Table 5-76 defines the bit fields of GTPSR_n.

Table 5-76. GTPSR_n Bit Settings

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–15	PPS	Primary prescaler bits The primary prescaler is programmed to divide the clock input to corresponding timer by values from 1 to 256. The value 0x00 divides the clock by 1 and 0xFF divides the clock by 256.

NOTE

The total timer prescale value is calculated as follows:

$$GTMn_{prescaler} = (GTPSRn[PPS] + 1) \cdot (GTMDRn[SPS] + 1)$$

This gives a total prescale range from 1 (GTPSR_n[PPS] = 0x00, GTMDR_n[SPS] = 0x00) to 65,536 (GTPSR_n[PPS] = 0xFF, GTMDR[SPS] = 0xFF).

5.8.6 Functional Description

5.8.6.1 General-Purpose Timer Units

The clock input to the timer's prescaler can be selected from the following sources:

- The system clock
- The system slow go clock (internally divided by 16)

The general system clock is generated in the clock synthesizer and defaults to the system frequency. However, the general system clock has the option to be divided before it leaves the clock synthesizer. This mode, called slow go, is used to save power. Whatever the resulting frequency of the general system clock, the user can either choose that frequency or the frequency divided by 16 as the input to the prescaler of each timer. Alternatively, the user may prefer TIN_n to be the clock source. TIN_n is internally synchronized to the internal clock. If the user has chosen to internally cascade two 16-bit timers to a 32-bit timer, then a timer can use the clock generated by the output of another timer.

The clock input source is selected by the corresponding $GTMDR_n[ICLK]$ bits. The prescalers ($GTMDR_n[SPS]$ and $GTPSR_n[PPS]$) can be programmed to divide the clock input by values from 1 to 65,536 and the output of the prescaler is used as an input to the 16-bit counters. The best resolution of the timer is one clock cycle (3 ns at a 333-MHz system clock, for example). The maximum period (when the reference value is all ones and the prescaler divides by 256) for one 16-bit timer is ~206 s at 333 MHz.

5.8.6.2 Reference Modes

Each timer can be configured to count until a reference is reached and then either begin a new time count immediately or continue to run. The FRR bit of the corresponding $GTMRR$ selects each mode.

- Free run reference mode ($GTMDR_n[FRR] = 0$)
The corresponding timer count continues to increment after the reference value is reached.
- Reset reference mode ($GTMDR_n[FRR] = 1$)
The corresponding timer count is reset immediately after the reference value is reached.

Upon reaching the reference value, the corresponding $GTEVR_n[REF]$ bit is set and an interrupt is issued if $GTMDR_n[ORI] = 1$. The timers can output a signal on the timer output pin \overline{TOUT}_n if the reference value is reached (selected by the corresponding $GTMDR_n[OM]$). This signal can be an active-low pulse or a toggle of the current output. The output can also be connected internally to the input of another timer, resulting in a 32- or 64-bit timer.

5.8.6.3 Capture Modes

In addition, each timer has a 16-bit field in $GTCPR$, used to latch the value of the counter when a defined transition of TIN_n is sensed by the corresponding input capture edge detector. The timers may be gated/restarted by an external gate signals (\overline{TGATE}_n) that controls the timers. The type of transition triggering the capture is selected by the corresponding $GTMDR_n[CE]$ bits. Upon a capture or reference

event, corresponding $GTEVR_n[REF]$ or $GTEVR_n[CAP]$ is set and a maskable interrupt request is issued to the interrupt controller.

- Normal gate mode enables the count on a falling edge of \overline{TGATE} and disables the count on the rising edge of \overline{TGATE} . This mode allows the timer to count conditionally, based on the state of \overline{TGATE} .
- The restart gate mode performs the same function as normal mode, except it also resets the counter on the falling edge of \overline{TGATE} .

This mode has applications in pulse interval measurement and bus monitoring as follows:

- Pulse measurement—The restart gate mode can measure a low pulse on \overline{TGATE} . The rising edge of \overline{TGATE} completes the measurement and if \overline{TGATE}_n is connected externally to TIN_n , it causes the timer to capture the count value and generate a rising-edge interrupt.
- Bus monitoring—The restart gate mode can detect a signal that is stuck abnormally low. The bus signal should be connected to \overline{TGATE} . The timer count is reset on the falling edge of the bus signal and if the bus signal does not go high again within the number of user-defined clocks, an interrupt can be generated.

The gate function is enabled in the $GTMDR$; the gate operating mode is selected in the $GTCFR_n$.

NOTE

\overline{TGATE} is internally synchronized to the system clock. If \overline{TGATE} meets the asynchronous input setup time, the counter begins or stops counting after one system clock when working with the internal clock.

5.8.6.4 Cascaded Modes

$GTCFR_n[PCAS]$ and $GTCFR2[SCAS]$ are used to put the timers into different cascaded modes:

- Non-cascaded mode ($GTCFR_n[PCAS] = 0$ and $GTCFR2[SCAS] = 0$)
 If $GTCFR_n[PCAS] = 0$ and $GTCFR2[SCAS] = 0$, the each timer (timer 1, timer 2, timer 3, and timer 4), function as a independent 16-bit timer with a 16-bit $GTRFR$, $GTCPR$, $GTMDR$, and $GTCNR$ for each one (Figure 5-57). When working in the none-cascaded mode, the non-cascaded $GTRFR$, $GTCPR$, and $GTCNR$ should be referenced with appropriate 16-bit bus cycles.

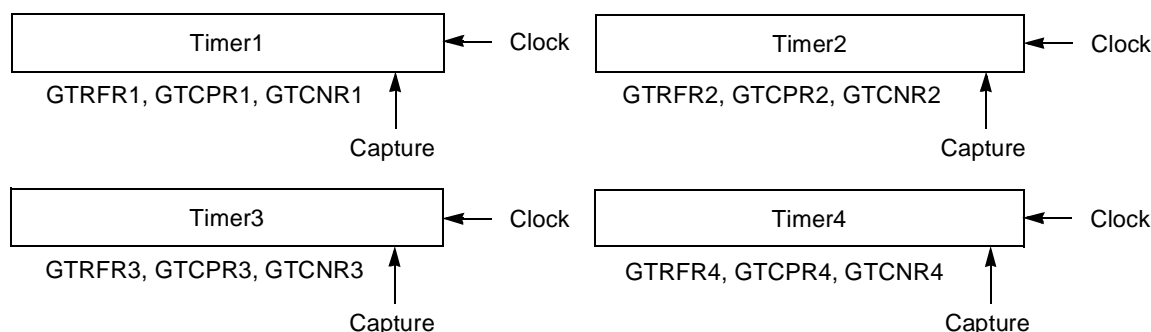


Figure 5-57. Timers Non-Cascaded Mode Block Diagram

- Pair-cascaded mode ($GTCFR1[PCAS] = 1$ and/or $GTCFR2[PCAS] = 1$, $GTCFR2[SCAS] = 0$)
 In this mode, two 16-bit timers can be internally cascaded to form a 32-bit counter: timer 1 may be internally cascaded to timer 2 and timer 3 may be internally cascaded to timer 4, as shown in [Figure 5-58](#). Since the decision to cascade timers is made independently, the user has the option of selecting two 16-bit timers and one 32-bit timer ($GTCFR1[PCAS] = 1$, $GTCFR2[PCAS] = 0$ or $GTCFR1[PCAS] = 0$, $GTCFR2[PCAS] = 1$), or two 32-bit timers ($GTCFR1[PCAS] = 1$ and $GTCFR2[PCAS] = 1$).

If $GTCFR1[PCAS] = 1$ and/or $GTCFR2[SCAS] = 1$, the two 16-bit timers (timer 1 and timer 2 or timer 3 and timer 4) function as a 32-bit timer with a 32-bit GTRFR, GTCPR, and GTCNR. In this case, GTMDR1/GTMDR3 is ignored, and the modes and functions are defined using GTMDR2/GTMDR4, and GTCFR1/GTCFR2. The capture are controlled from TIN2, and the interrupts are generated from GTEVR2. When working in the pair-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with 32-bit bus cycles.

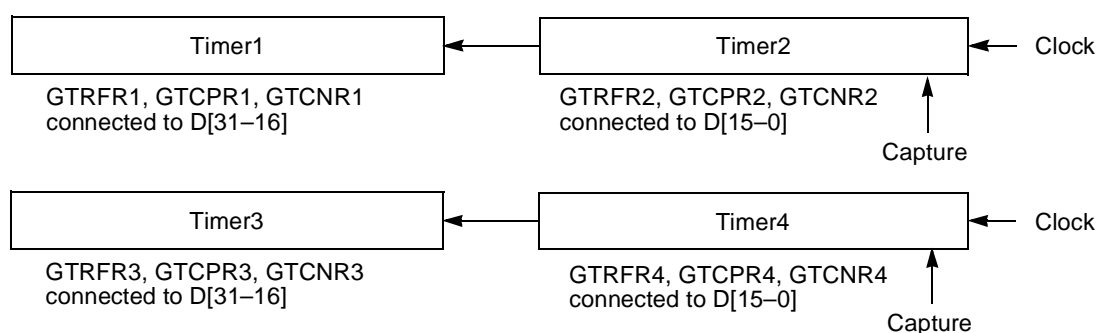


Figure 5-58. Timer Pair-Cascaded Mode Block Diagram

- Super-cascaded mode ($GTCFR2[SCAS] = 1$)
 In this mode, all four 16-bit timers can be internally cascaded to form a 64-bit counter, as shown in [Figure 5-59](#).
 If $GTCFR2[SCAS] = 1$, the all four 16-bit timers function as a 64-bit timer with a cascaded 32-bit GTRFR, GTCPR, and GTCNR. In this case, registers GTMDR1, GTMDR2, GTMDR3, and GTCFR1 are ignored, and the modes and functions are defined using GTMDR4 and GTCFR2 only. The capture are controlled from TIN4, and the interrupts are generated from GTEVR4. When working in the super-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with two 32-bit bus cycles.

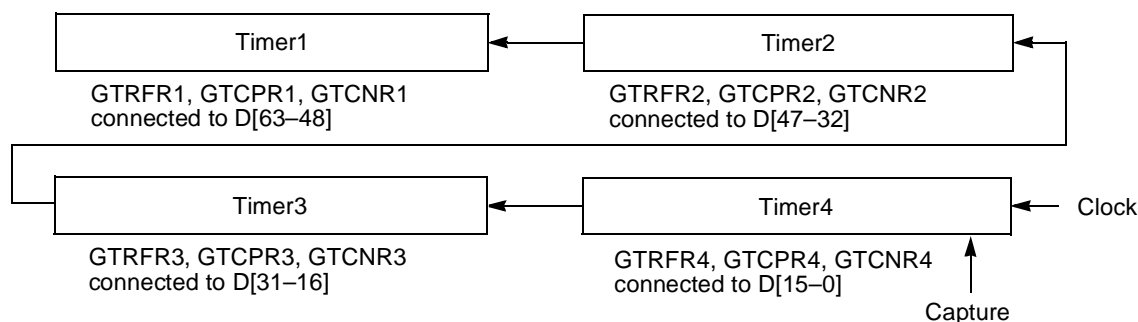


Figure 5-59. Timers Super-Cascaded Mode Block Diagram

5.8.7 Initialization/Application Information

5.8.7.1 Programming Guidelines

5.8.7.1.1 GTM Registers

The following initialization sequence of GTM is recommended:

- Write to $GTCFR_n$ in order to reset, to stop or to configure the appropriate timer's operation: cascaded timers configuration, gate mode configuration.
- Write to $GTPSR_n[PPS]$ fields in order to program the appropriate timer's clock primary prescaler.
- Write to $GTMDR_n$ in order to choose an input clock, to program the secondary prescaler and to set a desirable appropriate timer's operational mode.

NOTE

Erratic behavior may occur if $GTCFR_n$ and $GTPSR$ are not initialized before the $GTMDR$. Only $GTCFR_n[RST_n]$ can be modified at any time

- Clear $GTEVR_n[REF]$ and $GTEVR_n[CAP]$ by writing 1s in order to clear the previous events.
- Write to $GTRFR$ and to $GTCNR_n$ according to appropriate timer's $GTMDR_n$ programming.

NOTE

A write cycle to a $GTCNR_n[CNV]$ fields sets the register to the written value, causing its corresponding primary and secondary prescalers, ($GTPSR_n[PPS]$ and $GTMDR_n[SPS]$), to be reset.

- Write to $GTCFR_n[STP_n]$ and to $GTCFR_n[RST_n]$ in order to initialize the appropriate timer's operation.

5.9 Power Management Control (PMC)

The device provides a power management control (PMC) unit, which enables the device to smoothly enter and exit low power modes. Low power modes may be used when internal units in the device temporarily or permanently do not perform any action.

The device uses one or more of the following methods for power saving:

- Dynamic power management
- Shutting down unused blocks
- Software-controlled power-down states

5.9.1 External Signal Description

Table 5-77 describes the power management signals.

Table 5-77. System Control Signals—Detailed Signal Descriptions

Signal	I/O	Description						
$\overline{\text{QUIESCE}}$	O	Quiesce state. Indicates that the processor system and PowerPC core are in low power state.						
		<table border="1"> <thead> <tr> <th>State Meaning</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Asserted</td> <td>The system and PowerPC core are in low power state.</td> </tr> <tr> <td>Negated</td> <td>The system and PowerPC core are not in low power state.</td> </tr> </tbody> </table>	State Meaning	Description	Asserted	The system and PowerPC core are in low power state.	Negated	The system and PowerPC core are not in low power state.
State Meaning	Description							
Asserted	The system and PowerPC core are in low power state.							
Negated	The system and PowerPC core are not in low power state.							
		<table border="1"> <thead> <tr> <th>Timing</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>The timing between a quiesce request from the PowerPC core and the assertion of the external indication or between negation of the core's quiesce request and negation of the external indication depends on the current state of the internal system units and may vary accordingly.</td> </tr> </tbody> </table>	Timing	Description	The timing between a quiesce request from the PowerPC core and the assertion of the external indication or between negation of the core's quiesce request and negation of the external indication depends on the current state of the internal system units and may vary accordingly.			
Timing	Description							
The timing between a quiesce request from the PowerPC core and the assertion of the external indication or between negation of the core's quiesce request and negation of the external indication depends on the current state of the internal system units and may vary accordingly.								

5.9.2 PMC Memory Map/Register Definition

Table 5-78 shows the memory map for the power management controller registers.

Table 5-78. Power Management Controller Registers Memory Map

Offset	Register	Access	Reset	Section/Page
0x00B00	Power management controller configuration register (PMCCR)	R/W	All zeros	5.9.2.1/5-73
0x00B04	Power management controller event register (PM CER)	R/W	All zeros	5.9.2.2/5-74
0x00B08	Power management controller mask register (PM C MR)	R/W	All zeros	5.9.2.3/5-75
0x00B0C–0x00BFC	Reserved	—	—	—

5.9.2.1 Power Management Controller Configuration Register (PMCCR)

The power management controller configuration register (PMCCR), shown in Figure 5-60, controls whether only the PowerPC core will enter low power state upon quiesce request or additional parts of the device will also enter low power state.

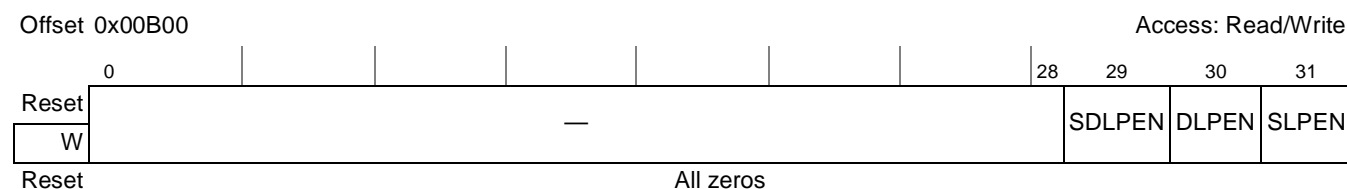


Figure 5-60. Power Management Controller Configuration Register

Table 5-5 defines the bit fields of PMCCR.

Table 5-79. PMCCR Bit Settings

Bits	Name	Description
0–28	—	Reserved. Write has no effect, read returns 0.
29	SDLPEN	Secondary DDR SDRAM low power enable 0 The secondary DDR SDRAM memory controller is prevented from entering low power state. 1 The secondary DDR SDRAM memory controller will enter low power state when the rest of the system enters low power state, according to SLPEN setting. Secondary DDR SDRAM will enter self-refresh mode (if enabled by DDR_SDRAM_CFG[SREN] memory controller register) and clocks (MCK n) are shut off. This bit is cleared when the MPC8360E exits from low power state. Note that setting this bit without setting SLPEN has no effect.
30	DLPEN	DDR SDRAM low power enable 0 The DDR SDRAM memory controller is prevented from entering low power state. 1 The DDR SDRAM memory controller will enter low power state when the rest of the system enters low power, according to SLPEN setting. DDR SDRAM will enter self-refresh mode (if enabled by DDR_SDRAM_CFG[SREN] memory controller register) and DDR clocks (MCK n) are shut off. This bit is cleared when the device exits from low power state. Note that setting this bit without setting SLPEN has no effect.
31	SLPEN	System low power enable 0 The system is prevented from entering low power state. 1 The system will enter low power state when a quiesce request from the PowerPC core arrives. This bit is cleared when the device exits from low power state.

5.9.2.2 Power Management Controller Event Register (PM CER)

The power management controller event register (PM CER), shown in Figure 5-61, indicates with the PMCI bit that the power management controller has detected a wake-up event, that the system is not in idle state anymore, and that the device should exit low power state. If PMCMR[PM CIE] is set, the PMC interrupt request to the PowerPC core is driven.



Figure 5-61. Power Management Controller Event Register

Table 5-80 defines the bit fields of PMCER.

Table 5-80. PMCER Bit Settings

Bits	Name	Description
0–30	—	Reserved. Write has no effect, read returns 0.
31	PMCI	Power management controller interrupt. When set, indicates that the power management controller has detected that the system is not in idle state anymore, and that the device is required to exit low power state. If PMCMR[PMCIIE] is set, the PMC interrupt request to the PowerPC core is driven, causing the PowerPC core to exit its low power state. PMCI can be cleared by writing a 1 to it (writing zero has no effect).

5.9.2.3 Power Management Controller Mask Register (PMCMR)

The power management controller mask register (PMCMR), shown in Figure 5-62, controls through the PMCIIE bit whether the PMC interrupt request to the PowerPC core is enabled. The PMC interrupt request causes the PowerPC core to exit its low power state before any transaction on the system bus occurs.



Figure 5-62. Power Management Controller Mask Register

Table 5-81 defines the bit fields of PMCMR.

Table 5-81. PMCMR Bit Settings

Bits	Name	Description
0–30	—	Reserved. Write has no effect, read returns 0.
31	PMCIIE	Power management controller interrupt enable. 0 PMC interrupt request (PMCI) is disabled. 1 PMC interrupt request (PMCI) is enabled.

NOTE

The user is also required to enable the PMC interrupt in the programmable interrupt controller by setting SIMSR_L[PMC].

5.9.3 Functional Description

The device has features to minimize power consumption at several levels. Dynamic power management locally minimizes power consumption when a block is idle. Software can also shut down clocks to

individual blocks when they are not needed through a memory-mapped register in the clock unit (SCCR). Additionally, software running on the PowerPC core can access the core's SPRs to put the device into doze, nap, or sleep power down states. Finally, software can access the PMCCR register to enable the device to go to low power state whenever the PowerPC core enters nap or sleep states. These power management features are described in further detail in this section.

5.9.3.1 Dynamic Power Management

Many blocks in the device can dynamically turn off clocks within the block when sections of the block are idle. This feature is always enabled and occurs automatically.

5.9.3.2 Shutting Down Unused Blocks

As described in [Section 4.5.2.3, "System Clock Control Register \(SCCR\),"](#) SCCR provides a way to shut down certain functional blocks within the device when they are not needed in a particular system. SCCR can be written by the PowerPC core or by an external master. Powering down a block in this way turns off all clocks to that block. It does not remove power. It is required that the SCCR is written to shut down a certain functional block only when that block is idle.

NOTE

Functional blocks disabled using SCCR cannot respond to configuration accesses. Any access to configuration, control, and status registers of a disabled block is a programming error.

5.9.3.3 Software-Controlled Power-Down States

PowerPC software can place the core in doze, nap, or sleep power-down states by writing to HID0 in the core, as described in detail in the section "Hardware Implementation Register 0 (HID0)," of the *e300 PowerPC Core Reference Manual*. In addition, if PMCCR[SLPEN] is set when the PowerPC core request to enter nap or sleep modes, it will also cause the system internal logic units to enter low power mode.

5.9.3.3.1 Entering Low Power States—Core-Only Mode

Entering Doze mode is controlled only by the e300 PowerPC core itself, and does not involve the power management controller or other blocks. For a more detailed description, see [Table 7-1](#).

Entering Nap or Sleep modes occurs by writing to HID0 in the core, causing the core to make a quiesce request to the power management controller while PMCCR[SLPEN] is cleared. The core is immediately enabled to enter low power state, regardless of the system status. Note that since the core does not snoop the bus in this mode, it is the user's responsibility to keep the cache coherent. Other device peripheral and internal units continue to operate in full-on mode while the core is in low power state in this mode.

5.9.3.3.2 Entering Low Power States—Core and System Mode

Core and system mode is achieved when the core makes a quiesce request to the power management controller after PMCCR[SLPEN] is set. To preserve cache coherency and otherwise avoid loss of system

state, the core's transition to low-power modes is coordinated with other functional blocks. The power management controller allows the core to enter power down mode only when the rest of the system is idle.

When the power management controller detects that the internal system bus is idle, and there are no outstanding transactions, it signals the internal logic units to enter low power state.

If PMCCR[DLPEN] and/or PMCCR[SDLPEN] is set, the DDR SDRAM and/or secondary DDR SDRAM is first set to self-refresh mode (if enabled by DDR_SDRAM_CFG[SREN] memory controller register) before the memory controller stops driving refresh commands. Self-refresh mode guarantees that the memory content will remain valid while the memory controller and its clocks are off. The DDR clocks are then disabled. Finally the DDR SDRAM memory controller enters low power state and acknowledges the power management controller.

The power management controller then signals the core and acknowledges its request to enter power down mode. Finally the QUIESCE output signal is asserted.

5.9.3.4 Exiting Core and System Low Power States

The device can exit low power state and return to full-on mode for one of the following reasons:

- The core internal time base unit invokes a request to exit low power state.
- The core has received an interrupt request.
- The power management controller has detected that the system is not idle and there are outstanding requests for transactions on the internal bus.

The actions taken to exit low power state depend on the mode and whether the system or only the core are in this state. The following sections describe the various scenarios.

5.9.3.4.1 Exiting Low Power States—Core-Only Mode

Exit from Doze mode is controlled only by the core itself and does not involve the power management controller or other blocks in the device. For a detailed description, see [Table 7-1](#).

Nap or Sleep modes are exited when the core has received an interrupt request, or according to the internal time base unit of the core (Nap mode only). The source of the interrupt can be an internal block or external signal. When the core returns to full-on state, it signals to the power management controller that it is ready and is immediately acknowledged to access the rest of the system.

5.9.3.4.2 Exiting Low Power States—Core and System Mode

The power management controller decides to exit low power state when it detects that the system is not idle anymore. The device may exit idle state when one of the peripheral interfaces makes a request to access the internal bus or when the core returns to full-on state, as described above, and makes a request to access the internal bus. For example, the QUICC Engine block receives an Ethernet frame, and requires to store it on the DDR SDRAM memory.

If the particular DDR SDRAM memory controller is in low power state (PMCCR[DxLPEN] was set when entering low power state), the power management controller initially enables the DDR SDRAM memory controller. DDR SDRAM clocks (MCK_n) are enabled and the memory controller exit self-refresh and returns to auto-refresh mode.

The power management controller then enables other internal units and interrupts the PowerPC core. When all internal units, including the core, are ready, the power management controller enables the device to return to full-on state, negate the $\overline{\text{QUIESCE}}$ output, and clear $\text{PMCCR}[\text{SLPEN}]$. Outstanding requests for transactions are now granted to execute on the internal bus.

NOTE

Software is required to enable PMCI interrupt by setting $\text{PMCMR}[\text{PMCIE}]$, otherwise exiting from low power state is not possible.

NOTE

It is the software's responsibility to clear $\text{PMCMR}[\text{PMCI}]$.

5.9.4 Initialization/Application Information

5.9.4.1 Core Disable in Low Power Mode

If the device is required to operate with the core permanently disabled, the following steps must be taken:

1. Initialize the device with the core enable.
2. Clear $\text{PMCCR}[\text{SLPEN}]$ and disable the core time base unit by clearing $\text{SPCR}[\text{TBEN}]$. See [Section 5.4.3.4, "System Priority and Configuration Register \(SPCR\),"](#) for more information.
3. The e300 core enters low power state by accessing the HID0 register.
4. Set $\text{ACR}[\text{COREDIS}]$ in the system arbiter register with an external master (that is, PCI). This steers all device interrupts to the $\overline{\text{PCI_INTA}}$ so the core cannot receive any interrupt requests.

NOTE

Make sure that the core cannot receive any interrupt requests during the time interval between setting HID0 and setting $\text{ACR}[\text{COREDIS}]$. This can be achieved if the rest of the system is idle (during the initialization).

By following this flow, the e300 core remains in low power state while the rest of the system is operational, and does not get out of this state as a result of any interrupt or time-based event.

Part III

Core and I/O Interfaces

Part III describes the core and I/O interfaces of the MPC8360E integrated processor. The following chapters are included:

- [Chapter 6, “Arbiter and Bus Monitor,”](#) provides an overview of the arbiter in the MPC8360E device. It also describes configuration, control, and status registers of the arbiter.
- [Chapter 7, “e300 Processor Core Overview,”](#) provides an overview of the basic functionality of the e300 processor core and briefly describes how the functional units interact.
- [Chapter 8, “Integrated Programmable Interrupt Controller \(IPIC\),”](#) describes the IPIC interrupt protocol, various types of interrupt sources controlled by the IPIC unit, and the IPIC registers with some programming guidelines. It also provides a definition of the external interrupt signals and their functions. In addition, the interrupt configuration, control, and status registers are described in this chapter.
- [Chapter 9, “DDR Memory Controller,”](#) describes the DDR SDRAM memory controller of the MPC8360E. This fully programmable controller supports most DDR memories available today, including both buffered and unbuffered devices. The 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. Special features like ECC error injection support rapid system debug.
- [Chapter 10, “Local Bus Controller,”](#) describes the local bus controller (LBC) of the MPC8360E. It describes the external signals and the memory-mapped registers as well as a functional description of the general-purpose chip-select machine (GPCM), synchronous DRAM (SDRAM) machine, and user-programmable machines (UPMs) of the LBC. Also, it includes an initialization and applications information section with many specific examples of its use.
- [Chapter 11, “Sequencer,”](#) describes how the I/O sequencer switches transactions among its ports, using a buffer pool to minimize blocking. It also provides address translation on outbound PCI transactions.
- [Chapter 12, “DMA/Messaging Unit,”](#) describes the four-channel high speed general-purpose DMA controller of the MPC8360E. The channels share buffer space in the IOS to facilitate the gathering and sending of data. The DMA/messaging unit supports communication between two processors on different buses, for example, a local processor and a processor on a PCI bus. This communication unit operates with generic messages and doorbell registers. This block also provides a DMA controller that transfers blocks of data independent of the local processor or PCI hosts.
- [Chapter 13, “PCI Bus Interface,”](#) describes the PCI interface, which complies with the *PCI Local Bus Specification*, Rev. 2.3. This chapter provides a basic description of PCI bus operations. The specific emphasis is directed at how this device implements the PCI specification.

- [Chapter 14, “Security Engine \(SEC\) 2.4,”](#) describes the SEC 2.4 that is designed to offload computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the e300 core of the MPC8360E. It is optimized to process all the algorithms associated with IPsec, IKE, SSL/TLS, iSCSI, SRTP, and IEEE Std 802.11i.™.
- [Chapter 15, “I2C Interfaces,”](#) describes the inter-IC (IIC or I²C) bus controller of the MPC8360E. This synchronous, serial, bidirectional, multiple-master bus allows two-wire connection of devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The MPC8360E powers up in boot sequencer mode which allows the I²C controller to initialize configuration registers.
- [Chapter 16, “DUART,”](#) describes the (dual) universal asynchronous receiver/transmitters (UARTs) which feature a PC16552D-compatible programming model. These independent UARTs are provided specifically to support system debugging.
- [Chapter 17, “JTAG/Testing Support,”](#) describes the joint test action group (JTAG) interface of the MPC8360E to facilitate boundary-scan testing. The JTAG interface complies with the IEEE Std. 1149.1™ boundary-scan specification.
- [Chapter 18, “Delay Lock Loop \(DLL\),”](#) describes the theory of operation of the delay lock loop (DLL) module in the integrated device. Additionally, the configuration, control, and status registers are described.

Chapter 6

Arbiter and Bus Monitor

This chapter describes operation theory of the arbiter in the device. In addition, it describes configuration, control, and status registers of the arbiter.

6.1 Arbiter Overview

The arbiter is responsible for providing coherent system bus arbitration. It tracks all address and data tenures and provides all the arbitration signals to masters and slaves. In addition, it monitors the bus and reports on errors and protocol violations.

The arbiter includes the following features:

- Supports a programmable pipeline depth (from 1 to 4)
- Supports four levels of priority for bus arbitration
- Supports repeat-request mode: number of programmable consecutive transactions from the same master (up to eight transactions)
- Supports data streaming operations
- Supports programmable address bus parking mode: disable, park to last bus owner, park to software selected master
- Claims address only, reserved and illegal transaction types, report on it and can raise maskable interrupt
- Provides timers for address tenure time out and data tenure time out detection and can issue maskable interrupt, if any timer expired
- Reports on transfer error and can issue maskable interrupt
- Can issue regular or machine check interrupt for each type of error event (programmable)

6.1.1 Coherent System Bus Overview

The coherent system bus is the central bus of the device. Any data transaction from master to slave in the device passes through the coherent system bus. The device coherent system bus supports pipelined transactions. It has independent address and data tenures. Pipeline depth determines the number of address tenures that can be started before the first data tenure is finished.

Basic burst size is equal to cache line length of the core, which is 32 bytes. Using repeat request mode enables up to eight consecutive bursts to be executed by the same master. Maximum number of consecutive transactions can be limited by programming arbiter configuration register. See [Section 6.2.1, “Arbiter Configuration Register \(ACR\)”](#) for more details.

NOTE

Write accesses to different interfaces are not guaranteed to finish in order.

6.2 Arbiter Memory Map/Register Definition

Table 6-1 shows the memory map for arbiter’s configuration, control and status registers.

Table 6-1. Arbiter Register Map

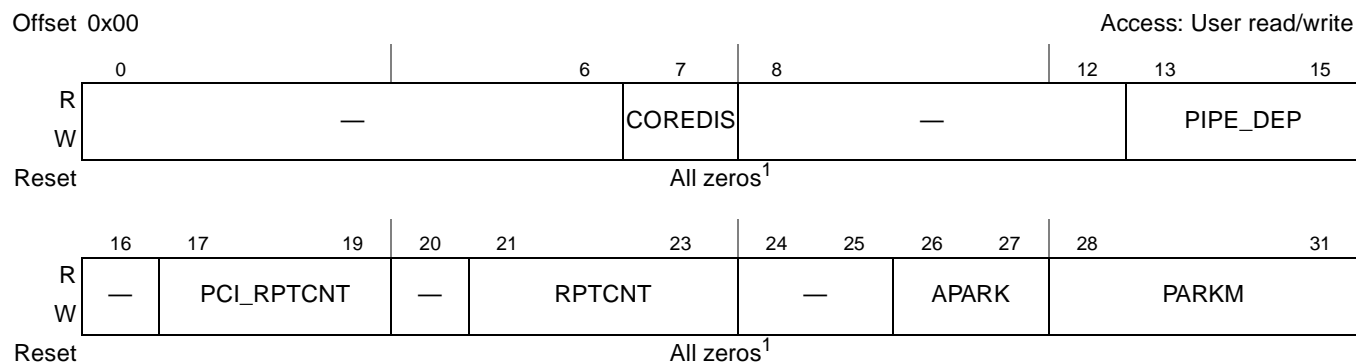
Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x00	Arbiter configuration register (ACR)	R/W	All zeros/ 0x0010_0000 ¹	6.2.1/6-2
0x04	Arbiter timers register (ATR)	R/W	0xFFFF_FFFF	6.2.2/6-4
0x0C	Arbiter event register (AER)	w1c	All zeros	6.2.3/6-5
0x10	Arbiter interrupt definition register (AIDR)	R/W	All zeros	6.2.4/6-6
0x14	Arbiter mask register (AMR)	R/W	All zeros	6.2.5/6-7
0x18	Arbiter event attributes register (AEATR)	R	All zeros ²	6.2.6/6-8
0x1C	Arbiter event address register (AEADR)	R	All zeros ²	6.2.7/6-9
0x20	Arbiter event response register (AERR)	R/W	All zeros	6.2.8/6-10

¹ Reset value is determined from the core PLL configuration of the reset configuration word. See Chapter 4, “Reset, Clocking, and Initialization,” for details.

² The registers AEATR and AEADR are affected only by the assertion of $\overline{\text{PORESET}}$

6.2.1 Arbiter Configuration Register (ACR)

Arbiter configuration register (ACR) defines the arbiter modes and parked master on the bus. Figure 6-1 shows the fields of ACR.



¹ Note that the reset value of COREDIS and bits 10–11 are determined from reset configuration word. (See Section 4.3.2, “Reset Configuration Words,” for more details on reset configuration word.)

Figure 6-1. Arbiter Configuration Register (ACR)

Table 6-2 describes ACR fields.

Table 6-2. ACR Field Descriptions

Bits	Name	Description
0–6	—	Write reserved, read = 0
7	COREDIS	Core disable. Specifies whether CPU is disabled. When CPU is disabled, it cannot be granted on the bus by the arbiter. After reset, this bit receives its value from the reset configuration bit of COREDIS and can be configured by software. Also, if boot source is boot sequencer, COREDIS must be set to 1 at reset and the last transaction of the boot sequencer must set COREDIS to 0, if CPU enable is needed. 0 CPU enabled. 1 CPU disabled.
8–9	—	Write reserved, read = 0
10–11	—	Reserved. Write should preserve reset value. The reset value is a function of the core PLL configuration, which is part of the reset configuration word. When the core is set to operate at 1:1 or 3:2 bus clock, these bits are set to '01' during reset; otherwise, they are set to '00'.
12	—	Write reserved, read = 0
13–15	PIPE_DEP	Pipeline depth (number of outstanding transactions). 000 Pipeline depth 1 (1 outstanding transaction) 001 Pipeline depth 2 (2 outstanding transactions) 010 Pipeline depth 3 (3 outstanding transactions) 011 Pipeline depth 4 (4 outstanding transactions) 1xx Reserved
16	—	Write reserved, read = 0
17–19	PCI_RPTCNT	PCI repeat count. Specifies the maximum number of consecutive transactions, that PCI master can perform, using $\overline{\text{REPEAT}}$ request mode. 000 One consecutive transaction ($\overline{\text{REPEAT}}$ request mode disable) 001 Two consecutive transactions 010 Three consecutive transactions 011 Four consecutive transactions 100 Five consecutive transactions 101 Six consecutive transactions 110 Seven consecutive transactions 111 Eight consecutive transactions
20	—	Write reserved, read = 0
21–23	RPTCNT	Repeat count. Specifies the maximum number of consecutive transactions, that any master (except PCI) can perform, using $\overline{\text{REPEAT}}$ request mode. 000 1 consecutive transactions ($\overline{\text{REPEAT}}$ request mode disable) 001 2 consecutive transactions 010 3 consecutive transactions 011 4 consecutive transactions 100 5 consecutive transactions 101 6 consecutive transactions 110 7 consecutive transactions 111 8 consecutive transactions Note: It is recommended not to program this field for more than four consecutive transactions.
24–25	—	Write reserved, read = 0

Table 6-2. ACR Field Descriptions (continued)

Bits	Name	Description
26–27	APARK	Address parking. Specifies arbiter bus parking mode. 00 Park to master. Arbiter parks the address bus to the master, that is selected by numeric value of PARKM field. 01 Park to last owner. Arbiter parks the address bus to last bus owner. 10 Disable. Arbiter does not assert \overline{BG} to any master, if no \overline{BR} is present. 11 Reserved
28–31	PARKM	Parking master. 0000 e300 core 0001 PCI, DMA 0010 Reserved 0011 QUICC Engine block 0100 Encryption core 0101–1111Reserved

6.2.2 Arbiter Timers Register (ATR)

The arbiter timers register (ATR) defines the arbiter address time out (ATO) and data time out (DTO) values. [Figure 6-2](#) shows the fields of ATR.

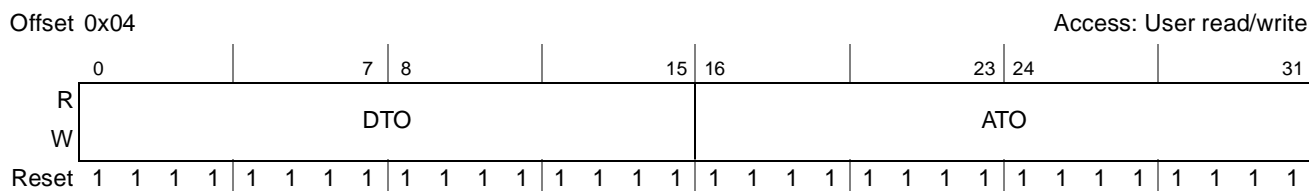


Figure 6-2. Arbiter Timers Register (ATR)

[Table 6-3](#) describes ATR fields.

Table 6-3. ATR Field Descriptions

Bits	Name	Description
0–15	DTO	Data time out. Specifies the time-out period for the data tenure. The granularity of this field is 128 bus clocks. The maximum value is 8355840 coherent system bus clocks. Data time_out occurs if the data tenure does not end before the specified time-out period. When DTO = n, the timeout cycle is n*128. 0000 Reserved 0001 128 clock cycles 0002 256 clock cycles 0003 384 clock cycles ... FFFF 8355840 clock cycles
16–31	ATO	Address time out. Specifies the time-out period for the address tenure. The granularity of this field is 128 bus clocks. Maximum value is 8355840 coherent system bus clocks. Address time-out occurs if the address tenure did not end before the specified time-out period. When ATO = n, the timeout cycle is n*128. 0000 Reserved 0001 128 clock cycles 0002 256 clock cycles 0003 384 clock cycles ... FFFF 8355840 clock cycles

6.2.4 Arbiter Interrupt Definition Register (AIDR)

Arbiter interrupt definition register (AIDR) determines the interrupt that responds to different error conditions. Setting a bit defines the corresponding interrupt as MCP interrupt; clearing a bit defines the corresponding interrupt as regular interrupt. Figure 6-4 shows the fields of AIDR.

Offset 0x10

Access: User read/write

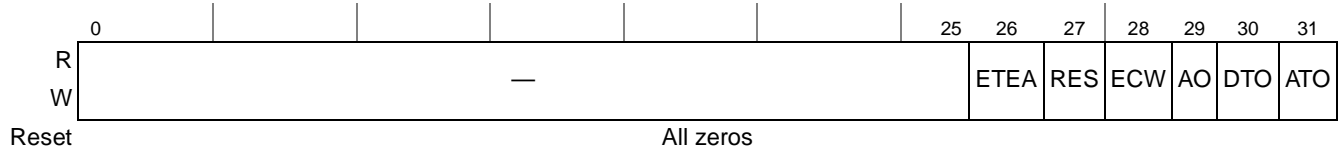


Figure 6-4. Arbiter Interrupt Definition Register (AIDR)

Table 6-5 describes AIDR fields.

Table 6-5. AIDR Field Descriptions

Bits	Name	Description
0–25	—	Write reserved, read = 0
26	ETEA	Transfer error. Detection of transfer error by one of the slaves interrupt definition. 0 Detection of transfer error by one of the slaves causes regular interrupt. 1 Detection of transfer error by one of the slaves causes MCP interrupt.
27	RES	Reserved transfer type. Transaction with reserved transfer type interrupt definition. 0 Transaction with reserved transfer type causes regular interrupt. 1 Transaction with reserved transfer type causes MCP interrupt.
28	ECW	External control word transfer type. Transaction with external control word transfer type interrupt definition. 0 Transaction with external control word transfer type causes regular interrupt. 1 Transaction with external control word transfer type causes MCP interrupt.
29	AO	Address only transfer type. Transaction with address only transfer type interrupt definition. 0 Transaction with address only transfer type causes regular interrupt. 1 Transaction with address only transfer type causes MCP interrupt.
30	DTO	Data time out. Data tenure time out interrupt definition. 0 Data tenure time out causes regular interrupt. 1 Data tenure time out causes MCP interrupt.
31	ATO	Address time out. Address tenure time out interrupt definition. 0 Address tenure time out causes regular interrupt. 1 Address tenure time out causes MCP interrupt.

6.2.5 Arbiter Mask Register (AMR)

Arbiter mask register (AMR) is used to mask interrupts. Setting a mask bit enables the corresponding interrupt; clearing a bit masks it. Regular interrupts and MCP interrupts can be masked by the AMR register, except for scenarios that results in a transfer error when the master ID is 0. Figure 6-5 shows the fields of AMR.

Offset 0x14

Access: User read/write



Figure 6-5. Arbiter Mask Register (AMR)

Table 6-6 describes AMR fields.

Table 6-6. AMR Field Descriptions

Bits	Name	Description
0–25	—	Write reserved, read = 0
26	ETEA	Transfer error. Detection of transfer error by one of the slaves interrupt mask bit. 0 Detection of transfer error by one of the slaves interrupt disabled. 1 Detection of transfer error by one of the slaves interrupt enabled. Note: Generation of regular or MCP interrupt is not possible for scenarios that results in a transfer error if the core master ID is 0.
27	RES	Reserved transfer type. Transaction with reserved transfer type interrupt mask bit. 0 Transaction with reserved transfer type interrupt disabled. 1 Transaction with reserved transfer type interrupt enabled.
28	ECW	External control word transfer type. Transaction with external control word transfer type interrupt mask bit. 0 Transaction with external control word transfer type interrupt disabled. 1 Transaction with external control word transfer type interrupt enabled.
29	AO	Address only transfer type. Transaction with address only transfer type interrupt mask bit. 0 Transaction with address only transfer type interrupt disabled. 1 Transaction with address only transfer type interrupt enabled.
30	DTO	Data time out. Data tenure time out interrupt mask bit. 0 Data tenure time out interrupt disabled. 1 Data tenure time out interrupt enabled.
31	ATO	Address time out. Address tenure time out interrupt mask bit. 0 Address tenure time out interrupt disabled. 1 Address tenure time out interrupt enabled.

6.2.6 Arbiter Event Attributes Register (AEATR)

Arbiter event attributes register (AEATR) reports the type of transaction that causes error, which is specified in the event register. See [Section 6.2.3, “Arbiter Event Register \(AER\),”](#) for more information. AEATR is cleared only by power-on reset. The attributes of the first error event are stored. Note that this means that AEATR does not change its value when AER is not clear. As AEATR is not affected by soft or hard reset, software can read this register and determine the cause of the bus failure, even if the failure caused a deadlock situation. Refer to [Section 6.4.2, “Error Handling Sequence,”](#) for more information.

Figure 6-6 shows the fields of AEATR.

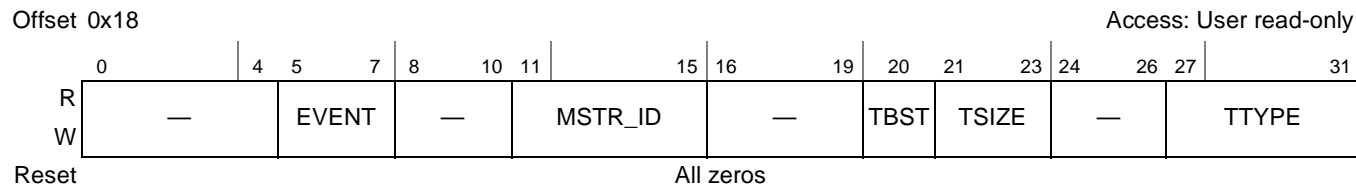


Figure 6-6. Arbiter Event Attributes Register (AEATR)

Table 6-7 describes AEATR fields.

Table 6-7. AEATR Field Descriptions

Bits	Name	Description
0–4	—	Write reserved, read = 0
5–7	EVENT	Event type. 000 Address time out 100 Reserved transfer type 001 Data time out 101 Transfer error 010 Address only transfer type 11c Reserved 011 External control word transfer type
8–10	—	Write reserved, read = 0
11–15	MSTR_ID	Master Id. 00000 e300 core data transaction 01011 Reserved 00001 Reserved 01100 Reserved 00010 e300 core instruction fetch 01101 PCI 00011–00111 Reserved 01110 Reserved 01000 Encryption core 01111 DMA 01001 I ² C (boot sequencer) 10000–11111 QUICC Engine block as follows: 01010 JTAG 100xy are used by the QUICC Engine block as follows: x = MSTR_ID[3] Least significant bit of SNUM. y = MSTR_ID[4] CETM bit from RBMR/TBMR registers. RBMR/TBMR registers are described in protocol chapters. 101xx Reserved 11xxx Reserved Note: Master Id reflects the source of transaction and is used for debug purposes.
16–19	—	Write reserved, read = 0
20	TBST	Transfer burst. 0 Burst transaction. Transfer size is greater than 8 bytes 1 Single-beat transaction. Transfer size is up to 8 bytes

Table 6-7. AEATR Field Descriptions (continued)

Bits	Name	Description
21–23	TSIZE	Transfer size. Transfer size encoding depends on the value of the field TBST. TBST = 1: 001 1 byte 010 2 bytes 011 3 bytes 100 4 bytes 101 5 bytes 110 6 bytes 111 7 bytes 000 8 bytes TBST = 0: 000 16 bytes 001 24 bytes 010 32 bytes 011–111 Reserved
24–26	—	Write reserved, read = 0
27–31	TTYPE	Transfer type. 00000 Address-only 00001 Address-only 00010 Single-beat or burst write 00011 Reserved 00100 Address-only 00101 Reserved 00110 Burst write 00111 Reserved 0100x Address-only 0101x Single-beat or burst read 0110x Address-only 01110 Burst read 01111 Reserved 10000 Address-only 1XX01 Reserved 10010 Single-beat write 1XX11 Reserved 10100 ecowx —Illegal single-beat write 10110 Reserved 11000 Address-only 11010 Single-beat or burst read 11100 eciw x—Illegal single-beat read 11110 Burst read

6.2.7 Arbiter Event Address Register (AEADR)

Arbiter event address register (AEADR) reports the address of transaction that causes the error, which is specified in the event register. See [Section 6.2.3, “Arbiter Event Register \(AER\),”](#) for more information. AEADR is cleared only by power-on reset. The address of the first error event is stored. Note that this means that AEADR does not change its value when AER is not clear. As AEADR is not effected by soft or hard reset, software can read this register and determine the cause of the bus failure, even if the bus failure had caused a deadlock situation. Refer to [Section 6.4.2, “Error Handling Sequence,”](#) for more information.

Figure 6-7 shows the fields of AEADR.


Figure 6-7. Arbiter Event Address Register (AEADR)

Table 6-8 describes AEADR fields.

Table 6-8. AEADR Field Descriptions

Bits	Name	Description
0–31	ADDR	Address of the event reported in AEATR register. See Section 6.2.6, “Arbiter Event Attributes Register (AEATR),” for more information.

6.2.8 Arbiter Event Response Register (AERR)

Arbiter event response register (AERR) determines whether different error conditions cause interrupt or reset request. Setting a bit defines the corresponding error condition to cause reset request; clearing a bit defines the corresponding error condition to cause interrupt. [Figure 6-8](#) shows the fields of AERR.

Offset 0x20

Access: User read/write

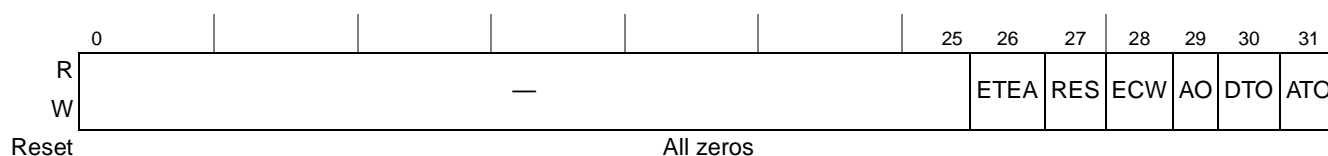


Figure 6-8. Arbiter Event Response Register (AERR)

Table 6-9 describes AERR field.

Table 6-9. AERR Field Descriptions

Bits	Name	Description
0–25	—	Write reserved, read = 0
26	ETEA	Transfer error. Detection of transfer error by one of the slaves event response. 0 Detection of transfer error by one of the slaves causes interrupt. 1 Detection of transfer error by one of the slaves causes reset request.
27	RES	Reserved transfer type. Transaction with reserved transfer type interrupt definition. 0 Transaction with reserved transfer type causes interrupt. 1 Transaction with reserved transfer type causes reset request.
28	ECW	External control word transfer type. Transaction with external control word transfer type interrupt definition. 0 Transaction with external control word transfer type causes interrupt. 1 Transaction with external control word transfer type causes reset request.
29	AO	Address only transfer type. Transaction with address only transfer type interrupt definition. 0 Transaction with address only transfer type causes interrupt. 1 Transaction with address only transfer type causes reset request.
30	DTO	Data time out. Data tenure time out interrupt definition. 0 Data tenure time out causes interrupt. 1 Data tenure time out causes reset request.
31	ATO	Address time out. Address tenure time out interrupt definition. 0 Address tenure time out causes interrupt. 1 Address tenure time out causes reset request.

6.3 Functional Description

The following sections describe arbiter functionality: arbitration policy and bus error detection.

6.3.1 Arbitration Policy

The arbitration process involves the masters and the arbiter. Masters arbitrate on the privilege to own an address tenure. For data tenures, the arbiter uses the same order of transactions as address tenures.

Figure 6-9 shows the interface signals between the arbiter and masters that are involved in address bus arbitration.

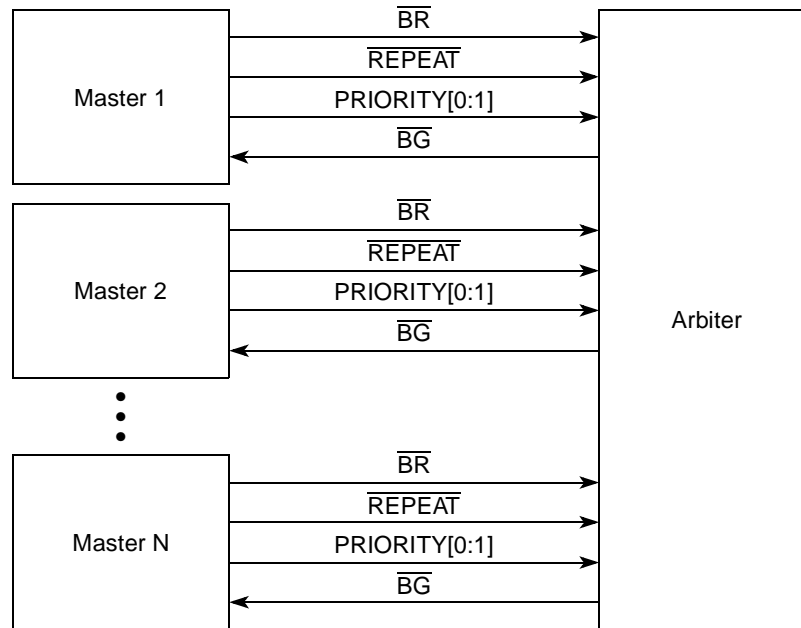


Figure 6-9. Address Bus Arbitration

A master has to acquire address bus ownership before it starts any transaction. The master asserts its own bus request signal along with the arbitration attribute signals \overline{REPEAT} & $PRIORITY[0:1]$. The arbiter later asserts the corresponding address bus grant signal to the requesting master depending on the system states and arbitration scheme. See Section 6.3.1.1, “Address Bus Arbitration with $PRIORITY[0:1]$,” for details on arbitration scheme. When address bus grant is received the master can start the address tenure.

6.3.1.1 Address Bus Arbitration with $PRIORITY[0:1]$

Whenever a master asserts its bus request to acquire address bus ownership, it can drive its $PRIORITY[0:1]$ signals to indicate request priority. The master would be served sooner because of its higher priority level. The arbiter takes this extra information into consideration in order to yield better service for a higher priority request than a lower priority request. Therefore, the arbiter operates according to the following priority based arbitration scheme:

1. For every priority level a fair arbitration scheme is used (a simple round robin scheme)
2. For every priority level other than 0, one place is reserved as a place holder for lower level arbitration rings.

3. Each master can change its priority level at any time.

Figure 6-10 shows an example of priority-based arbitration algorithm with four priority levels. In this example, if all masters request the bus continuously, the following order of bus grants occurs with the specific bandwidth:

- M6 gets 1/2 of the bus bandwidth
- M4 and M5 each gets 1/6 of the bus bandwidth
- M0 and M3 each gets 1/18 of the bus bandwidth
- M1 and M2 each gets 1/36 of the bus bandwidth

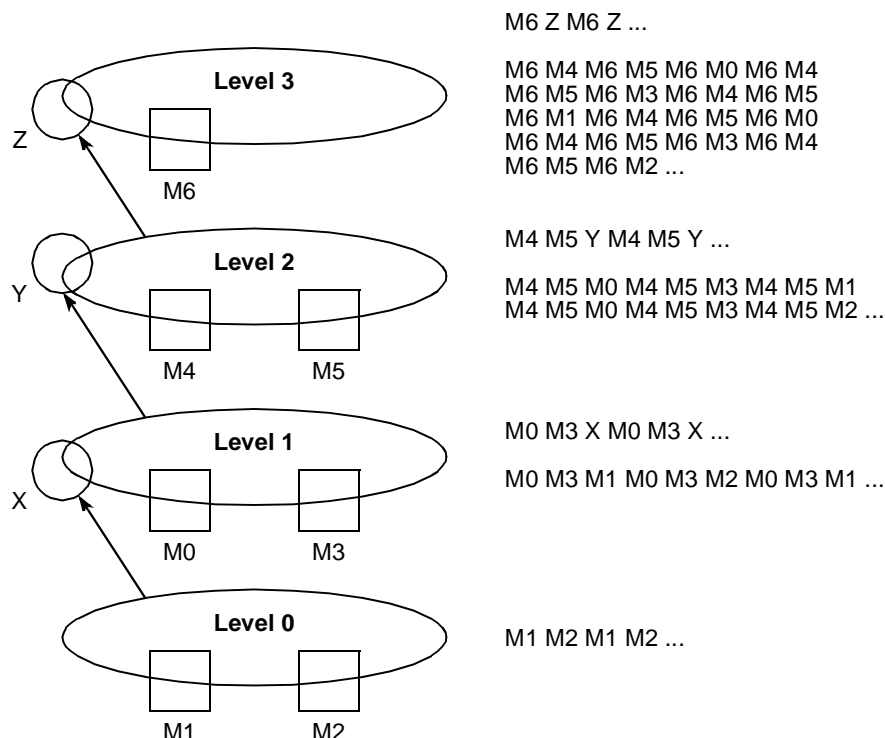


Figure 6-10. An Example of Priority-Based Arbitration Algorithm

NOTE

See each bus master’s chapter and [Section 5.4.3.4, “System Priority and Configuration Register \(SPCR\),”](#) for more details about priority programming.

6.3.1.2 Address Bus Arbitration with REPEAT

When a master owns the address bus and wants to perform another transaction, it can assert bus request along with REPEAT, to make a repeat request to the arbiter. Consequently, the arbiter asserts bus grant to the same master if the current address tenure is not being ARTRYed. This happens regardless of the priority level of bus request from other masters. In another word, “repeat request” overrides the priority scheme.

Even though repeat request can improve the page hit ratio and the overall memory bandwidth efficiency, it can increase the worst case latency of individual master. Therefore, the arbiter has programmable

counter to limit the maximum number of consecutive transactions that are performed by masters. Whenever the counter expires, arbiter ignores the $\overline{\text{REPEAT}}$ signal and falls back to the regular arbitration scheme. PCI master has a dedicated repeat counter as it might need more repeated transactions before accepting read requests. PCI ordering rules require that the PCI bridge should empty all queued write operations before any new read operation can begin. See Section 3.2.5, “Transaction Ordering and Posting,” of the *PCI Local Bus Specifications Rev 2.2*, for more information.

See [Section 6.2.1, “Arbiter Configuration Register \(ACR\),”](#) for more details about programming ACR[RPTCNT] and ACR[PCI_RPTCNT].

6.3.1.3 Address Bus Arbitration After $\overline{\text{ARTRY}}$

The $\overline{\text{ARTRY}}$ protocol is used primarily by the CPU to interrupt a transaction that hits to a modified line in its D-cache, so that it can maintain data coherency by performing the snoop copyback. When CPU asserts ARTRY, the bus is immediately granted to the CPU to perform snoop copyback. After the completion of snoop copyback, the arbiter grants the bus to the most ahead master among those masters which have an active bus request signal at that time, which may or may not be the same master that had its transaction ARTRYed. Only when a transaction address phase is completed with no ARTRY (and no repeat conditions), the master moves to the end of the line.

6.3.1.4 Address Bus Parking

The arbiter supports address bus parking. This feature implies that when no master is requesting the bus (all bus requests are negated), the arbiter can choose to park the address bus (or assert the address bus grant) to a master. The parked master can skip the bus request and assume the bus mastership directly. This reduces the access latency for parked master.

See [Section 6.2.1, “Arbiter Configuration Register \(ACR\),”](#) for more details about ACR[APARK] and ACR[PARKM].

6.3.1.5 Data Bus Arbitration

For every committed address tenure a data tenure is required to complete the transaction.

In the device system, the arbiter controls the issuing of data bus grants to a master and a slave, which are involved in a data tenure of a previously performed address tenure.

6.3.2 Bus Error Detection

The arbiter is responsible for tracking the following cases on the bus:

- Address time out
- Data time out
- Transfer error
- Address only transaction type
- Reserved transaction type
- Illegal (**eciwx/ecowx**) transaction type

6.3.2.1 Address Time Out

Address time out occurs, if the address tenure was not ended before the specified time-out period (programmed by ATR[ATO]). In this case, the arbiter performs as follows:

1. Ends the address tenure.
2. Starts data tenure and ends it by asserting transfer error.
3. Reports on the event to AER[ATO].
4. Issues reset request, MCP or regular interrupt according to AERR[ATO] and AIDR[ATO], if enabled by AMR[ATO].
5. Updates transaction attributes and address of AEATR and AEADR for the first error event.

6.3.2.2 Data Time Out

Data time out occurs, if the data tenure was not ended before the specified time-out period (programmed by ATR[DTO]). In this case, the arbiter performs as follows:

1. Ends the data tenure by asserting transfer error.
2. Reports on this event in AER[DTO].
3. Issues reset request, MCP or regular interrupt according to AERR[DTO] and AIDR[DTO], if enabled by AMR[DTO].
4. Updates transaction attributes and address of AEATR and AEADR for the first error event.

6.3.2.3 Transfer Error

The arbiter tracks the transfer error asserted by one of the slaves. In this case, the arbiter performs as follows:

1. Reports on the event to AER[ETEA].
2. Issues reset request, MCP or regular interrupt according to AERR[ETEA] and AIDR[ETEA] if enabled by AMR[ETEA].
3. Updates transaction attributes and address of AEATR and AEADR for the first error event.

6.3.2.4 Address Only Transaction Type

Table 6-10 shows transaction types, which are defined as address only:

Table 6-10. Address Only Transaction Type Encoding

ttype[0:4]	Bus Commands
00000	Clean block
00100	Flush block
01000	Sync
01100	Kill block
10000	eieio
11000	TLB Invalidate
00001	Iwarx reservation set
01001	tlbsync

Table 6-10. Address Only Transaction Type Encoding (continued)

ttype[0:4]	Bus Commands
01101	icbi

The arbiter allows address-only (AO) transactions on the bus and the G2 core has ability to issue address-only (AO) transactions (see HID0 [ABE] in the *G2 PowerPC Core Reference Manual, Rev 1*). As there is no advantage in using AO transaction in this system, the bus monitor allows the detection of AO transactions and treats them as an error.

The transaction with an address only transfer type, the arbiter performs as follows:

1. Ends the address tenure by asserting $\overline{\text{AACK}}$.
2. Reports on the event to AER[AO].
3. Issues reset request, MCP or regular interrupt according to AERR[AO] and AIDR[AO] if enabled by AMR[AO].
4. Updates transaction attributes and address of AEATR and AEADR for the first error event.

6.3.2.5 Reserved Transaction Type

Table 6-11 shows transaction types defined as reserved.

Table 6-11. Reserved Transaction Type Encoding

ttype[0:4]	Bus Commands
00101	Reserved
1xx01	Reserved for customer
10110	Reserved
00011	Reserved
00111	Reserved
01111	Reserved
1xx11	Reserved for customer

The transaction with a reserved transfer type, the arbiter performs as follows:

1. Ends the address tenure by asserting $\overline{\text{AACK}}$.
2. Reports on the event to AER[RES].
3. Issues reset request, MCP or regular interrupt according to AERR[RES] and AIDR[RES], if enabled by AMR[RES].
4. Updates transaction attributes and address of AEATR and AEADR for the first error event.

6.3.2.6 Illegal (eciwx/ecowx) Transaction Type

Table 6-12 shows transaction types defined as illegal.

Table 6-12. Illegal Transaction Type Encoding

ttype[0:4]	Bus command
10100	External control word write (ecowx)
11100	External control word read (eciwx)

The transaction with an illegal (eciwx, ecowx) transfer type, the arbiter performs as follows:

1. Ends the address tenure by asserting $\overline{\text{AACK}}$.
2. Starts data tenure and ends data tenure by asserting $\overline{\text{TEA}}$.
3. Reports on the event in AER[ECW].
4. Issues reset request, MCP or regular interrupt according to AERR[ECW] and AIDR[ECW], if enabled by AMR[ECW].
5. Updates transaction attributes and address of AEATR and AEADR for the first error event.

See [Section 6.2.3, “Arbiter Event Register \(AER\),”](#) [Section 6.2.4, “Arbiter Interrupt Definition Register \(AIDR\),”](#) [Section 6.2.5, “Arbiter Mask Register \(AMR\),”](#) [Section 6.2.6, “Arbiter Event Attributes Register \(AEATR\),”](#) [Section 6.2.7, “Arbiter Event Address Register \(AEADR\),”](#) and [Section 6.2.8, “Arbiter Event Response Register \(AERR\),”](#) for more information.

6.4 Initialization/Applications Information

The following sections describe the initialization and error handling sequences for the arbiter.

6.4.1 Initialization Sequence

The following initialization sequence is recommended:

1. Write to ACR to configure pipeline depth, address bus parking mode, global maximum repeat count, PCI maximum repeat count.
2. Write to AERR defines whether different error events cause a reset request or an interrupt.
3. Write to AIDR defines the kind of interrupt (regular or MCP) caused by each error event. Note that this is necessary only if interrupts are enabled and AERR defines error events to cause interrupt.
4. Write to AMR to enable interrupts.
5. Write to ATR to set the ATO and DTO timers. Note that this is only necessary if the required timers are less than the maximum value (which is default).

6.4.2 Error Handling Sequence

The following error handling sequence is recommended:

1. Read to AER to find out about the error that occurred in the system. Also, read the values of AEATR and AEADR to check on the first error event in the system.
2. If those registers are not accessible because of a bus deadlock situation, reset the chip and read the values of the AEATR and AEADR registers to check on the event that causes this problem to the system. Use $\overline{\text{HRESET}}$ to reset the chip to guarantee that the information stored in AEATR and AEADR is not lost.
3. Clear all the previous events by writing ones to the AER. This register is also cleared after reset.

Chapter 7

e300 Processor Core Overview

This chapter provides an overview of features for the embedded microprocessors in the e300 core family, which are PowerPC microprocessors built on Power Architecture technology. Throughout this chapter, the terms ‘e300 core’, ‘core’, and ‘processor’ are used interchangeably. The term ‘e300c1’ is used when describing an implementation-specific feature or when a difference exists between different configurations. The term ‘e300’ is used when describing a feature that pertains to the family of e300 processors. The MPC8360 uses an e300c1 core.

7.1 e300c1 Overview

This section describes the details of the e300 core, provides a block diagram showing the major functional units, and briefly describes how these units interact. All differences between the e300 and previous PowerPC implementations derived from the MPC603e processor are noted. For additional information, please refer to the *e300 PowerPC Core Family Reference Manual*.

The e300 core is a low-power implementation of this microprocessor family of reduced instruction set computing (RISC) microprocessors. The core implements the 32-bit portion of the PowerPC architecture, which defines 32-bit effective addresses, integer data types of 8, 16, and 32 bits, and floating-point data types of 32 and 64 bits.

The core is a superscalar processor that can issue and retire as many as three instructions per clock cycle. Instructions can execute out of program order for increased performance; however, the core makes completion appear sequential.

The e300 core integrates independent execution units including: an integer unit (IU) a floating-point unit (FPU), a branch processing unit (BPU), a load/store unit (LSU), and a system register unit (SRU). The ability to execute instructions in parallel and the use of simple instructions with rapid execution times yield high efficiency and throughput for e300-core-based systems. Most integer instructions execute in one clock cycle. In the e300c1 core, the FPU is pipelined so a single-precision multiply-add instruction can be issued and completed every clock cycle. The e300c1core provide hardware support for all single- and double-precision floating-point operations for most value representations and all rounding modes.

Figure 7-1 shows a block diagram of the e300c1 core.

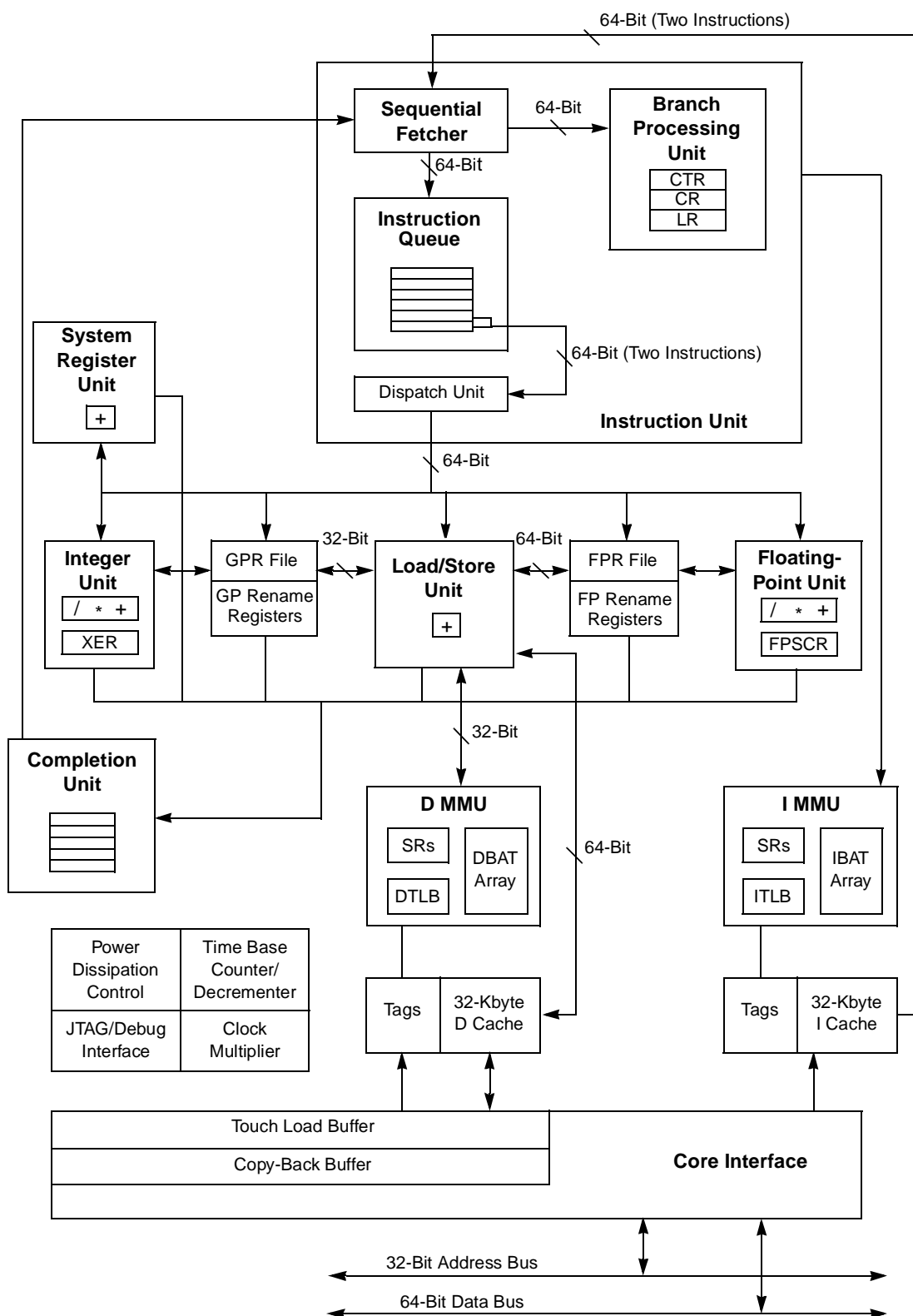


Figure 7-1. e300c1 Core Block Diagram

The e300c1 provide independent, on-chip, 32-Kbyte, eight-way, set-associative, physically-addressed caches for instructions and data, and on-chip instruction and data memory management units (MMUs). The MMUs contain 64-entry, two-way, set-associative, data and instruction translation lookaside buffers (DTLB and ITLB) that provide support for demand-paged, virtual-memory, address translation, and variable-sized block translation. The TLBs use a least recently used (LRU) replacement algorithm and the caches use a pseudo least recently used algorithm (PLRU).

The core also supports block address translation through the use of two independent instruction and data block address translation (IBAT and DBAT) arrays, each containing eight pairs of BATs, an increase from four pairs of each type of BATs in the G2 core. This increase provides more flexibility in protecting accesses and providing translation on a segment, block, or page basis for memory accesses and I/O accesses. Effective addresses are compared simultaneously with all eight entries in the BAT array during block translation. In accordance with the PowerPC architecture, if an effective address hits in both the TLB and BAT array, the BAT translation takes priority.

As part of the coherent system bus (CSB), the e300 core has a 64-bit data bus and a 32-bit address bus. During normal operation, the e300 core provides a three-state (modified, exclusive, and invalid) coherency protocol which is a compatible subset of a four-state (modified/exclusive/shared/invalid) MESI protocol. However, the e300 data cache contains a programmable MESI extension that supports the shared cache coherency state (similar to other PowerPC processors). Both protocols operate coherently in systems that contain four-state caches. Although MESI is supported by the e300 core, it is not implemented on the MPC8360. The core also supports single-beat and burst data transfers for memory accesses and supports memory-mapped I/O operations.

The true little-endian mode is another enhanced capability of the e300 core. Unlike the PowerPC little-endian mode (which manipulates only the address bits), no longer supported on the e300, the true little-endian mode actually operates on true little-endian instructions and data from memory.

The critical interrupt is an additional interrupt in the e300 core and has higher priority order than the system management interrupt. Also, debug features are improved in the e300. Additional SPRG interrupt handling registers are provided for enhancing flexibility for the operating system.

7.1.1 e300c1 Features

This section describes the major features of the e300 core:

- High-performance, superscalar microprocessor core
 - As many as three instructions issued and retired per clock (two instructions plus one branch instruction)
 - As many as five instructions in execution per clock
 - Single-cycle execution for most instructions
 - Pipelined floating-point unit (FPU) for all single- and double-precision operations
- Independent execution units and two register files
 - Branch processing unit (BPU) featuring static branch prediction
 - One 32-bit integer unit (IU) in the e300c1
 - FPU based on the IEEE Std 754™ for both single- and double-precision operations

- Load/store unit (LSU) for data transfer between data-cache and general-purpose registers (GPRs) and floating-point registers (FPRs)
- System register unit (SRU) that executes condition register (CR), special-purpose register (SPR), and integer add/compare instructions. Add/compare instructions are also executed in the IUs.
- Thirty-two 32-bit GPRs for integer operands
- Thirty-two 64-bit FPRs for single- or double-precision operands
- High instruction and data throughput
 - Zero-cycle branch capability (branch folding)
 - Programmable static branch prediction on unresolved conditional branches
 - Instruction fetch unit capable of fetching two instructions per clock from the instruction cache
 - A six-entry instruction queue (IQ) that provides lookahead capability
 - Independent pipelines with feed-forwarding that reduces data dependencies in hardware
 - 32-Kbyte data cache and 32-Kbyte instruction cache with parity—eight-way, set-associative, physically addressed, PLRU replacement algorithm on the e300c1.
 - Cache write-back or write-through operation programmable on a per-page or per-block basis
 - Features for instruction and data cache locking and protection
 - BPU that performs CR lookahead operations
 - Address translation facilities for 4-Kbyte page size, variable block size, and 256-Mbyte segment size
 - A 64-entry, two-way, set-associative ITLB and DTLB
 - Eight-entry data and instruction BAT arrays providing 128-Kbyte to 256-Mbyte blocks
 - Software table search operations and updates supported through fast trap mechanism
 - 52-bit virtual address; 32-bit physical address
- Facilities for enhanced system performance
 - A 64-bit split-transaction internal data bus interface to the coherent system bus (CSB) with burst transfers
 - Support for one-level address pipelining on the CSB interface
 - True little-endian mode for compatibility with other true little-endian devices
 - Critical interrupt support
 - Hardware support for misaligned little-endian accesses
 - Configurable processor bus frequency multipliers as defined in the *MPC8360E Integrated Processor Hardware Specifications*
- Integrated power management
 - Internal processor/bus clock multiplier ratios
 - Three power-saving modes: doze, nap, and sleep
 - Automatic dynamic power reduction when internal functional units are idle
- In-system testability and debugging features through JTAG boundary-scan capability

Features specific to the e300 core not present on the G2 processors follow:

- Enhancements to the register set
 - The e300 core has one more HID0 bit than the G2:
 - The enable cache parity checking (ECPE) bit, HID0[1], gives the e300 core the ability to enable the taking of a machine check interrupt based on the detection of a cache parity error
- Enhancements to cache implementation
 - 32-Kbyte data cache and 32-Kbyte instruction cache with parity—eight-way, set-associative, physically addressed, PLRU replacement algorithm on the e300c1.
 - Full parity checking is performed on both instruction and data cache memory arrays
 - Lockable L1 instruction and data caches—entire cache or on a per-way basis up to 7 of 8 ways on the e300c1
 - New **icbt** instruction supports initialization of instruction cache
 - Data cache supports four-state MESI coherency protocol (not implemented on MPC8360E)
 - The instruction cache is blocked only until the critical load completes (hit under reloads allowed)
 - Instruction cancel mechanism improves utilization of instruction cache by supporting hits-under-cancels and misses-under-cancels.
 - The critical double word is simultaneously written to the cache and forwarded to the requesting unit, thus minimizing stalls due to load delays.
 - Data cache queue sharing makes cast-outs and snoop pushes more efficient
 - Provides for an optional data cache operation broadcast feature (enabled by HID0[ABE]) that allows for coherent system management. All of the data cache control instructions, except **dcbz** (**dcbi**, **dcbf**, and **dcbst**) require that HID0[ABE] be enabled to broadcast.
 - Instruction fetch burst feature allows all instruction fetches from caching-inhibited space to be performed on the bus as burst transactions
- Interrupts
 - The e300 core offers hardware support for misaligned little-endian accesses. Little-endian load/store accesses that are not on a word boundary, except for strings and multiples, generate interrupts under the same circumstances as big-endian accesses.
 - The e300 core supports true little-endian mode to minimize the impact on software porting from true little-endian systems.
 - An input interrupt signal, \overline{cint} , is provided to trigger the critical interrupt exception on the e300 core.
- Bus clock—PLL configuration signals include seven signals for settings and control: *pll_cfg[0:6]*.
- Debug features
 - Breakpoint status recorded in DBCR and IBCR control registers
 - Two signals for the debug interface: $\overline{stopped}$ and $\overline{ext_halt}$

Figure 7-1 provides a block diagram of the e300 core that shows how the execution units—IU, FPU, BPU, LSU, and SRU—operate independently and in parallel. It should be noted that this is a conceptual diagram and does not attempt to show how these features are physically implemented on the device.

The e300 core provides address translation and protection facilities, including an ITLB, DTLB, and instruction and data BAT arrays. Instruction fetching and issuing are handled in the instruction unit. Translation of addresses for cache or external memory accesses are handled by the MMUs. Both units are discussed in more detail in [Section 7.1.2, “Instruction Unit,”](#) and [Section 7.1.5.1, “Memory Management Units \(MMUs\).”](#)

7.1.2 Instruction Unit

As shown in [Figure 7-1](#), the e300 core instruction unit, containing a fetch unit, instruction queue, dispatch unit, and BPU, provides centralized control of instruction flow to the execution units. The instruction unit determines the address of the next instruction to be fetched based on information from the sequential fetcher and from the BPU.

The instruction unit fetches the instructions from the instruction cache into the instruction queue. The BPU receives branch instructions from the fetcher and uses static branch prediction to allow fetching from a predicted instruction stream while a conditional branch is evaluated. The BPU folds out for unconditional branch instructions and conditional branch instructions unaffected by instructions in the execution pipeline.

Instructions issued beyond a predicted branch cannot complete execution until the branch is resolved, preserving the programming model of sequential execution. If any of these are branch instructions, they are decoded but not issued. Instructions to be executed by the FPU, IU, LSU, and SRU are issued and allowed to progress up to the register write-back stage. Write-back is allowed when a correctly predicted branch is resolved, and execution continues along the predicted path.

If branch prediction is incorrect, the instruction unit flushes all predicted path instructions, and instructions are issued from the correct path.

7.1.2.1 Instruction Queue and Dispatch Unit

The instruction queue (IQ), shown in [Figure 7-1](#), holds as many as six instructions and loads up to two instructions from the instruction unit during a single cycle. The instruction fetch unit continuously loads as many instructions as space in the IQ allows. Instructions are dispatched to their respective execution units from the dispatch unit at a maximum rate of two instructions per cycle. Dispatching is facilitated to the IUs, FPU, LSU, and SRU by the provision of a reservation station at each unit. The dispatch unit performs source and destination register dependency checking, determines dispatch serializations, and inhibits subsequent instruction dispatching as required.

For a more detailed overview of instruction dispatch, see [Section 7.3.6, “Instruction Timing.”](#)

7.1.2.2 Branch Processing Unit (BPU)

The BPU receives branch instructions from the fetch unit and performs CR lookahead operations on conditional branches to resolve them early, achieving the effect of a zero-cycle branch in many cases.

The BPU uses a bit in the instruction encoding to predict the direction of the conditional branch. Therefore, when an unresolved conditional branch instruction is encountered, the core fetches instructions from the predicted target stream until the conditional branch is resolved.

The BPU contains an adder to compute branch target addresses and three user-control registers: the link register (LR), the count register (CTR), and the conditional register (CR). The BPU calculates the return pointer for sub-routine calls and saves it into the LR for certain types of branch instructions. The LR also contains the branch target address for the Branch Conditional to Link Register (**bclrx**) instruction. The CTR contains the branch target address for the Branch Conditional to Count Register (**bctrx**) instruction. The contents of the LR and CTR can be copied to or from any GPR. Because the BPU uses dedicated registers rather than GPRs or FPRs, execution of branch instructions is largely independent from execution of integer and floating-point instructions.

7.1.3 Independent Execution Units

The PowerPC architecture's support for independent execution units allows implementation of processors with out-of-order instruction execution. For example, because branch instructions do not depend on GPRs or FPRs, branches can often be resolved early, eliminating stalls caused by taken branches.

The four other execution units and the completion unit are described in the following sections.

7.1.3.1 Integer Unit (IU)

The IU executes all integer instructions. The IU executes one integer instruction at a time, performing computations with its arithmetic logic unit (ALU), multiplier, divider, and XER register. Most integer instructions are single-cycle instructions. The 32 GPRs hold integer operands. Stalls due to contention for GPRs are minimized by the automatic allocation of rename registers. The core writes the contents of the rename registers to the appropriate GPR when integer instructions are retired by the completion unit.

7.1.3.2 Floating-Point Unit (FPU)

The FPU contains a single-precision multiply-add array and the floating-point status and control register (FPSCR). The multiply-add array allows the core to efficiently implement multiply and multiply-add operations. The FPU is pipelined so that single- and double-precision instructions can be issued back-to-back. The 32 FPRs are provided to support floating-point operations. Stalls due to contention for FPRs are minimized by the automatic allocation of rename registers. The core writes the contents of the rename registers to the appropriate FPR when floating-point instructions are retired by the completion unit.

The e300c1 core supports all floating-point data types based on the IEEE 754 standard (normalized, denormalized, NaN, zero, and infinity) in hardware, eliminating the latency incurred by software interrupt routines.

7.1.3.3 Load/Store Unit (LSU)

The LSU executes all load and store instructions and provides the data transfer interface between the GPRs, FPRs, and the cache/memory subsystem. The LSU calculates effective addresses, performs data alignment, and provides sequencing for load/store string and multiple instructions.

Load and store instructions are issued and executed in program order; however, the memory accesses can occur out of order. Synchronizing instructions are provided to enforce strict ordering.

Cacheable loads, when free of data bus dependencies, can execute out of order with a maximum throughput of one per cycle and with a two-cycle total latency. Data returned from the cache is held in a rename register until the completion logic commits the value to a GPR or FPR. Stores cannot be executed in a predicted manner and are held in the store queue until the completion logic signals that the store operation is to be completed to memory. The core executes store instructions with a maximum throughput of one per cycle and with a three-cycle total latency. The time required to perform the actual load or store depends on whether the operation involves the cache, system memory, or an I/O device.

7.1.3.4 System Register Unit (SRU)

The SRU executes various system-level instructions, including condition register logical operations and move to/from special-purpose register instructions. It also executes integer add/compare instructions. In order to maintain system state, most instructions executed by the SRU are completion-serialized; that is, the instruction is held for execution in the SRU until all prior instructions issued have completed. Results from completion-serialized instructions executed by the SRU are not available or forwarded for subsequent instructions until they complete.

7.1.4 Completion Unit

The completion unit tracks instructions in program order from dispatch through execution and then completes. Completing an instruction commits the core to any architectural register changes caused by that instruction. In-order completion ensures the correct architectural state when the core must recover from a mispredicted branch or an interrupt.

Instruction state and other information required for completion is kept in a five-entry FIFO completion queue. A single completion queue entry is allocated for each instruction once it enters the execution unit from the dispatch unit. An available completion queue entry is a required resource for dispatch; if no completion entry is available, dispatch stalls. A maximum of two instructions per cycle are completed in order from the queue.

7.1.5 Memory Subsystem Support

The core provides separate instruction and data caches and MMUs. The core also provides an efficient processor bus interface to facilitate access to main memory and other bus subsystems. The memory subsystem support functions are described in the following sections.

7.1.5.1 Memory Management Units (MMUs)

The core MMUs support up to 4 Petabytes (2^{52}) of virtual memory and 4 Gigabytes (2^{32}) of physical memory (referred to as real memory in the architecture specification) for instruction and data. The MMUs also control access privileges for these spaces on block and page granularities. Referenced and changed status is maintained by the processor for each page to assist implementation of a demand-paged virtual memory system. Note that software assistance is required for the device to maintain reference and changed status. A key bit is implemented to provide information about memory protection violations prior to page table search operations.

The LSU calculates effective addresses for data loads and stores, performs data alignment to and from cache memory, and provides the sequencing for load and store string and multiple word instructions. The instruction unit calculates effective addresses for instruction fetching.

After an EA is generated, its higher-order bits are translated by the appropriate MMU into physical address bits. The lower-order EA bits are the same on the physical address which are directed to the on-chip cache and formed the index into a four-way set-associative tag array. After translating the address, the MMU passes the higher-order physical address bits to the cache and the cache lookup completes. For caching-inhibited accesses or accesses that miss in the cache, the untranslated lower-order address bits are concatenated with the translated higher-order address bits; the resulting 32-bit physical address is then used by the memory unit and the core interface to access external memory.

The MMU also directs the address translation and enforces the protection hierarchy programmed by the operating system in relation to the supervisor/user privilege level of the access and in relation to whether the access is a load or store.

For instruction fetches, the IMMU looks for the address in the ITLB and in the IBAT array. If an address hits both, the IBAT array translation is used. Data accesses cause a lookup in the DTLB and DBAT array. In most cases, the translation is in a TLB and the physical address bits are available to the on-chip cache.

The e300 core implements four more IBAT and four more DBAT entries than the G2.

When the EA misses in the TLBs, the core provides hardware assistance for software to perform a search of the translation tables in memory. The hardware assist consists of the following features:

- Automatic storage of the missed effective address in IMISS and DMISS
- Automatic generation of the primary and secondary hashed real addresses of the page-table entry group (PTEG), which are readable from the HASH1 and HASH2 register locations.
The HASH data is generated from the contents of the IMISS or DMISS register. The register that is selected depends on the miss (instruction or data) that was last acknowledged.
- Automatic generation of the first word of the page table entry (PTE) of the tables being searched
- A real page address (RPA) register that matches the format of the lower word of the PTE
- TLB access instructions (**tlbli** and **tlbld**) that are used to load an address translation into the instruction or data TLBs
- Shadow registers for GPR0–GPR3 that allow miss code to execute without corrupting the state of any of the existing GPRs. Shadow registers are used only for servicing a TLB miss.

See [Section 7.3.5.2, “Implementation-Specific Memory Management,”](#) for more information about memory management for the core.

7.1.5.2 Cache Units

The e300c1 provides independent, 32-Kbyte, eight-way, set-associative, instruction and data caches. The cache block is 32 bytes long. The caches adhere to a write-back policy, but the e300 core allows control of cacheability, write policy, and memory coherency at the page and block levels. The caches use a pseudo LRU replacement policy.

As shown in [Figure 7-1](#), the caches provide a 64-bit interface to the instruction fetch unit and LSU. The surrounding logic selects, organizes, and forwards the requested information to the requesting unit. Write operations to the cache can be performed on a byte basis, and a complete read-modify-write operation to the cache can occur in each cycle.

The load/store and instruction fetch units provide the caches with the address of the data or instruction to be fetched. In the case of a cache hit, the cache returns two words to the requesting unit.

Because the data cache tags are single-ported, simultaneous load/store and snoop accesses cause resource contention. Snoop accesses have the highest priority and are given first access to the tags, unless the snoop access coincides with a tag write; in this case the snoop is retried and must re-arbitrate for cache access. Loads or stores deferred due to snoop accesses are performed on the clock cycle following the snoop.

The e300 core includes a new instruction cancel extension. The instruction cancel extension improves utilization of the instruction cache during cancel operations. It allows a new instruction fetch to be issued to the cache or to the bus if a canceled instruction fetch is pending or active on the bus. This supports hit-under-cancel and miss-under-cancel instruction fetch operations.

7.1.6 Bus Interface Unit (BIU)

Because the caches are on-chip, write-back caches, the most common transactions are burst-read memory operations, burst-write memory operations, and single-beat (noncacheable or write-through) memory read and write operations. There can also be address-only operations, variants of the burst and single-beat operations, (for example, global memory operations that are snooped and atomic memory operations), and address retry activity (for example, when a snooped read access hits a modified cache block).

Memory accesses can occur in single-beat (1–8 bytes) and four-beat burst (32 bytes) data transfers on the 64-bit data bus. The address and data buses operate independently to support pipelining and split transactions during memory accesses.

The e300 bus interface unit (BIU) has been enhanced to allow a pipeline slot to become available once a previous transaction has been granted the data bus (that is, as early as when the data tenure starts rather than after the data tenure completes), thus allowing for greater bus utilization in systems that support it. This is sometimes referred to as 1 1/2-level pipelining.

Typically, memory accesses are weakly ordered, meaning that sequences of operations, including load/store string and multiple instructions, do not necessarily complete in the order they begin. This weak ordering maximizes the efficiency of the bus without sacrificing coherency of the data. The core allows read operations to precede store operations (except when a dependency exists, or in cases where a noncacheable access is performed), and provides support for a write operation to proceed a previously queued read data tenure (for example, allowing a snoop push to be enveloped by the address and data tenures of a read operation). Because the processor can dynamically optimize run-time ordering of load/store traffic, overall performance is improved.

7.1.7 System Support Functions

The e300 core implements several support functions that include power management, time base/decrementer registers for system timing tasks, a JTAG (based on IEEE Std 1149.1™) interface, hardware debug, and a phase-locked loop (PLL) clock multiplier. These system support functions are described in the following sections.

7.1.7.1 Power Management

The e300 core provides four power modes, selectable by setting the appropriate control bits in the machine state register (MSR) and the hardware implementation register 0 (HID0). When entering into a power mode other than full-power, the core will request entry via a *qreq* signal and will only enter another power mode after an acknowledge (*qack*) is received. The four power modes are as follows:

- Full-power—This is the default power state of the e300 core. The e300 core is fully powered and the internal functional units are operating at the full processor clock speed. If the dynamic power management mode is enabled, functional units that are idle will automatically enter a low-power state without affecting performance, software execution, or external hardware.
- Doze—All the functional units of the e300 core are disabled except for the time base/decrementer registers and the bus snooping logic. When the processor is in doze mode, an external asynchronous interrupt, system management interrupt, decremter interrupt, hard or soft reset, or machine check brings the e300 core into the full-power state. The core in doze mode maintains the PLL in a fully-powered state and locked to the system external clock input (*sysclk*), so a transition to the full-power state takes only a few processor clock cycles.
- Nap—The nap mode further reduces power consumption by disabling bus snooping, leaving only the time base register and the PLL in a powered state. The core returns to the full-power state on receipt of an external asynchronous interrupt, system management interrupt, decremter interrupt, hard or soft reset, or machine check input (*mcp*) signal. A return to full-power state from a nap state takes only a few processor clock cycles.
- Sleep—Sleep mode reduces power consumption to a minimum by disabling all internal functional units; then external system logic may disable the PLL and *sysclk*. Returning the core to the full-power state requires the enabling of the PLL and *sysclk*, followed by the assertion of an external asynchronous interrupt, system management interrupt, hard or soft reset, or *mcp* signal after the time required to relock the PLL.

7.1.7.2 Time Base/Decrementer

The time base is a 64-bit register (accessed as two 32-bit registers) that is incremented once every four bus clock cycles; external control of the time base is provided through the time base/decrementer clock base enable (*tben*) signal. The decremter is a 32-bit register that generates a decremter interrupt after a programmable delay. The contents of the decremter register are decremented once every four bus clock cycles, and the decremter interrupt is generated as the count passes through zero.

7.1.7.3 JTAG Test and Debug Interface

The core provides JTAG and hardware debug functions for facilitating board testing and chip debugging. The JTAG test interface (based on IEEE 1149.1) provides a means for boundary-scan testing of the core and the attached system logic. The hardware debug function accesses the JTAG test port, providing a means for executing test routines and facilitating chip and software debugging.

All instruction and data address breakpoints are accessible in the IBCR and DBCR. See [Section 7.3.8, “Debug Features,”](#) for more information.

7.1.7.4 Clock Multiplier

The internal clocking of the e300 core is generated from and synchronized to the external clock signal, *sysclk*, by means of a voltage-controlled, oscillator-based PLL. The PLL provides programmable internal processor clock multiplier ratios which multiply the externally supplied clock frequency. The bus clock is the same frequency and is synchronous with *sysclk*. The configuration of the PLL can be read by software from the hardware implementation register 1 (HID1).

7.2 PowerPC Architecture Implementation

The PowerPC architecture consists of the following layers, and adherence to the PowerPC architecture can be measured in terms of which of the following levels of the architecture is implemented:

- User instruction set architecture (UISA)—Defines the base user-level instruction set, user-level registers, data types, floating-point interrupt model, memory models for a uniprocessor environment, and programming model for a uniprocessor environment.
- Virtual environment architecture (VEA)—Describes the memory model for a multiprocessor environment, defines cache control instructions, and describes other aspects of virtual environments. Implementations that conform to the VEA also adhere to the UISA but may not necessarily adhere to the OEA.
- Operating environment architecture (OEA)—Defines the memory management model, supervisor-level registers, synchronization requirements, and interrupt model. Implementations that conform to the OEA also adhere to the UISA and VEA.

The PowerPC architecture allows a wide range of designs for such features as cache and core interface implementations.

7.3 Implementation-Specific Information

This section describes the PowerPC architecture in general and specific details about the implementation of the e300 core as a low-power, 32-bit member of this PowerPC core family. The main topics addressed are as follows:

- [Section 7.3.1, “Register Model,”](#) describes the registers for the operating environment architecture common among e300 cores that implement the PowerPC architecture and describes the programming model. It also describes the additional registers that are unique to the core.
- [Section 7.3.2, “Instruction Set and Addressing Modes,”](#) describes the PowerPC instruction set and addressing modes for the OEA, and defines and describes the instructions implemented in the core.

- [Section 7.3.3, “Cache Implementation,”](#) describes the cache model that is defined generally for cores that implement the PowerPC architecture by the VEA. It also provides specific details about the e300 core cache implementation.
- [Section 7.3.4, “Interrupt Model,”](#) describes the interrupt model of the OEA and the differences in the core interrupt model.
- [Section 7.3.5, “Memory Management,”](#) describes generally the conventions for memory management among these cores. This section also describes the core implementation of the 32-bit PowerPC memory management specification.
- [Section 7.3.6, “Instruction Timing,”](#) provides a general description of the instruction timing provided by the superscalar, parallel execution supported by the PowerPC architecture and the e300 core.
- [Section 7.1.6, “Bus Interface Unit \(BIU\),”](#) describes the signals implemented on the core.

The e300 core is a high-performance, superscalar processor core. The PowerPC architecture allows optimizing compilers to schedule instructions to maximize performance through efficient use of the PowerPC instruction set and register model. The multiple, independent execution units allow compilers to optimize instruction throughput. Compilers that take advantage of the flexibility of the PowerPC architecture can additionally optimize system performance.

The following sections summarize the features of the core, including both those that are defined by the architecture and those that are unique to the various core implementations.

Specific features of the core are listed in [Section 7.1.1, “e300c1 Features.”](#)

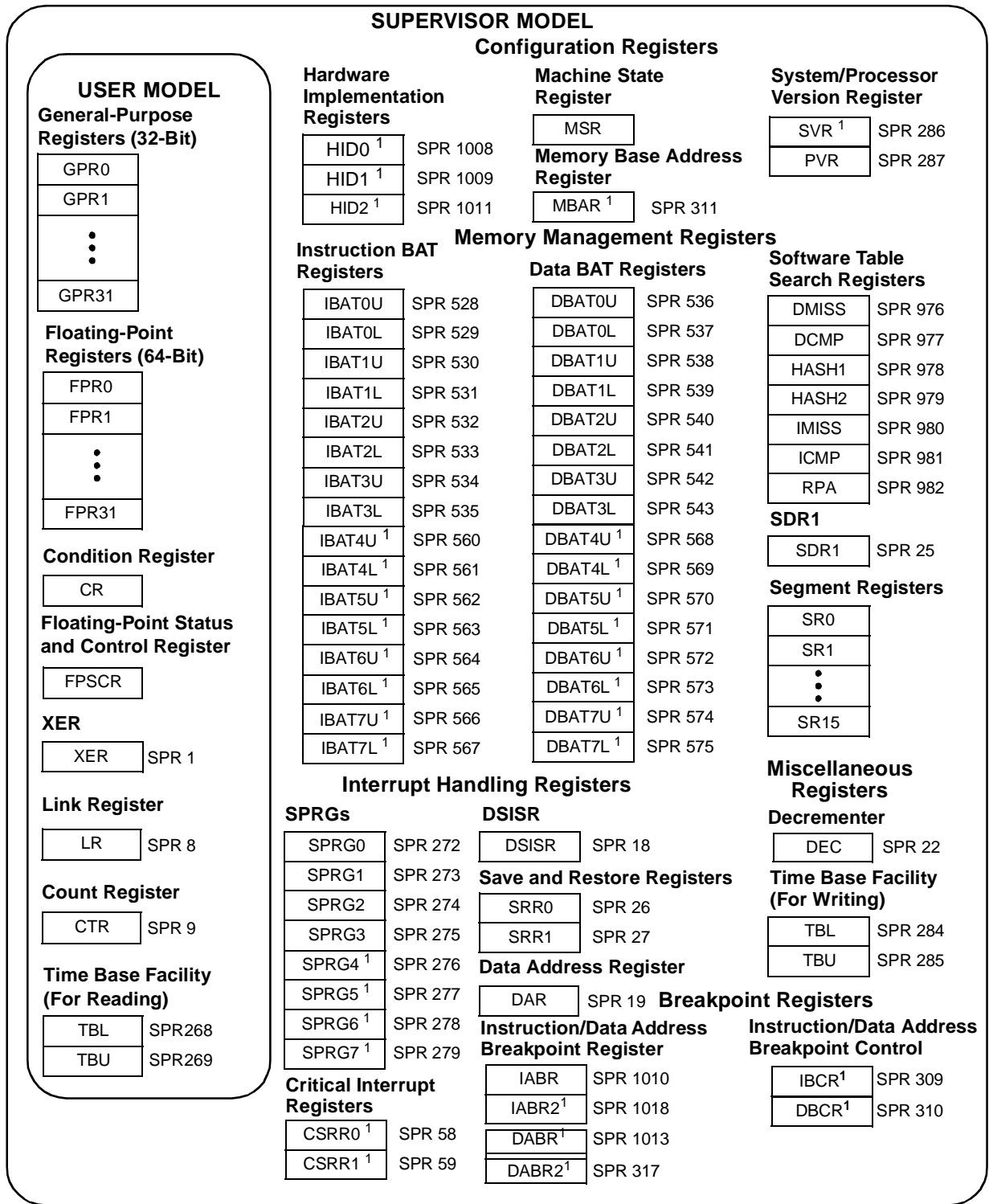
7.3.1 Register Model

The PowerPC architecture defines register-to-register operations for most computational instructions. Source operands for these instructions are accessed from the registers or are provided as immediate values embedded in the instruction opcode. The three-register instruction format allows specification of a target register distinct from the two-source operands. Load and store instructions transfer data between registers and memory.

The e300 core has two levels of privilege: supervisor mode of operation (typically used by the operating system) and user mode of operation (used by the application software). The programming models incorporate 32 GPRs, 32 FPRs, special-purpose registers (SPRs), and several miscellaneous registers. Each core also has its own unique set of hardware implementation (HID) registers.

Having access to privileged instructions, registers, and other resources allows the operating system to control the application environment (providing virtual memory and protecting operating system and critical machine resources). Instructions that control the state of the e300 core, the address translation mechanism, and supervisor registers can be executed only when the core is operating in supervisor mode.

[Figure 7-2](#) shows all the core registers available at the user and supervisor level. The numbers to the right of the SPRs indicate the number that is used in the syntax of the instruction operands for the move to/from SPR instructions.



The following sections describe the e300-core-implementation-specific features as they apply to registers.

7.3.1.1 UISA Registers

UISA registers are user-level registers that include the following.

7.3.1.1.1 General-Purpose Registers (GPRs)

The PowerPC architecture defines 32 user-level GPRs that are 32 bits wide in 32-bit cores. The GPRs serve as the data source or destination for all integer instructions.

7.3.1.1.2 Floating-Point Registers (FPRs)

The PowerPC architecture also defines 32 user-level, 64-bit FPRs. The FPRs serve as the data source or destination for floating-point instructions. These registers can contain data objects of either single- or double-precision floating-point formats.

7.3.1.1.3 Condition Register (CR)

The CR is a 32-bit user-level register that provides a mechanism for testing and branching. It consists of eight 4-bit fields that reflect the results of certain operations, such as move, integer and floating-point comparisons, arithmetic, and logical operations.

7.3.1.1.4 Floating-Point Status and Control Register (FPSCR)

The user-level FPSCR contains all floating-point exception signal bits, exception summary bits, exception enable bits, and rounding control bits needed for compliance with the IEEE 754 standard.

7.3.1.1.5 User-Level SPRs

The PowerPC architecture defines numerous special purpose registers that serve a variety of functions, such as providing controls, indicating status, configuring the core, and performing special operations. During normal execution, a program can access the registers, as shown in [Figure 7-2](#), depending on the program's access privilege (supervisor or user, determined by the privilege-level bit, MSR[PR]). Note that GPRs and FPRs are accessed through operands that are part of the instructions. Access to registers can be explicit (that is, through the use of specific instructions for that purpose such as Move to Special-Purpose Register (**mtspr**) and Move from Special-Purpose Register (**mfspir**) instructions) or implicit, as the part of the execution of an instruction. Some registers are accessed both explicitly and implicitly. In the e300 core, all SPRs are 32 bits wide.

The following SPRs are accessible by user-level software:

- Link register (LR)—The LR can be used to provide the branch target address and to hold the return address after branch and link instructions. The LR is 32 bits wide in 32-bit implementations.
- Count register (CTR)—The CTR is decremented and tested automatically as a result of branch-and-count instructions. The CTR is 32 bits wide in 32-bit implementations.
- XER register—The 32-bit XER contains the summary overflow bit, integer carry bit, overflow bit, and a field specifying the number of bytes to be transferred by a Load String Word Indexed (**lswx**) or Store String Word Indexed (**stswx**) instruction.

7.3.1.2 VEA Registers

The VEA introduces the time base facility (TB) for reading. The TB is a 64-bit register pair whose contents are incremented once every four core input clock cycles. The TB consists of two 32-bit registers—time base upper (TBU) and time base lower (TBL). Note that the time base registers are read-only in user state.

7.3.1.3 OEA Registers

OEA registers are supervisor-level registers that include the following.

7.3.1.3.1 Machine State Register (MSR)

The MSR, shown in [Figure 7-3](#), is a supervisor-level register that defines the state of the core. The contents of this register are saved when an interrupt is taken, and restored when the interrupt handling completes. A critical interrupt is taken in the e300 core when the *cint* signal is asserted and MSR[CE] is set. The e300 core implements the MSR as a 32-bit register.

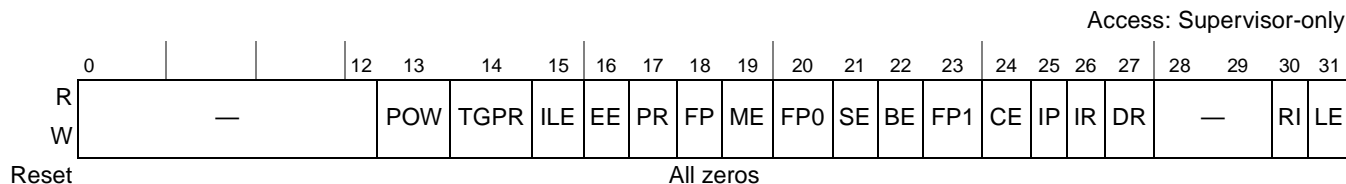


Figure 7-3. Machine State Register (MSR)

Table 7-1 shows the bit definitions for MSR.

Table 7-1. MSR Bit Descriptions

Bits	Name	Description
0 ¹	—	Reserved. Full function.
1–4 ¹	—	Reserved. Partial function.
5–9 ¹	—	Reserved. Full function.
10–12 ¹	—	Reserved. Partial function.
13	POW	Power management enable (implementation-specific) 0 Disables programmable power modes (normal operation mode) 1 Enables programmable power modes (nap, doze, or sleep mode). This bit controls the programmable power modes only; it has no effect on dynamic power management (DPM). MSR[POW] may be altered with an mtmsr instruction only. Also, when altering the POW bit, software may alter only this bit in the MSR and no others. The mtmsr instruction must be followed by a context-synchronizing instruction.
14	TGPR	Temporary GPR remapping (implementation-specific) 0 Normal operation 1 TGPR mode. GPR0–GPR3 are remapped to TGPR0–TGPR3 for use by TLB miss routines. The contents of GPR0–GPR3 remain unchanged while MSR[TGPR] = 1. Attempts to use GPR4–GPR31 with MSR[TGPR] = 1 yield undefined results. Temporarily replaces TGPR0–TGPR3 with GPR0–GPR3 for use by TLB miss routines. The TGPR bit is set when either an instruction TLB miss, data read miss, or data write miss interrupt is taken. The TGPR bit is cleared by an rfi instruction.

Table 7-1. MSR Bit Descriptions (continued)

Bits	Name	Description
15	ILE	Interrupt little-endian mode. When an interrupt occurs, this bit is copied into MSR[LE] to select the endian mode for the context established by the interrupt.
16	EE	External interrupt enable 0 The processor ignores external interrupts, system management interrupts, and decremter interrupts. 1 The processor is enabled to take an external interrupt, system management interrupt, or decremter interrupt.
17	PR	Privilege level 0 The processor can execute both user- and supervisor-level instructions 1 The processor can only execute user-level instructions
18	FP	Floating-point available 0 The processor prevents dispatch of floating-point instructions, including floating-point loads, stores, and moves. 1 The processor can execute floating-point instructions and can take floating-point enabled exception type program interrupts.
19	ME	Machine check enable 0 Machine check interrupts are disabled 1 Machine check interrupts are enabled
20	FE0	Floating-point exception mode 0
21	SE	Single-step trace enable 0 The processor executes instructions normally 1 The processor generates a trace interrupt upon the successful completion of the next instruction
22	BE	Branch trace enable 0 The processor executes branch instructions normally 1 The processor generates a trace interrupt upon the successful completion of a branch instruction
23	FE1	Floating-point exception mode 1
24	CE	Critical interrupt enable 0 Critical interrupts disabled 1 Critical interrupts enabled; critical interrupt and rfci instruction enabled The critical interrupt is an asynchronous implementation-specific interrupt. The critical interrupt vector offset is 0x00A00. The rfci instruction is implemented to return from these interrupt handlers. Also, CSRR0 and CSRR1 are used to save and restore the processor state for critical interrupts.
25	IP	Interrupt prefix. The setting of this bit specifies whether an interrupt vector offset is prepended with Fs or 0s. In the following description, <i>nnnn</i> is the offset of the interrupt. 0 Interrupts are vectored to the physical address 0x000n_nnnn 1 Interrupts are vectored to the physical address 0xFFFFn_nnnn
26	IR	Instruction address translation 0 Instruction address translation is disabled 1 Instruction address translation is enabled
27	DR	Data address translation 0 Data address translation is disabled 1 Data address translation is enabled
28–29 ¹	—	Reserved. Full function.

Table 7-1. MSR Bit Descriptions (continued)

Bits	Name	Description
30	RI	Recoverable interrupt (for system reset and machine check interrupts) 0 Interrupt is not recoverable 1 Interrupt is recoverable
31	LE	Little-endian mode enable 0 The processor runs in big-endian mode 1 The processor runs in little-endian mode.

¹ All reserved bits should be set to zero for future compatibility.

7.3.1.3.2 Segment Registers (SRs)

For memory management, 32-bit processors implement sixteen 32-bit SRs. To speed access, the core implements the SRs as two arrays: a main array, for data memory accesses, and a shadow array, for instruction memory accesses. Loading a segment entry with the Move to Segment Register (**mtsr**) instruction loads both arrays.

7.3.1.3.3 Supervisor-Level SPRs

The e300 core, like the G2_LE core, has additional supervisor-level SPRs, which are shown in [Figure 7-2](#). Two critical interrupt SPRs (CSRR0 and CSRR1), eight SPRGs (SPRG0–SPRG7), eight pairs of instruction BATs (IBAT0–IBAT7), eight pairs of data BATs (DBAT0–DBAT7), one system version register (SVR), one system memory base address (MBAR), one instruction address breakpoint control (IBCR), one data address breakpoint control (DBCR), a new instruction breakpoint register (IABR2), and two data address breakpoint registers (DABR and DABR2) are integrated into the core.

Supervisor-level SPRs include the following:

- The DSISR defines the cause of data access and alignment interrupts. The cause of a DSI interrupt for a data breakpoint (match with DABR and DABR2) can be determined by the value of the DSISR[DABR] bit (bit 9).
- The data address register (DAR) holds the address of an access after an alignment or DSI interrupt. For example, it contains the address of the breakpoint match condition.
- The decremter register (DEC) is a 32-bit decremting counter that provides a mechanism for causing a decremter interrupt after a programmable delay.
- SDR1 specifies the page table format used in virtual-to-physical address translation for pages. (Note that physical address is referred to as ‘real address’ in the architecture specification.)
- The machine status save/restore register 0 (SRR0) is used for saving the address of the instruction that caused the interrupt, and the address to return to when a Return from Interrupt (**rfi**) instruction is executed.
- The machine status save/restore register 1 (SRR1) is used to save machine status on interrupts and to restore machine status when an **rfi** instruction is executed.
- The SPRG0–SPRG7 registers are provided for operating system use. They reduce the latency that may be incurred in the saving of registers to memory while in a handler. Note that the e300 implements four more SPRGs than the G2 (SPRG0–SPRG3).

- The time base register (TB) is a 64-bit register that maintains the time of day and operates interval timers. It consists of two 32-bit fields: time base upper (TBU) and time base lower (TBL).
- The processor version register (PVR) is a read-only register that identifies the version (model) and revision level of the processor. See [Table 7-8](#) for the version and revision level of the PVR for the e300 processor core.
- Block address translation (BAT) arrays—The PowerPC architecture defines 16 BAT registers. The e300 core includes a total of eight pairs of DBAT and eight pairs of IBAT registers. See [Figure 7-2](#) for a list of the SPR numbers for the BAT arrays.

The following supervisor-level SPRs are implementation-specific (not defined in the PowerPC architecture):

- DMISS and IMISS are read-only registers that are loaded automatically on an instruction or data TLB miss.
- HASH1 and HASH2 contain the physical addresses of the primary and secondary page table entry groups (PTEGs).
- ICMP and DCMP contain a duplicate of the first word in the page table entry (PTE) for which the table search is looking.
- The required physical address (RPA) register is loaded by the core with the second word of the correct PTE during a page table search.
- The system version register (SVR) is available on the e300 core, which identifies the specific version (model) and revision level of the system-on-a-chip (SOC) integration.
- System memory base address (MBAR) is an implementation-specific register available on the e300 core. It supports a temporary storage for the system-level memory map.
- The instruction and data address breakpoint registers (IABR, IABR2, DABR, DABR2) are loaded with an instruction or data address, respectively, that is compared to instruction addresses in the dispatch queue or to the data address in the LSU. When an address match occurs, a breakpoint interrupt is generated.
- One instruction breakpoint control register (IBCR) and one data breakpoint control register (DBCR) are implemented in the e300 core.
- To support critical interrupts, two registers (CSRR0 and CSRR1) are included in the e300 core.
- Eight SPRG registers (SPRG0–SPRG7) are in the e300 core.
- Block address translation (BAT) arrays—The e300 core has eight instruction and eight data BAT registers.
- The hardware implementation (HID0 and HID1) registers provide the means for enabling core checkstops and features and allow software to read the configuration of the PLL configuration signals. The HID2 register enables the true little-endian mode, cache way-locking, and the additional BAT registers.

Table 7-2 shows the bit definitions for HID0.

Table 7-2. e300 HID0 Bit Descriptions

Bits	Name	Function
0	EMCP	Enable \overline{mcp} . The purpose of this bit is to mask out machine check interrupts caused by assertion of \overline{mcp} , similar to how MSR[EE] can mask external interrupts. 0 Masks \overline{mcp} . Asserting \overline{mcp} does not generate a machine check interrupt or a checkstop. 1 Asserting \overline{mcp} causes checkstop if MSR[ME] = 0 or a machine check interrupt if ME = 1
1	ECPE	Enable cache parity errors. 0 Disables instruction and data cache parity error reporting 1 Allows a detected cache parity error to cause a machine check interrupt if MSR[ME] = 1 or a checkstop if MSR[ME] = 0
2	EBA	Enable $\overline{ap_in[0:3]}$ and \overline{ape} for address parity checking. 0 Disables address parity checking during a snoop operation 1 Allows an address parity error during snoop operations to cause a checkstop if MSR[ME] = 0 or a machine check interrupt if MSR[ME] = 1 Note: Do not set this bit; the CSB does not have parity signals.
3	EBD	Enable \overline{dpe} for data parity checking. 0 Disables data parity checking 1 Allows a data parity error during reads to cause a checkstop if MSR[ME] = 0 or a machine check interrupt if MSR[ME] = 1 Note: Do not set this bit; the CSB does not have parity signals.
4	SBCLK	clk_out output enable. Used in conjunction with HID0[ECLK] and \overline{hreset} to configure clk_out . See Table 7-3 for settings.
5	—	Reserved
6	ECLK	clk_out output enable. Used in conjunction with HID0[SBCLK] and the \overline{hreset} signal to configure clk_out . See Table 7-3 for settings.
7	PAR	Disable precharge of $\overline{artry_out}$ 0 Precharge of $\overline{artry_out}$ enabled 1 Alters bus protocol slightly by preventing the processor from driving $\overline{artry_out}$ to high (negated) state. If this is done, the integrated device must restore the signals to the high state.
8	DOZE	Doze mode enable. Operates in conjunction with MSR[POW]. 0 Doze mode disabled 1 Doze mode enabled. Doze mode is invoked by setting MSR[POW] while this bit is set. In doze mode, the PLL, time base, and snooping remain active.
9	NAP	Nap mode enable. Operates in conjunction with MSR[POW]. 0 Nap mode disabled 1 Nap mode enabled. Nap mode is invoked by setting MSR[POW] while this bit is set. In nap mode, the PLL and time base remain active.
10	SLEEP	Sleep mode enable. Operates in conjunction with MSR[POW]. 0 Sleep mode disabled 1 Sleep mode enabled. Sleep mode is invoked by setting MSR[POW] while this bit is set. \overline{qreq} is asserted to indicate that the processor is ready to enter sleep mode. If the system logic determines that the processor may enter sleep mode, the quiesce acknowledge signal, \overline{qack} , is asserted back to the processor. Once \overline{qack} assertion is detected, the processor enters sleep mode after several processor clocks. At this point, the system logic may turn off the PLL by first configuring $pll_cfg[0:6]$ to PLL bypass mode, then disabling $sysclk$.

Table 7-2. e300 HID0 Bit Descriptions (continued)

Bits	Name	Function
11	DPM	Dynamic power management enable 0 Dynamic power management is disabled 1 Functional units enter a low-power mode automatically if the unit is idle. This does not affect operational performance and is transparent to software or any external hardware.
12–15	—	Reserved
16	ICE	Instruction cache enable 0 The instruction cache is neither accessed nor updated. All pages are accessed as if they were marked cache-inhibited (WIM = x1x). Potential cache accesses from the bus (snoop and cache operations) are ignored. In the disabled state for the L1 caches, the cache tag state bits are ignored and all instruction fetches are propagated to the coherent system bus (CSB) as single-beat transactions. For those transactions, however, $\bar{c}i$ reflects the state of the I bit in the MMU for that page regardless of cache disabled status. ICE is zero at power-up. 1 The instruction cache is enabled
17	DCE	Data cache enable 0 The data cache is neither accessed nor updated. All pages are accessed as if they were marked cache-inhibited (WIM = x1x). Potential cache accesses from the bus (snoop and cache operations) are ignored. In the disabled state for the L1 caches, the cache tag state bits are ignored and all data read and write accesses are propagated to the CSB as single-beat transactions. For those transactions, however, $\bar{c}i$ reflects the state of the I bit in the MMU for that page regardless of cache disabled status. DCE is zero at power-up. 1 The data cache is enabled
18	ILOCK	Instruction cache lock 0 Normal operation 1 The entire instruction cache is locked (that is, all eight ways of the cache are locked). A locked cache supplies data normally on a hit, but the access is treated as a cache-inhibited transaction on a miss. On a miss, the transaction to the bus is single-beat; however, $\bar{c}i$ still reflects the state of the I bit in the MMU for that page independent of cache locked or disabled status. To prevent locking during a cache access, an isync instruction must precede the setting of ILOCK.
19	DLOCK	Data cache lock 0 Normal operation 1 The entire data cache is locked (that is, all eight ways of the cache are locked). A locked cache supplies data normally on a hit, but is treated as a cache-inhibited transaction on a miss. On a miss, the transaction to the bus is single-beat; however, $\bar{c}i$ still reflects the state of the I bit in the MMU for that page independent of cache locked or disabled status. A snoop hit to a locked L1 data cache performs as if the cache were not locked. A cache block invalidated by a snoop remains invalid until the cache is unlocked. To prevent locking during a cache access, a sync instruction must precede the setting of DLOCK.
20	ICFI	Instruction cache Flash invalidate 0 The instruction cache is not invalidated. The bit is cleared when the invalidation operation begins (usually the next cycle after the write operation to the register). The instruction cache must be enabled for the invalidation to occur. 1 An invalidate operation is issued that marks the state of each instruction cache block as invalid. Cache access is blocked during this time. Setting ICFI clears all the valid bits of the blocks and the PLRU bits to point to way L0 of each set. For the e300 core, the proper use of the ICFI and DCFI bits is to set and clear them with two consecutive mtspr operations.

Table 7-2. e300 HID0 Bit Descriptions (continued)

Bits	Name	Function
21	DCFI	<p>Data cache Flash invalidate</p> <p>0 The data cache is not invalidated. The bit is cleared when the invalidation operation begins (usually the next cycle after the write operation to the register). The data cache must be enabled for the invalidation to occur.</p> <p>1 An invalidate operation is issued that marks the state of each data cache block as invalid without writing back modified cache blocks to memory. Cache access is blocked during this time. Bus accesses to the cache are signaled as a miss during invalidate-all operations. Setting DCFI clears all the valid bits of the blocks and the PLRU bits to point to way L0 of each set.</p> <p>For the e300 core, the proper use of the ICFI and DCFI bits is to set and clear them with two consecutive mtspr operations.</p>
22–23	—	Reserved, should be cleared.
24	IFEM	<p>Enable M bit on bus for instruction fetches</p> <p>0 M bit not reflected on bus for instruction fetches. Instruction fetches are treated as nonglobal on the bus.</p> <p>1 Instruction fetches reflect the M bit from the WIM settings</p>
25	DECAREN	<p>Decrementer auto reload</p> <p>0 Normal operation.</p> <p>1 Decrementer loads last mtdec value for precise periodic interrupt.</p>
26	—	Reserved, should be cleared.
27	FBIOB	<p>Force branch indirect on the bus</p> <p>0 Register indirect branch targets are fetched normally</p> <p>1 Forces register indirect branch targets to be fetched externally</p>
28	ABE	<p>Address broadcast enable. Controls whether certain address-only operations (such as cache operations) are broadcast on the bus.</p> <p>0 Address-only operations affect only local caches and are not broadcast</p> <p>1 Address-only operations are broadcast on the bus</p> <p>Affected instructions are dcbi, dcbf, and dcbst. Note that these cache control instruction broadcasts are not snooped by the e300 core. Refer to Section 4.3.3, “Data Cache Control,” for more information.</p>
29–30	—	Reserved
31	NOOPTI	<p>No-op the data cache touch instructions</p> <p>0 The dcbt and dcbst instructions are enabled</p> <p>1 The dcbt and dcbst instructions are no-oped internal to the e300 core</p>

Table 7-3 shows how HID0[ECLK] and HID0[SBCLK] are used to configure the *clk_out* signal.

Table 7-3. Using HID0[ECLK] and HID0[SBCLK] to Configure *clk_out*

\overline{hreset}	ECLK	SBCLK	<i>clk_out</i>
Asserted	x	x	Bus clock (small pulse for every rising edge of sysclk)
Negated	0	0	Clock output off
	0	1	Core clock/2
	1	0	Core clock
	1	1	Bus clock

Table 7-4 shows the bit definitions for HID1

Table 7-4. HID1 Bit Descriptions

Bits	Name	Description
0	PC0	PLL configuration bit 0 (read-only)
1	PC1	PLL configuration bit 1 (read-only)
2	PC2	PLL configuration bit 2 (read-only)
3	PC3	PLL configuration bit 3 (read-only)
4	PC4	PLL configuration bit 4 (read-only)
5	PC5	PLL configuration bit 5 (read-only)
6	PC6	PLL configuration bit 6 (read-only)
7–31	—	Reserved, should be cleared

Note: The clock configuration bits reflect the state of the *pll_cfg[0:6]* signals.

Table 7-5 shows the bit definitions for HID2.

Table 7-5. e300HID2 Bit Descriptions

Bits	Name	Description
0–3	—	Reserved, should be cleared.
4	LET	True little-endian. This bit enables true little-endian mode operation for instruction and data accesses. This bit is set to reflect the state of the <i>tle</i> signal at the negation of <i>hreset</i> . This bit is used in conjunction with MSR[LE] to determine the endian mode of operation. 0 No function 1 True little-endian mode, when MSR[LE] = 1 Changing the value of this bit during normal operation is not recommended
5	IFEB	Instruction fetch burst extension. This bit enables the instruction fetch burst extension. 0 Instruction fetch burst extension disabled 1 Instruction fetch burst extension enabled
6	—	Reserved, should be cleared.
7	MESISTATE	MESI state enable. This bit enables the four-state MESI cache coherency protocol. 0 MESI disabled. The data cache uses a three-state MEI coherency protocol. 1 MESI enabled. The data cache uses a four-state MESI protocol.
8	IFEC	Instruction fetch cancel extension. This bit enables the instruction fetch cancel extension. 0 Instruction fetch cancel extension disabled 1 Instruction fetch cancel extension enabled
9	EBQS	Enable BIU queue sharing. This bit enables data cache queue sharing. 0 Data cache queue sharing disabled 1 Data cache queue sharing enabled
10	EBPX	Enable BIU pipeline extension. This bit enables the bus interface unit pipeline extension. 0 BIU pipeline extension disabled; 1 level pipeline 1 BIU pipeline extension enabled; 1-1/2 level pipeline
11–12	—	Reserved for e300c1

Table 7-5. e300HID2 Bit Descriptions (continued)

Bits	Name	Description
13	HBE	High BAT enable. Regardless of the setting of HID2[HBE], these BATs are accessible by mf spr and mt spr . 0 IBAT[4–7] and DBAT[4–7] are disabled 1 IBAT[4–7] and DBAT[4–7] are enabled
14–15	—	Reserved, should be cleared.
16–18	IWLCK[0–2]	Instruction cache way-lock. Useful for locking blocks of instructions into the instruction cache for time-critical applications that require deterministic behavior. 000 no ways locked 001 way 0 locked 010 way 0 through way 1 locked 011 way 0 through way 2 locked 100 way 0 through way 3 locked in e300c1. 101 way 0 through way 4 locked in e300c1. 110 way 0 through way 5 locked in e300c1. 111 way 0 through way 6 locked in e300c1. Setting HID0[ILOCK] will lock all ways.
19	ICWP	Instruction cache way protection. Used to protect locked ways in the instruction cache from being invalidated. 0 Instruction cache way protection disabled 1 Instruction cache way protection enabled
20–23	—	Reserved, should be cleared.
24–26	DWLCK[0–2]	Data cache way-lock. Useful for locking blocks of data into the data cache for time-critical applications where deterministic behavior is required. 000 no ways locked 001 way 0 locked 010 way 0 through way 1 locked 011 way 0 through way 2 locked 100 way 0 through way 3 locked in e300c1. 101 way 0 through way 4 locked in e300c1. 110 way 0 through way 5 locked in e300c1. 111 way 0 through way 6 locked in e300c1. Setting HID0[DLOCK] will lock all ways.
27–31	—	Reserved, should be cleared.

7.3.2 Instruction Set and Addressing Modes

The following sections describe the PowerPC instruction set and addressing modes in general.

7.3.2.1 PowerPC Instruction Set and Addressing Modes

All PowerPC instructions are encoded as single-word (32-bit) opcodes. Instruction formats are consistent among all instruction types, permitting efficient decoding to occur in parallel with operand accesses. This fixed instruction length and consistent format simplifies instruction pipelining.

The PowerPC instructions are divided into the following categories:

- Integer instructions—These include computational and logical instructions.
 - Integer arithmetic instructions
 - Integer compare instructions
 - Integer logical instructions
 - Integer rotate and shift instructions
- Floating-point instructions—These include floating-point computational instructions, as well as instructions that affect the FPSCR.
 - Floating-point arithmetic instructions
 - Floating-point multiply/add instructions
 - Floating-point rounding and conversion instructions
 - Floating-point compare instructions
 - Floating-point status and control instructions
- Load/store instructions—These include integer and floating-point load and store instructions.
 - Integer load and store instructions
 - Integer load and store multiple instructions
 - Floating-point load and store
 - Primitives used to construct atomic memory operations (**lwarx** and **stwex**. instructions)
- Flow control instructions—These include branching instructions, condition register logical instructions, trap instructions, and other instructions that affect the instruction flow.
 - Branch and trap instructions
 - Condition register logical instructions
- Processor control instructions—These instructions are used for synchronizing memory accesses and management of caches, TLBs, and the segment registers.
 - Move to/from SPR instructions
 - Move to/from MSR
 - Synchronize
 - Instruction synchronize

- Memory control instructions—These instructions provide control of caches, TLBs, and segment registers.
 - Supervisor-level cache management instructions
 - Translation lookaside buffer management instructions. Note that there are additional implementation-specific instructions.
 - User-level cache instructions
 - Segment register manipulation instructions
- The e300 core implements the following instructions which are defined as optional by the PowerPC architecture:
 - Floating Select (**fsel**)
 - Floating Reciprocal Estimate Single-Precision (**fres**)
 - Floating Reciprocal Square Root Estimate (**frsqrte**)
 - Store Floating-Point as Integer Word (**stfiwx**)

Note that this grouping of instructions does not indicate the execution unit that executes a particular instruction or group of instructions.

Integer instructions operate on byte, half-word, and word operands. Floating-point instructions operate on single-precision (one word) and double-precision (one double word) floating-point operands. The PowerPC architecture uses instructions that are 4 bytes long and word-aligned. It provides for byte, half-word, and word operand loads and stores between memory and a set of 32 GPRs. It also provides for word and double-word operand loads and stores between memory and a set of 32 FPRs.

Computational instructions do not modify memory. To use a memory operand in a computation and then modify the same or another memory location, the memory contents must be loaded into a register, modified, and then written back to the target location with distinct instructions.

The core follows the program flow when it is in the normal execution state. However, the flow of instructions can be interrupted directly by the execution of an instruction or by an asynchronous event. Either kind of interrupt may cause one of several components of the system software to be invoked.

7.3.2.2 Implementation-Specific Instruction Set

The e300 core instruction set is defined as follows:

- The core provides hardware support for all 32-bit PowerPC instructions.
- The core provides two implementation-specific instructions used for software table search operations following TLB misses:
 - Load Data TLB Entry (**tlbld**)
 - Load Instruction TLB Entry (**tlbli**)
- The core implements the following instruction which is added to support critical interrupts (also supported on the G2_LE). This is a supervisor-level, context synchronizing instruction.
 - Return from Critical Interrupt (**rftci**)

- The core implements the following instruction which is added to support easy start-up initialization or reloading of the instruction cache.
 - Instruction Cache Block Touch (**icbt**)

7.3.3 Cache Implementation

The following sections describe the general cache characteristics as implemented in the PowerPC architecture and the core implementation.

7.3.3.1 PowerPC Cache Characteristics

The PowerPC architecture does not define hardware aspects of cache implementations. The e300 core controls the following memory access modes on a page or block basis:

- Write-back/write-through mode
- Caching-inhibited mode
- Memory coherency

Note that in the core, a cache block is defined as eight words. The VEA defines cache management instructions that provide a means by which the application programmer can affect the cache contents.

7.3.3.2 Implementation-Specific Cache Organization

The e300c1 provides independent, 32-Kbyte, eight-way, set-associative, instruction and data caches. provides 16-Kbyte, four-way set-associative instruction and data caches. The caches are physically addressed, and the data cache can operate in either write-back or write-through mode as specified by the PowerPC architecture.

The data cache is configured as 128 sets of 8 blocks each on the e300c1. Each block consists of 32 bytes, 2 state bits, and an address tag. The two state bits implement the three-state MEI (modified/exclusive/invalid) protocol. Each block contains eight 32-bit words. Note that the PowerPC architecture defines the term ‘block’ as the cacheable unit. For the core, the block size is equivalent to a cache line. A block diagram of the data cache organization is shown in [Figure 7-4](#).

The instruction cache is configured as 128 sets of 8 blocks each on the e300c1. Each block consists of 32 bytes, an address tag, and a valid bit. The instruction cache may not be written to, except through a block fill operation. In the e300 core, the instruction cache is blocked only until the critical load completes. The e300 core supports instruction fetching from other instruction cache lines following the forwarding of the critical-first-double-word of a cache line load operation. Successive instruction fetches from the cache line being loaded are forwarded, and accesses to other instruction cache lines can proceed during the cache line load operation. The instruction cache is not snooped, and cache coherency must be maintained by software. A fast hardware invalidation capability is provided to support cache maintenance. The organization of the instruction cache for the e300c1 is very similar to the data cache shown in [Figure 7-4](#).

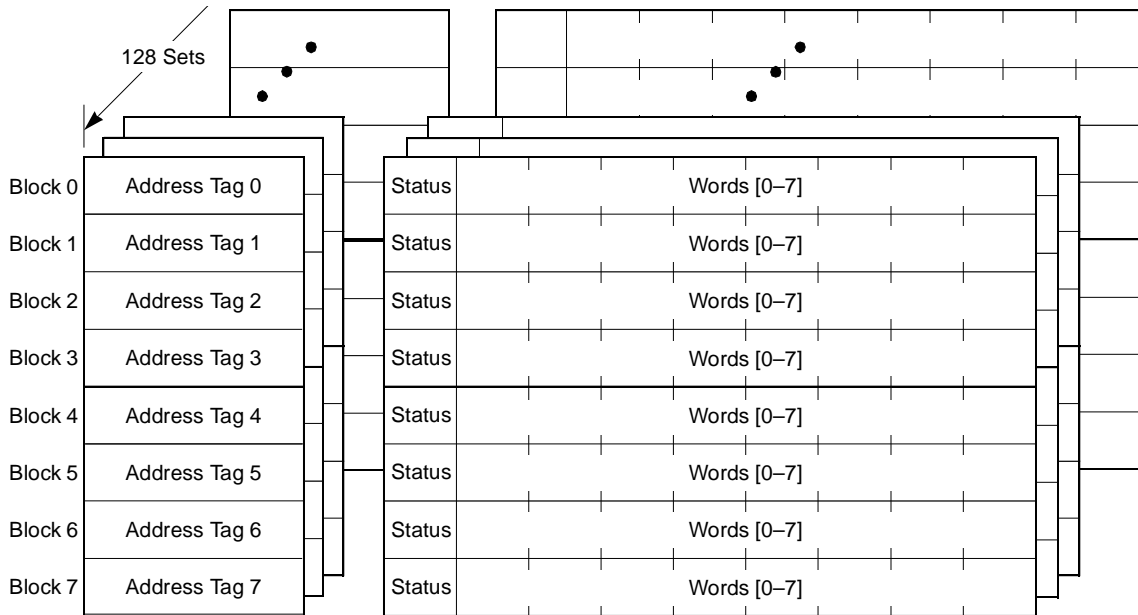


Figure 7-4. e300c1 Data Cache Organization

Each cache block contains eight contiguous words from memory that are loaded from an 8-word boundary (that is, bits A[27–31] of the effective addresses are zero); thus, a cache block never crosses a page boundary. Misaligned accesses across a page boundary can incur a performance penalty.

The e300 core cache blocks are loaded in four beats of 64 bits each on the 64-bit data bus. The burst load is performed as critical-double-word-first. The data cache is blocked to internal accesses until the load completes; the instruction cache allows sequential fetching during a cache block load. In the core, the critical-double-word is simultaneously written to the cache and forwarded to the requesting unit, thus minimizing stalls due to load delays.

To ensure coherency among caches in a multiprocessor (or multiple caching-device) implementation, the core implements the MEI protocol during normal operation of the data cache. The new data cache MESI extension supports the additional fourth cache coherency shared state for the data cache. To support this feature, the shared signal, *shd*, has been added to the bus interface. Although the MESI protocol is supported by the e300 core, it is not implemented on MPC8360. The following four states indicate the state of the cache block:

- **Modified**—The cache block is modified with respect to system memory; that is, data for this address is valid only in the cache and not in system memory.
- **Exclusive**—This cache block holds valid data that is identical to the data at this address in system memory. No other cache has this data.
- **Shared**—Only available if HID2[MESISTATE] register bit is set. The address block is valid in the cache and in at least one other cache. This block is always consistent with system memory. That is, the shared state is shared-unmodified; there is no shared-modified state. Although the MESI protocol is supported by the e300 core, it is not implemented on MPC8360.
- **Invalid**—This cache block does not hold valid data.

Cache coherency is enforced by on-chip bus snooping logic. Because the e300 core data cache tags are single-ported, a simultaneous load/store and snoop access represents a resource contention. The snoop access is given first access to the tags. The load or store then occurs on the clock following the snoop.

Parity is now integrated into both instruction and data cache memory. A machine check interrupt is now taken upon the detection of an instruction or data cache parity error. Parity is checked whenever valid data is returned from the instruction or data cache for a cache hit or whenever valid data is read out of the cache for a castout or snoop-push operation.

7.3.3.3 Instruction and Data Cache Way-Locking

The e300 core implements instruction and data cache way-locking, which guarantees that certain memory accesses will hit in the cache. This provides deterministic access times for those accesses.

7.3.4 Interrupt Model

This section describes the PowerPC interrupt model and the e300 core implementation specifically.

7.3.4.1 PowerPC Interrupt Model

The PowerPC interrupt mechanism allows the core to change to supervisor state as a result of external signals, errors, or unusual conditions arising in the execution of instructions. The conditions that can cause interrupts are called exceptions. When interrupts occur, information about the state of the core is saved to certain registers and the core begins execution at an address (interrupt vector) predetermined for each interrupt type. Interrupts are processed in supervisor mode.

Some interrupts, such as program interrupts, can be triggered by a broad range of exception conditions. Other interrupts, such as the decremter interrupt, have only a single exception condition. Although multiple exception conditions can map to a single interrupt vector, a more specific condition may be determined by examining a register associated with the interrupt—for example, the DSISR and the FPSCR. Additionally, some exception conditions can be explicitly enabled or disabled by software.

The PowerPC architecture requires that interrupts be handled in program order; therefore, although a particular implementation may recognize exception conditions out of order, they are presented strictly in order. When an instruction-caused interrupt is recognized, any unexecuted instructions that appear earlier in the instruction stream, including any that have not yet entered the execute stage, are required to complete before the interrupt is taken. Any interrupts caused by those instructions are handled first. Likewise, asynchronous, precise interrupts are recognized when they occur, but are not handled until the instruction currently in the completion stage successfully completes execution or generates an interrupt, and the completed store queue is emptied.

Unless a catastrophic condition causes a system reset or machine check interrupt, only one interrupt is handled at a time. If, for example, a single instruction encounters multiple interrupt conditions, those conditions are handled sequentially. After the interrupt handler completes, the instruction execution continues until the next interrupt condition is encountered. However, in many cases there is no attempt to re-execute the instruction. This method of recognizing and handling interrupts sequentially guarantees that interrupts are recoverable.

To prevent the program state from being lost due to a system reset, a machine check interrupt, or an instruction-caused interrupt in the interrupt handler, interrupt handlers should save the information stored in SRR0 and SRR1 early and before enabling external interrupts.

The PowerPC architecture supports four types of interrupts:

- Synchronous, precise—These are caused by instructions. All instruction-caused interrupts are handled precisely; that is, the machine state at the time the interrupt occurs is known and can be completely restored. This means that (excluding the trap and system call interrupts) the address of the faulting instruction is provided to the interrupt handler and neither the faulting instruction nor subsequent instructions in the code stream will complete execution before the interrupt is taken. Once the interrupt is processed, execution resumes at the address of the faulting instruction (or at an alternate address provided by the interrupt handler). When an interrupt is taken due to a trap or system call instruction, execution resumes at an address provided by the handler.
- Synchronous, imprecise—The PowerPC architecture defines two imprecise floating-point exception modes: recoverable and nonrecoverable. Even though the core provides a means to enable the imprecise modes, it implements these modes identically to the precise mode (that is, all enabled floating-point exceptions are always precise on the core).
- Asynchronous, maskable—The external system management interrupt (SMI) and decremter interrupts are maskable, asynchronous interrupts. When these interrupts occur, their handling is postponed until the next instruction and any of its associated interrupts complete execution. If there are no instructions in the execution units, the interrupt is taken immediately upon determination of the correct restart address (for loading SRR0).
- Asynchronous, nonmaskable—The system reset and the machine check interrupt are nonmaskable, asynchronous interrupts. They may not be recoverable, or they may provide a limited degree of recoverability. All interrupts report recoverability through MSR[RI].

7.3.4.2 Implementation-Specific Interrupt Model

As specified by the Power Architecture, all interrupts can be described as either precise or imprecise and either synchronous or asynchronous. Asynchronous interrupts (some of which are maskable) are caused by events external to the processor’s execution; synchronous interrupts, which are all handled precisely by the e300 core, are caused by instructions. A system management interrupt is an implementation-specific interrupt. The interrupt classes are shown in [Table 7-6](#).

Table 7-6. Interrupt Classifications

Synchronous/Asynchronous	Precise/Imprecise	Interrupt Type
Asynchronous, nonmaskable	Imprecise	Machine check System reset
Asynchronous, maskable	Precise	External interrupt Decrementer System management interrupt Critical interrupt
Synchronous	Precise	Instruction-caused interrupts

Although interrupts have other characteristics, such as whether they are maskable, the distinctions shown in Table 7-6 define categories of interrupts that the core handles uniquely. Note that Table 7-6 includes no synchronous, imprecise instructions. While the PowerPC architecture supports imprecise handling of floating-point exceptions, the core implements floating-point exception modes as precise.

The e300 core interrupts and exception conditions that cause them are listed in Table 7-7.

Table 7-7. Exceptions and Interrupts

Interrupt Type	Vector Offset (hex)	Exception Conditions
Reserved	00000	—
System reset	00100	Caused by the assertion of either \overline{hreset} .
Machine check	00200	Caused by the assertion of the \overline{tea} signal during a data bus transaction, assertion of \overline{mcp} , an address or data parity error, or an instruction or data cache parity error. Note that the e300 has SRR1 register values that are different from the G2/G2_LE cores' when a machine check occurs.
DSI	00300	Determined by the bit settings in the DSISR, listed as follows: 1 Set if the translation of an attempted access is not found in the primary hash table entry group (HTEG), or in the rehashed secondary HTEG, or in the range of a DBAT register; otherwise cleared 4 Set if a memory access is not permitted by the page or DBAT protection mechanism; otherwise cleared 6 Set for a store operation and cleared for a load operation 9 Set if a data address breakpoint interrupt occurs when the data [0–28] in the DABR or DABR2 matches the next data access (load or store instruction) to complete in the completion unit. The different breakpoints are enabled as follows: • Write breakpoints enabled when DABR[30] is set • Read breakpoints enabled when DABR[31] is set
ISI	00400	Caused when an instruction fetch cannot be performed for any of the following reasons: • The effective (logical) address cannot be translated. That is, there is a page fault for this portion of the translation, so an ISI interrupt must be taken to load the PTE (and possibly the page) into memory. • The fetch access violates memory protection (indicated by SRR1[4] set). If the key bits (Ks and Kp) in the segment register and the PP bits in the PTE are set to prohibit read access, instructions cannot be fetched from this location.
External interrupt	00500	Caused when MSR[EE] = 1 and the \overline{int} signal is asserted.
Alignment	00600	Caused when the core cannot perform a memory access for any of the reasons described below: • The operand of a floating-point load or store instruction is not word-aligned. • The operands of lmw , stmw , lwarx , and stwcx instructions are not aligned. • The instruction is lswi , lswx , stswi , stswx , and the core is in little-endian mode. Note that PowerPC little-endian mode is not supported on the e300 core. • The operand of dcbz is in memory that is write-through-required or caching-inhibited.

Table 7-7. Exceptions and Interrupts (continued)

Interrupt Type	Vector Offset (hex)	Exception Conditions
Program	00700	<p>Caused by one of the following exception conditions, which correspond to bit settings in SRR1 and arise during execution of an instruction.</p> <p>Floating-point enabled exception—A floating-point enabled exception condition is generated when the following condition is met: (MSR[FE0] MSR[FE1]) and FPSCR[FEX] is 1.</p> <ul style="list-style-type: none"> FPSCR[FEX] is set by the execution of a floating-point instruction that causes an enabled exception or by the execution of one of the Move to FPSCR instructions that results in both an exception condition bit and its corresponding enable bit being set in the FPSCR. Illegal instruction—An illegal instruction program interrupt is generated when execution of an instruction is attempted with an illegal opcode or illegal combination of opcode and extended opcode fields (including PowerPC instructions not implemented in the core), or when execution of an optional instruction not provided in the core is attempted (these do not include those optional instructions that are treated as no-ops). Privileged instruction—A privileged instruction program interrupt is generated when the execution of a privileged instruction is attempted and the MSR register user privilege bit, MSR[PR], is set. In the e300 core, this interrupt is generated for mtspr or mf spr with an invalid SPR field if SPR[0] = 1 and MSR[PR] = 1. This may not be true for all cores that implement the PowerPC architecture. Trap—A trap type program interrupt is generated when any of the conditions specified in a trap instruction are met.
Floating-point unavailable	00800	Caused by an attempt to execute a floating-point instruction (including floating-point load, store, and move instructions) when the floating-point available bit (MSR[FP]) is cleared.
Decrementer	00900	Occurs when DEC[0] changes from 0 to 1. This interrupt is enabled with MSR[EE].
Critical interrupt	00A00	Taken when \overline{cint} is asserted and MSR[CE] = 1.
Reserved	00B00–00BFF	—
System call	00C00	Occurs when a System Call (sc) instruction is executed.
Trace	00D00	Taken when MSR[SE] = 1 or when the currently completing instruction is a branch and MSR[BE] = 1.
Reserved	00E00	The e300 core does not generate an interrupt to this vector. Other devices may use this vector for floating-point assist interrupts.
Instruction translation miss	01000	Caused when the effective address for an instruction fetch cannot be translated by the ITLB.
Data load translation miss	01100	Caused when the effective address for a data load operation cannot be translated by the DTLB.
Data store translation miss	01200	Caused when the effective address for a data store operation cannot be translated by the DTLB, or when a DTLB hit occurs and the change bit in the PTE must be set due to a data store operation.
Instruction address breakpoint	01300	Occurs when the address (bits 0–29) in the IABR matches the next instruction to complete in the completion unit, and IABR[30] is set. Note that the e300 core also implements IABR2, which functions identically to IABR.

Table 7-7. Exceptions and Interrupts (continued)

Interrupt Type	Vector Offset (hex)	Exception Conditions
System management interrupt	01400	Caused when MSR[EE] = 1 and the \overline{smi} input signal is asserted.
Reserved	01500–02FFF	—

7.3.5 Memory Management

The following sections describe the memory management features of the PowerPC architecture and the e300 core implementation, respectively.

7.3.5.1 PowerPC Memory Management

The primary functions of the MMU are to translate logical (effective) addresses to physical addresses for memory accesses and to provide access protection on blocks and pages of memory.

The core generates two types of accesses that require address translation: instruction accesses and data accesses to memory generated by load and store instructions.

The PowerPC MMU and interrupt model support demand-paged virtual memory. Virtual memory management permits execution of programs larger than the size of physical memory; demand-paged implies that individual pages are loaded into physical memory from system memory only when they are first accessed by an executing program.

The hashed page table is a variable-sized data structure that defines the mapping between virtual page numbers and physical page numbers. The page table size is a power of two, and its starting address is a multiple of its size.

The page table contains a number of page-table entry groups (PTEGs). A PTEG contains eight page-table entries (PTEs) of 8 bytes each; therefore, each PTEG is 64 bytes long. PTEG addresses are entry points for table search operations.

Address translations are enabled by setting bits in the MSR—MSR[IR] enables instruction address translations, and MSR[DR] enables data address translations.

7.3.5.2 Implementation-Specific Memory Management

The instruction and data memory management units in the e300 core provide 4 Gbytes of logical address space accessible to supervisor and user programs with a 4-Kbyte page size and 256-Mbyte segment size. Block sizes range from 128 Kbytes to 256 Mbytes and are software selectable. In addition, the core uses an interim 52-bit virtual address and hashed page tables for generating 32-bit physical addresses. The MMUs in the e300 core rely on the interrupt processing mechanism for the implementation of the paged virtual memory environment and for enforcing protection of designated memory areas.

Instruction and data TLBs provide address translation in parallel with the on-chip cache access, incurring no additional time penalty in the event of a TLB hit. A TLB is a cache of the most recently used page table

entries. Software is responsible for maintaining the consistency of the TLB with memory. The core TLBs are 64-entry, two-way, set-associative caches that contain instruction and data address translations. The core provides hardware assist for software table search operations through the hashed page table on TLB misses. Supervisor software can invalidate TLB entries selectively.

For instructions and data that correspond to block address translation, the e300 core provides independent eight-entry BAT arrays. These entries define blocks that can vary from 128 Kbytes to 256 Mbytes. The BAT arrays are maintained by system software. HID2[HBE] is added to the e300 for enabling or disabling the four additional pairs of BAT registers. However, regardless of the setting of HID2[HBE], these BATs are accessible by **mfspr** and **mtspr**.

As specified by the PowerPC architecture, the hashed page table is a variable-sized data structure that defines the mapping between virtual page numbers and physical page numbers. The page table size is a power of two, and its starting address is a multiple of its size.

Also as specified by the PowerPC architecture, the page table contains a number of PTEGs. A PTEG contains 8 PTEs of 8 bytes each; therefore, each PTEG is 64 bytes long. PTEG addresses are entry points for table search operations.

7.3.6 Instruction Timing

The e300 core is a pipelined superscalar processor core. Because instruction processing is reduced into a series of stages, an instruction does not require all of the resources of an execution unit at the same time. For example, after an instruction completes the decode stage, it can pass on to the next stage, while the subsequent instruction can advance into the decode stage. This improves the throughput of the instruction flow. For example, it may take three cycles for a single floating-point instruction to execute, but if there are no stalls in the floating-point pipeline, a series of floating-point instructions can have a throughput of one instruction per cycle.

The core instruction pipeline has four major pipeline stages, described as follows:

- The fetch pipeline stage primarily involves retrieving instructions from the memory system and determining the location of the next instruction fetch. Additionally, if possible, the BPU decodes branches during the fetch stage and folds out branch instructions before the dispatch stage.
- The dispatch pipeline stage is responsible for decoding the instructions supplied by the instruction fetch stage and determining which of the instructions are eligible to be dispatched in the current cycle. In addition, the source operands of the instructions are read from the appropriate register file and dispatched with the instruction to the execute pipeline stage. At the end of the dispatch pipeline stage, the dispatched instructions and their operands are latched by the appropriate execution unit.
- In the execute pipeline stage, each execution unit with an instruction executes the selected instruction (perhaps over multiple cycles), writes the instruction's result into the appropriate rename register, and notifies the completion stage when the execution has finished. In the case of an internal interrupt, the execution unit reports the interrupt to the completion/write-back pipeline stage and discontinues instruction execution until the interrupt is handled. The interrupt is not signaled until that instruction is the next to be completed. Execution of most floating-point instructions is pipelined within the FPU, allowing up to three instructions to execute in the FPU concurrently. The FPU pipeline stages are multiply, add, and round-convert. The LSU has two

pipeline stages: the first stage, for effective address calculation and MMU translation, and the second, for accessing data in the cache.

- The complete/write-back pipeline stage maintains the correct architectural machine state and transfers the contents of the rename registers to the GPRs and FPRs as instructions are retired. If the completion logic detects an instruction causing an interrupt, all subsequent instructions are canceled, their execution results in rename registers are discarded, and instructions are fetched from the correct instruction stream.

A superscalar processor core issues multiple, independent instructions into multiple pipelines, allowing instructions to execute in parallel. The e300c1 core has independent execution units for: integer instructions, floating-point instructions, branch instructions, load/store instructions, and system register instructions. The IU and the FPU each have dedicated register files for maintaining operands (GPRs and FPRs, respectively), allowing integer and floating-point calculations to occur simultaneously without interference.

The core provides support for single-cycle store, and it provides an adder/comparator in the system register unit that allows the dispatch and execution of multiple integer add and compare instructions on each cycle.

Because the PowerPC architecture can be applied to such a wide variety of implementations, instruction timing among processor cores varies accordingly.

7.3.7 Core Interface

The core interface is specific for each processor core implementation.

The MPC8360 contains an internal coherent system bus (CSB) that interfaces the processor core to the peripheral logic. This internal bus is very similar in function to the external 60x bus interface on the MPC603e. In the case of the MPC8360, the CSB system logic decodes e300-initiated transactions and directs all accesses to the appropriate interface.

The e300 core can operate at a variety of frequencies allowing the designer to trade off performance for power consumption. The processor core is clocked from a separate PLL, which is referenced to the CSB frequency. This allows the processor core and the peripheral logic to operate at different frequencies.

The e300 core provides a versatile core interface that allows for a wide range of implementations. The interface includes a 32-bit address bus, a 64-bit data bus, and 56 control and information signals (see [Figure 7-5](#)). The core interface allows for address-only transactions, as well as address and data transactions. The core control and information signals include the address arbitration, address start, address transfer, transfer attribute, address termination, data arbitration, data transfer, data termination, and core state signals. Test and control signals provide diagnostics for selected internal circuits.

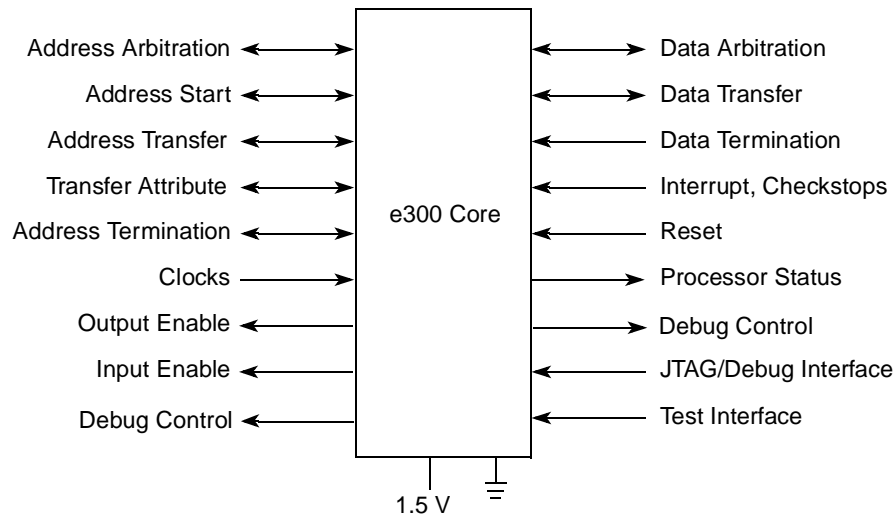


Figure 7-5. Core Interface

The core interface supports bus pipelining, allowing the address tenure of one transaction to overlap the data tenure of another. The extent of the pipelining depends on external arbitration and control circuitry. Similarly, the core supports split-bus transactions for systems with multiple potential bus masters—one device can have mastership of the address bus while another has mastership of the data bus. Allowing multiple bus transactions to occur simultaneously increases the available bus bandwidth for other activity and, as a result, improves performance.

The core clocking structure allows the bus to operate at integer multiples of the core cycle time.

The following sections describe the core bus support for memory operations. Note that some signals perform different functions depending on the addressing protocol used.

7.3.7.1 Memory Accesses

The e300 core CSB is a 64-bit data bus.

With a 64-bit CSB, memory accesses allow transfer sizes of 8, 16, 24, 32, 40, 48, 56, or 64 bits in one bus clock cycle. Data transfers occur in either single-beat transactions or four-beat burst transactions. Single-beat transactions are caused by noncached accesses that access memory directly (that is, reads and writes when caching is disabled, caching-inhibited accesses, and stores in write-through mode). Four-beat burst transactions, which always transfer an entire cache block (32 bytes), are initiated when a line is read from or written to memory.

7.3.7.2 Signals

The e300 core signals are grouped as follows:

- **Interrupts/Resets**—These signals include the external interrupt signal (\overline{int}), critical interrupt signal (\overline{cint}), checkstop signals, and both soft reset and hard reset signals. They are used to interrupt and, under various conditions, to reset the core.

- JTAG/debug interface signals—The JTAG (based on the IEEE 1149.1 standard) interface and debug unit provides a serial interface to the system for performing monitoring and boundary tests. Two additional signals are added to the e300 core to allow observation of the internal clock state of the core (*stopped*) and to allow the external input to force the core into a halted state (*ext_halt*).
- Core status and control—These signals include the memory reservation signal, machine quiesce control signals, time base/decrementer clock base enable signal, and the *tlbisync* signal.
- Clock control—These signals provide for system clock input and frequency control.
- Test interface signals—Signals like address matching, combinational matching, and watchpoint are used in the core for production testing.
- Transfer attribute signals—These signals provide information about the type of transfer, such as the transfer size and whether the transaction is bursted, write-through, or cache-inhibited.

7.3.8 Debug Features

Some new debug features are specific to the e300 core. Accesses to the debug facilities are available only in supervisor mode by using the **mtspr** and **mfspr** instructions. The e300 provides the following additional feature in the JTAG/debug interface: Inclusion of breakpoint status and control pins: *stopped* and *ext_halt*.

7.3.8.1 Breakpoint Signaling

The breakpoint signaling provided on the e300 core allows observability of breakpoint matches external to the core. The *iabr*, *iabr2*, *dabr*, and *dabr2* breakpoint signals are asserted for at least one bus clock cycle when the respective breakpoint occurs. The status of the run state of the e300 core is indicated by the *stopped* pin. An asynchronous external breakpoint can be asserted to the e300 core using the *ext_halt* pin:

- When DBCR and IBCR are configured for an OR combinational signal type, the breakpoint signals *iabr*, *iabr2* and *dabr*, *dabr2* reflect their respective breakpoints.
- When the DBCR and IBCR are configured for AND combinational signal type, only the *iabr2* and *dabr2* breakpoint signals are asserted after the AND condition is met (that is, both instruction breakpoints occurred or both data breakpoints occurred).
- When the core_stopped pin is asserted, the e300 core has entered a stopped state and all internal clocking has stopped, indicating that a hardware debug event has occurred.
- The *ext_halt* input pin can be used to force the core into halted state. The halted state may be a hardstop, conditional upon the HARDSTOP condition being set through the JTAG/debug interface

7.4 Differences Between Cores

The e300 core has similar functionality to the G2_LE core. [Table 7-8](#) describes the differences between the G2_LE and the e300.

Table 7-8. Differences Between e300 and G2_LE Cores

e300 Core	G2_LE Core	Impact
New HID0 bits	—	The e300 core has a new HID0 bit defined to enable cache parity error reporting (ECPE).
New HID1 bits	—	The e300 core has new HID1 bits defined to extend the number of PLL configuration signals to seven (PC5, PC6).
New HID2 bits	—	The e300 core has new HID2 bits defined to support instruction fetch bursting (IFEB), MESI coherency protocol (MESI), instruction fetch cancels (IFEC), data cache queue sharing (EBQS), pipelining extension (EBPX), additional cache way locking (IWLCK and DWLCK), and instruction cache way protection (ICWP).
New PVR register value	—	The processor version register values differ.
New IBCR and DBCR bits	—	The e300 core has new IBCR[IABRSTAT, IABR2STAT] and DBCR[DABR1STAT, DABR2STAT] fields to provide instruction and data address breakpoint status.
—	16-Kbyte, four-way, set-associative, instruction and data caches	Some e300 cores may have different cache sizes than the G2_LE. See the <i>e300 PowerPC Core Reference Manual</i> for detailed information.
L1 cache parity	—	The e300 core supports parity for both instruction and data caches; the G2_LE does not support cache parity.
MEI or MESI coherency protocols	MEI protocol only	The e300 supports two coherency protocols: MEI and MESI; the G2_LE only supports the MEI protocol. Not implemented on MPC8360.
Instruction cancel extension	—	The e300 instruction cancel mechanism improves utilization of instruction cache by supporting 'hits-under-cancels' and 'misses-under-cancels'; the G2_LE requires the cancel to complete before new instruction fetches can begin.
Instruction fetch bursts to caching-inhibited space	Single-beat instruction fetches to caching-inhibited space	The e300's instruction fetch burst extension allows all caching-inhibited instruction fetches to be performed on the bus as burst transactions, even though the instructions are not cached. This improves performance for instruction space that is caching-inhibited, because up to eight instructions are returned with one bus operation. The G2_LE core must use single-beat instruction fetches for caching-inhibited space, returning only two instructions per bus operation.
Instruction cache way protection	—	The e300 core can protect locked ways in the instruction cache from invalidation; the G2_LE does not support instruction cache way protection.

Table 7-8. Differences Between e300 and G2_LE Cores (continued)

e300 Core	G2_LE Core	Impact
Data cache queue sharing	—	The e300 has a new data cache queue sharing extension that allows the two burst-write queues in the bus unit to be used interchangeably for cache replacements and snoop pushes. Thus, the data cache can support two outstanding cache replacements or two outstanding snoop push operations on the bus at any given time.
icbt instruction	—	The e300 supports a new instruction cache block touch instruction that facilitates preloading the instruction cache before locking; the G2_LE core requires speculatively fetching instructions before locking the instruction cache.
1-1/2-level bus pipelining	1-level bus pipelining	For the e300, a new transaction can complete an address tenure when the previous transaction has been granted the data bus; for the G2_LE, a new transaction must wait until the previous data tenure has completed before completing its address tenure.
PowerPC little-endian not supported	PowerPC little-endian supported	PowerPC little-endian will not be supported in the e300 core, although true little-endian will be fully supported.
Data retry mode removed	Data retry mode available	\overline{drtry} and $drtrymode$ will no longer be supported on the e300 and future versions.
External control instructions removed	External control instructions available	The eciwx and ecowx instruction pair will not be supported on the e300 core. These are optional instructions in the PowerPC architecture.
Reduced pin mode removed	Reduced pin mode available	Reduced pinout mode and the signal $redpinmode$ will not be supported in the e300 core.



Chapter 8

Integrated Programmable Interrupt Controller (IPIC)

This chapter describes the integrated programmable interrupt controller (IPIC), including a definition of the external signals and their functions. Also, the configuration, control, and status registers are described in this chapter. Note that individual chapters in this reference manual describe specific initialization aspects for each individual block.

8.1 IPIC Introduction

This chapter describes the IPIC interrupt protocol, various types of interrupt sources controlled by the IPIC unit, and the IPIC registers with some programming guidelines. The programming model is similar to the interrupt controller of the MPC8260. The interrupt controller provides interrupt management that is responsible for receiving hardware-generated interrupts from different sources (both internal and external). It also prioritizes and delivers the interrupts to the CPU for servicing. The IPIC prioritizes and manages interrupts from the following controller units:

- Dual DDR memory controllers (DDR): DDR and secondary DDR
- Local bus memory controller (LBC)
- PCI
- Four-channel DMA controller (DMA)
- Message unit (MU)
- DUART communication module (DUART)
- Security engine (SEC)
- System bus arbiter (SBA)
- Periodic interval timer (PIT)
- Real time clock timer (RTC ALR and RTC SEC)
- Eight global timers (GTM1–GTM8)
- Software watchdog timer (WDT)
- I²C controller (I²C)
- Power management controller (PMC)
- QUICC Engine block
- External pins ($\overline{\text{IRQ}}[0:7]$)

The interrupt sources controlled by the IPIC unit cause exceptions in the processor core. The internal interrupt (*int*) signal is the main interrupt output from the IPIC to the core and it causes the regular interrupt exception. The *cint* signal is the critical interrupt output from the IPIC to the processor core and causes the critical interrupt exception. The *smi* signal is the system management interrupt output from the IPIC to the processor core and causes the system management interrupt exception. The machine check exception is

caused by the internal \overline{mcp} signal generated by the IPIC, informing the host processor of error conditions, assertion of the external $\overline{IRQ0}$ machine-check request (enabled when $SEMSR[SIRQ0] = 1$), and other conditions.

Table 8-1 shows the relationship of the various functional blocks and external signals of the device to the IPIC unit.

The IPIC receives interrupt request signals from the following two sources:

- External to the integrated device
- Internal to the integrated device

The unit selects the highest priority interrupt from all current interrupts and forwards it to the internal processor core, or off-chip for external servicing.

The IPIC also manages an internal non-maskable machine-check processor (\overline{mcp}) signal and the interrupt generated by the off-chip interrupt sources ($\overline{IRQ}[0:7]$).

The interrupt router of the IPIC monitors the outputs of the internal configuration registers. When the priority is highest in one of the received interrupt signals, the IPIC sets the corresponding bit in one of the interrupt pending registers—system internal interrupt pending register (SIPNR)/system external interrupt pending register (SEPNR). If the interrupt is not masked, the IPIC asserts the \overline{int} signal to indicate an interrupt request to the processor. When the processor is running the specific \overline{int} , \overline{cint} , or \overline{smi} interrupt handler code, the processor must vectorize the external interrupt handler by explicitly (in software) reading the corresponding interrupt vector register (SIVCR, SCVCR or SMVCR). In response to this read, the IPIC unit returns the vector (associated with the interrupt source) to the interrupt handler routine. In addition, the handler can vectorize different branches of interrupt handling.

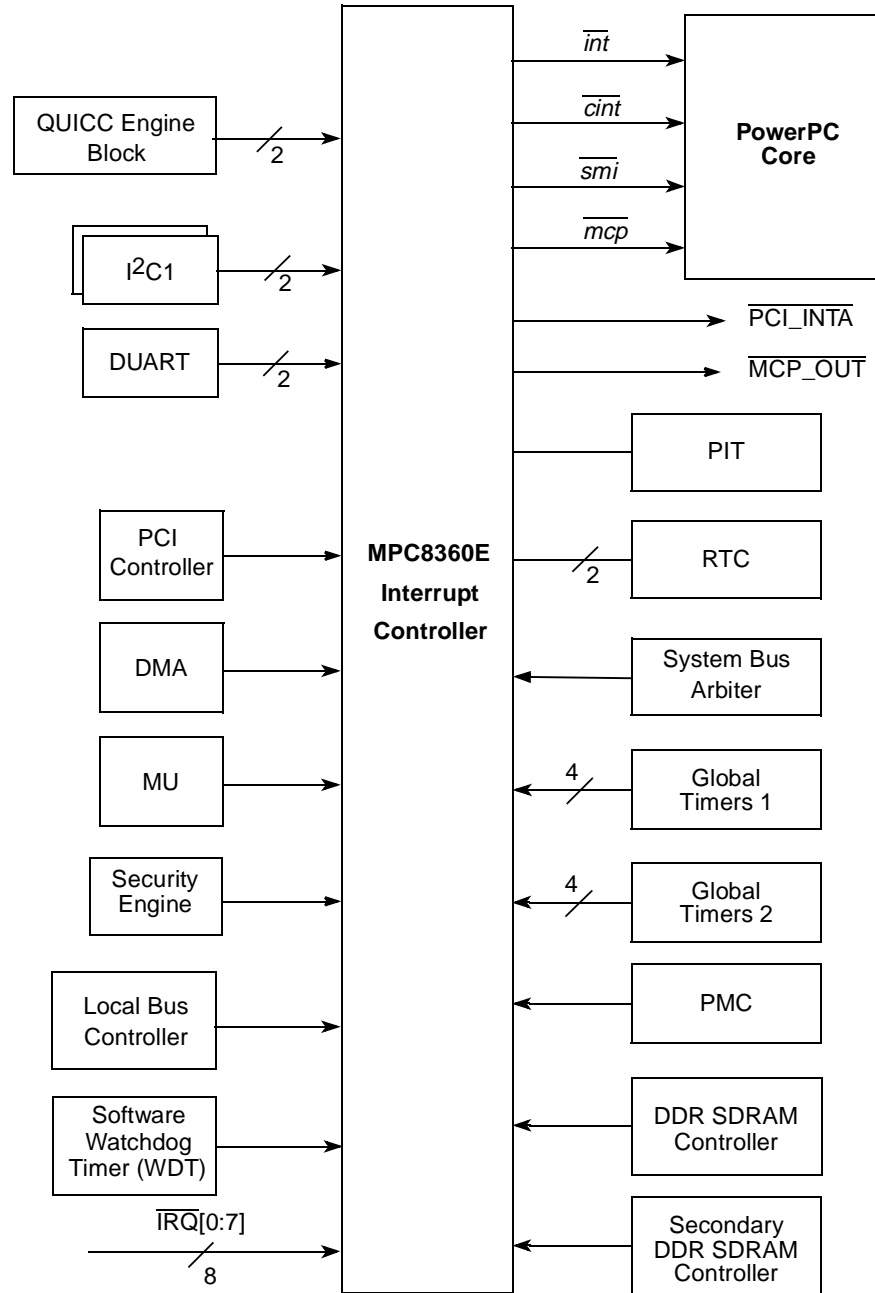


Figure 8-1. Interrupt Sources Block Diagram

The IPIC receives the following types of interrupts:

- External interrupt—triggered by the off-chip signals (\overline{IRQn}) listed in [Table 8-1](#)
- Internal interrupts—on-chip interrupts, triggered by the sources listed in [Table 8-8](#) and [Table 8-10](#)
- External and internal non-maskable machine check conditions, signaled by the sources listed in [Table 8-22](#) through \overline{mcp}

The interrupt controller provides the ability to mask each interrupt source. Any source that can be caused by multiple events are also maskable.

When the IPIC receives an internal or external interrupt, its configuration register is checked to determine if it should be routed off-chip (to the external PCI_INTA) or serviced as a normal external interrupt by the processor core (through the \overline{int} signal). As a third alternative, if the incoming interrupt has been configured as a critical or system management interrupt, the IPIC completes the processing of the interrupt by asserting \overline{cint} or \overline{smi} to the core. The assertion of the \overline{cint} or \overline{smi} signal to the core causes the interrupt to be serviced as a critical or a system management interrupt, respectively.

8.2 IPIC Features

The IPIC unit implements the following features:

- Functional and programming compatibility with the MPC8260 interrupt controller
- Support for external and internal discrete vectorized interrupt sources
- Support for external and internal non-maskable machine check conditions, signaled by \overline{mcp}
- Support for dedicated external interrupts—QUICC Engine ports interrupts
- Programmable highest priority request (can be programmed to support a critical (\overline{cint}) or system management interrupt (\overline{smi}) type)
- Two programmable priority mixed groups of four on-chip and four external interrupt signals with two priority schemes for each group: grouped and spread
- Two programmable priority internal groups of eight on-chip interrupt signals with two priority schemes for each group: grouped and spread
- Two highest priority interrupts from each group can be programmed to support a critical or system management interrupt type
- External and internal interrupts directed to host processor
- Unique vector number for each interrupt source

8.3 IIPC Modes of Operation

The IPIC unit can operate in the core enable or core disable mode.

8.3.1 Core Enable Mode

In core enable mode, all internal interrupts (including those from the PCI block) are routed to and from the IPIC; the interrupts are sent to the PowerPC core. The DMA controller can optionally (depending on the programming of the DMA registers) steer its interrupt to the PCI host through the PCI_INTA signal.

In this mode all machine check interrupts are gathered by the IPIC unit and sent to the PowerPC core. If the device performs as a PCI host, the interrupts of the other PCI agents should be connected to the implementation's \overline{IRQx} signals and treated like normal external interrupts (sent to the core).

8.3.2 Core Disable Mode

In core disable mode, all internal interrupts (including those from the PCI block) are routed to and from the IPIC, the interrupts are then sent through the $\overline{\text{PCI_INTA}}$ signal to the PCI host CPU. Note that the core interrupt signal is masked. The user should use in this mode only the $\overline{\text{int}}$ output interrupt type (should not use $\overline{\text{cint}}$ or $\overline{\text{smi}}$ output interrupt types) to read an updated SIVCR. (See Section 8.5.7, “System Internal Interrupt Control Register (SICNR),” and Section 8.5.12, “System External Interrupt Control Register (SECNR).”)

In this mode, machine check interrupts are driven either on $\overline{\text{PCI_INTA}}$ or on $\overline{\text{MCP_OUT}}$ as level-sensitive interrupts. SERCR[MCPR] (see Section 8.5.15, “System Error Control Register (SERCR)”) controls which external signal is used.

8.4 IPIC External Signal Description

The following sections provide an overview and detailed descriptions of the IPIC signals.

8.4.1 IPIC External Signals Overview

The device has 8 distinct external interrupt request input signals ($\overline{\text{IRQ}}[0:7]$) and one interrupt request output signal ($\overline{\text{PCI_INTA}}$). The IPIC interface signals are defined in Table 8-1.

Table 8-1. IPIC Signal Properties

Name	Port	Function	I/O	Reset	Requires Pull Up
$\overline{\text{IRQ}}[0:7]$	$\overline{\text{IRQ}}[0:7]$	External interrupts	I	—	Yes
$\overline{\text{PCI_INTA}}$	$\overline{\text{PCI_INTA}}$	Interrupt request output	O	Z	Yes
$\overline{\text{MCP_OUT}}$	$\overline{\text{MCP_OUT}}$	Interrupt request output	O	Z	Yes

8.4.2 IPIC Detailed Signal Descriptions

Table 8-2 provides detailed descriptions of the external IPIC signals.

Table 8-2. IPIC External Signals—Detailed Signal Descriptions

Signal	I/O	Description
$\overline{\text{IRQ}}[0:7]$	I	Interrupt request 0–7. The sense (level or edge) of each of these signals is programmable. All of these inputs can be driven completely asynchronously.
		State Meaning Asserted—When an external interrupt request signal is asserted the priority is checked by the IPIC unit, and the interrupt is conditionally passed to the processor. Negated—There is no incoming interrupt from that source.
		Timing Assertion—All of these inputs can be asserted completely asynchronously. Negation—Interrupts programmed as level-sensitive must remain asserted until serviced.

Table 8-2. IPIC External Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
$\overline{\text{PCI_INTA}}$	OD	Interrupt request out. Active-low, open drain. When the IPIC is programmed in core disable mode, this output reflects the raw interrupts generated by on-chip sources. See Section 8.3, “IPIPC Modes of Operation,” for details.
		State Meaning Asserted—At least one interrupt is currently being signalled to the external system. Negated—Indicates no interrupt source currently routed to $\overline{\text{IRQ_OUT}}$.
		Timing Because external interrupts are asynchronous with respect to the system clock, both assertion and negation of $\overline{\text{IRQ_OUT}}$ occur asynchronously with respect to the interrupt source. All timing given here is approximate. Assertion—Internal interrupt source: 3 system bus clock cycles after the interrupt occurs. External interrupt source: 4 cycles after the interrupt occurs. Negation—Follows interrupt source negation with the following delay: Internal interrupt: 3 system bus clock cycles. External interrupt: 4 cycles.
$\overline{\text{MCP_OUT}}$	OD	Non-maskable Interrupt (machine check) request out. Active-low, open drain. When the IPIC is programmed in core disable mode, this output reflects the <i>mcp</i> interrupts generated by on-chip sources. See Section 8.3, “IPIPC Modes of Operation.”
		State Meaning Asserted—At least one machine check interrupt is currently being signalled to the external system. Negated—Indicates no interrupt source currently routed to $\overline{\text{MCP_OUT}}$.
		Timing Because external interrupts are asynchronous with respect to the system clock, both assertion and negation of $\overline{\text{MCP_OUT}}$ occurs asynchronously with respect to the interrupt source. All timing given here is approximate. Assertion—Internal interrupt source: 2 system bus clock cycles after interrupt occurs. External interrupt source: 4 cycles after interrupt occurs. Negation—Follows interrupt source negation with the following delay: Internal interrupt: 2 system bus clock cycles. External interrupt: 4 cycles.

8.5 IPIC Memory Map/Register Definition

The IPIC programmable register map occupies 256 bytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All IPIC registers are 32 bits wide and they are located on 32-bit address boundaries. Software can perform byte, half-word or word accesses to any IPIC registers. All addresses used in this chapter are offsets from the IPIC base, as defined in [Chapter 2, “Memory Map.”](#)

[Table 8-3](#) shows the memory map of the IPIC unit. A special set of registers are the QUICC Engine Ports Interrupts registers. These registers have a different base address in the global memory map, and they are detailed in [Table 8-4](#).

Table 8-3. IPIC Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00	System global interrupt configuration register (SICFR)	R/W	All zeros	8.5.1/8-8
0x04	System regular interrupt vector register (SIVCR)	R	All zeros	8.5.2/8-9
0x08	System internal interrupt pending register (SIPNR_H)	R	All zeros	8.5.3/8-11

Table 8-3. IPIC Register Address Map (continued)

Offset	Register	Access	Reset Value	Section/ Page
0x0C	System internal interrupt pending register (SIPNR_L)	R	All zeros	8.5.3/8-11
0x10	System internal interrupt group A priority register (SIPRR_A)	R/W	0x0530_9770	8.5.4/8-13
0x14	Reserved	—	—	—
0x18	Reserved	—	—	—
0x1C	System internal interrupt group D priority register (SIPRR_D)	R/W	0x0530_9770	8.5.5/8-14
0x20	System internal interrupt mask register (SIMSR_H)	R/W	All zeros	8.5.6/8-15
0x24	System internal interrupt mask register (SIMSR_L)	R/W	All zeros	8.5.6/8-15
0x28	System internal interrupt control register (SICNR)	R/W	All zeros	8.5.7/8-16
0x2C	System external interrupt pending register (SEPNR)	R/W	Special	8.5.8/8-18
0x30	System mixed interrupt group A priority register (SMPRR_A)	R/W	0x0530_9770	8.5.9/8-18
0x34	System mixed interrupt group B priority register (SMPRR_B)	R/W	0x0530_9770	8.5.10/8-19
0x38	System external interrupt mask register (SEMSR)	R/W	All zeros	8.5.11/8-20
0x3C	System external interrupt control register (SECNR)	R/W	All zeros	8.5.12/8-21
0x40	System error status register (SERSR)	R/W	All zeros	8.5.13/8-22
0x44	System error mask register (SERMR)	R/W	0xFF00_0000	8.5.14/8-23
0x48	System error control register (SERCR)	R/W	All zeros	8.5.15/8-24
0x4C–0x4F	Reserved	—	—	—
0x50	System internal interrupt force register (SIFCR_H)	R/W	All zeros	8.5.16/8-25
0x54	System internal interrupt force register (SIFCR_L)	R/W	All zeros	8.5.16/8-25
0x58	System external interrupt force register (SEFCR)	R/W	All zeros	8.5.17/8-26
0x5C	System error force register (SERFR)	R/W	All zeros	8.5.18/8-26
0x60	System critical interrupt vector register (SCVCR)	R	All zeros	8.5.19/8-27
0x64	System management interrupt vector register (SMVCR)	R	All zeros	8.5.20/8-27
0x68–0xFF	Reserved	—	—	—

Table 8-4. QUICC Engine Ports Interrupts Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x0C	QUICC Engine ports interrupt event register (CEPIER)	w1c	Special	8.5.21/8-28
0x10	QUICC Engine ports interrupt mask register (CEPIMR)	R/W	All zeros	8.5.22/8-29
0x14	QUICC Engine ports interrupt control register (CEPICR)	R/W	All zeros	8.5.23/8-30

8.5.1 System Global Interrupt Configuration Register (SICFR)

SICFR, shown in [Figure 8-2](#), defines the highest priority interrupt and whether interrupts are grouped or spread in the priority table. See [Table 8-5](#) for more information.



Figure 8-2. System Global Interrupt Configuration Register (SICFR)

[Table 8-5](#) defines the bit fields of SICFR.

Table 8-5. SICFR Field Descriptions

Bits	Name	Description
0	—	Write ignored, read = 0
1–7	HPI	Highest priority interrupt. Specifies the 7-bit unique interrupt number/vector (see Table 8-7) of the single interrupt controller interrupt source that is advanced to the highest priority in the IPIC priority table (see Table 8-35). HPI can be modified dynamically.
8	—	Write ignored, read = 0
9	MPSB	Mixed interrupts priority scheme for group B. Selects the relative MIXB priority scheme. It cannot be changed dynamically. 0 Grouped. The MIXBs are grouped by priority at the top of the table. 1 Spread. The MIXBs are spread by priority in the table.
10	MPSA	Mixed interrupts priority scheme for group A. Selects the relative MIXA priority scheme. It cannot be changed dynamically. 0 Grouped. The MIXAs are grouped by priority at the top of the table. 1 Spread. The MIXAs are spread by priority in the table.
11	—	Write ignored, read = 0
12	IPSD	Internal interrupts priority scheme for group D. Selects the relative SYSD priority scheme. It cannot be changed dynamically. 0 Grouped. The SYSDs are grouped by priority at the top of the table. 1 Spread. The SYSDs are spread by priority in the table.
13–14	—	Write ignored, read = 0
15	IPSA	Internal interrupts priority scheme for group A. Selects the relative SYSA priority scheme. It cannot be changed dynamically. 0 Grouped. The SYSAs are grouped by priority at the top of the table. 1 Spread. The SYSAs are spread by priority in the table.
16–21	—	Write ignored, read = 0

Table 8-7 shows the definition of IVEC.

Table 8-7. IVEC/CVEC/MVEC Field Definition

Interrupt ID Number	Interrupt Meaning	Interrupt Vector
0	Error (no interrupt)	0b000_0000
1–8	Reserved	0b000_0001–0b000_1000
9	UART1	0b000_1001
10	UART2	0b000_1010
11	SEC	0b000_1011
12–13	Reserved	0b000_1100–0b000_1101
14	I2C1	0b000_1110
15	I2C2	0b000_1111
16	Reserved	0b001_0000
17	IRQ1	0b001_0001
18	IRQ2	0b001_0010
19	IRQ3	0b001_0011
20	IRQ4	0b001_0100
21	IRQ5	0b001_0101
22	IRQ6	0b001_0110
23	IRQ7	0b001_0111
24–31	Reserved	0b001_1000–0b001_1111
32	QUICC Engine High	0b010_0000
33	QUICC Engine Low	0b010_0001
34–47	Reserved	0b010_0010–0b010_1111
48	IRQ0	0b011_0000
49–63	Reserved	0b011_0001–0b011_1111
64	RTC SEC	0b100_0000
65	PIT	0b100_0001
66	PCI	0b100_0010
67	Reserved	0b100_0011
68	RTC ALR	0b100_0100
69	MU	0b100_0101
70	SBA	0b100_0110
71	DMA	0b100_0111
72	GTM4	0b100_1000
73	GTM8	0b100_1001
74	QUICC Engine Ports	0b100_1010
75	SDDR	0b100_1011

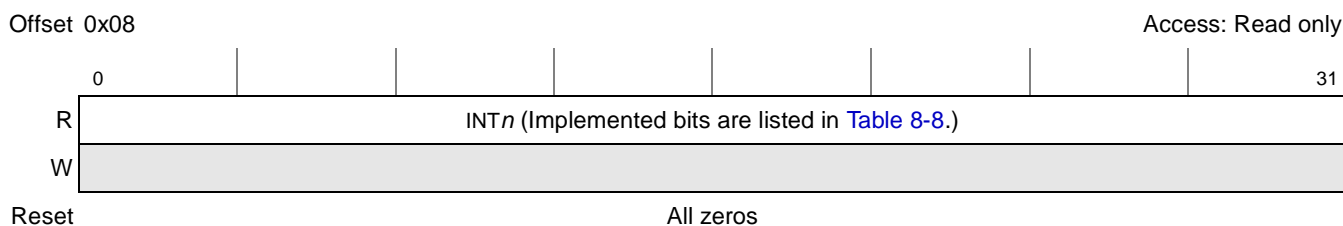
Table 8-7. IVEC/CVEC/MVEC Field Definition (continued)

Interrupt ID Number	Interrupt Meaning	Interrupt Vector
76	DDR	0b100_1100
77	LBC	0b100_1101
78	GTM2	0b100_1110
79	GTM6	0b100_1111
80	PMC	0b101_0000
81–83	Reserved	0b101_0001–0b101_0011
84	GTM3	0b101_0100
85	GTM7	0b101_0101
86–89	Reserved	0b101_0110–0b101_1001
90	GTM1	0b101_1010
91	GTM5	0b101_1011
92–127	Reserved	0b101_1100–0b111_1111

8.5.3 System Internal Interrupt Pending Registers (SIPNR_H and SIPNR_L)

Each bit in SIPNR_H and SIPNR_L, shown in [Figure 8-4](#) and [Figure 8-5](#), may be assigned an internal interrupt source. (Implemented bits are listed in [Table 8-8](#).) When an interrupt request is received, the interrupt controller sets the corresponding SIPNR bit. When a pending interrupt is handled, the user clears the SIPNR bit by clearing the corresponding event register bit.

Note that SIPNR bit positions are not changed according to relative priority.


Figure 8-4. System Internal Interrupt Pending Register (SIPNR_H)

[Table 8-8](#) lists implemented SIPNR_H fields. Note that these field descriptions are also valid for SIFCR_H and SIMSR_H.

Table 8-8. SIPNR_H/SIFCR_H/SIMSR_H Bit Assignments

Bits	Field
0	QE High
1	QE Low
2	—
3	—

Table 8-8. SIPNR_H/SIFCR_H/SIMSR_H Bit Assignments (continued)

Bits	Field
4	—
5	—
6	—
7	—
8–23	—
24	UART1
25	UART2
26	SEC
27–28	—
29	I2C1
30	I2C2
31	—

Table 8-9 defines the bit fields of SIPNR_H.

Table 8-9. SIPNR_H Field Descriptions

Bits	Name	Description
0–31	INT _n	Each implemented bit (listed in Table 8-8) corresponds to an internal interrupt source. When an interrupt is received, the interrupt controller sets the corresponding SIPNR bit. When a pending interrupt is handled, the user clears the SIPNR bit by clearing the corresponding event register bit. SIPNR bits are read only. Writing to this register has no effect. Note that the SIPNR bit positions are not changed according to their relative priority. For unimplemented bits, writes are ignored, read = 0.

SIPNR_L is shown in Figure 8-4.

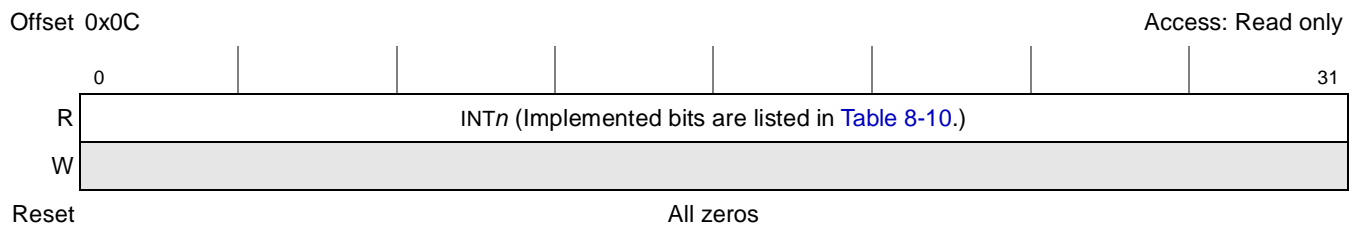


Figure 8-5. System Internal Interrupt Pending Register (SIPNR_L)

Table 8-10 lists implemented SIPNR_L fields. Note that these field assignments are also valid for SIFCR_L and SIMSR_L.

Table 8-10. SIPNR_L/SIFCR_L/SIMSR_L Bit Assignments

Bits	Field
0	RTC SEC
1	PIT
2	PCI
3	—

Table 8-10. SIPNR_L/SIFCR_L/SIMSR_L Bit Assignments (continued)

Bits	Field
4	RTC ALR
5	MU
6	SBA
7	DMA
8	GTM4
9	GTM8
10	QE Ports
11	SDDR
12	DDR
13	LBC
14	GTM2
15	GTM6
16	PMC
17–19	—
20	GTM3
21	GTM7
22–25	—
26	GTM1
27	GTM5
28–30	—
31	—

Table 8-11 defines the bit fields of SIPNR_L.

Table 8-11. SIPNR_L Field Descriptions

Bits	Name	Description
0–31	INT n	Each implemented bit (listed in Table 8-10) corresponds to an internal interrupt source. When an interrupt is received, the interrupt controller sets the corresponding SIPNR bit. When a pending interrupt is handled, the user clears the SIPNR bit by clearing the corresponding event register bit. SIPNR bits are read only. Writing to this register has no effect. Note that the SIPNR bit positions are not changed according to their relative priority. For unimplemented bits, writes are ignored, read = 0.

8.5.4 System Internal Interrupt Group A Priority Register (SIPRR_A)

The system internal interrupt group A priority register (SIPRR_A), shown in [Figure 8-6](#), defines the priority between QE High and QE Low internal interrupt signals.

Offset 0x10

Access: Read/write

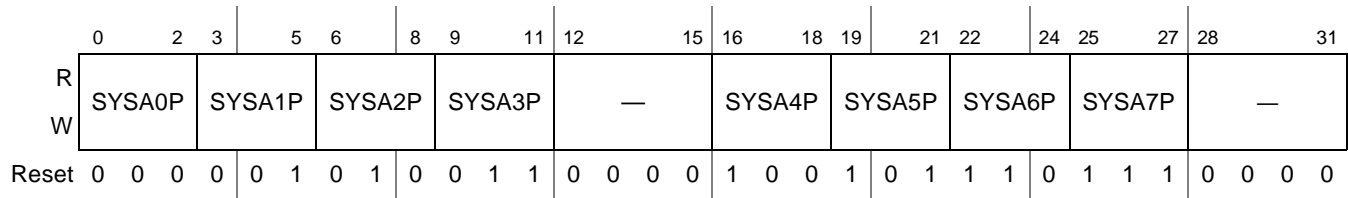


Figure 8-6. System Internal Interrupt Group A Priority Register (SIPRR_A)

Table 8-12 defines the bit fields of SIPRR_A.

Table 8-12. SIPRR_A Field Descriptions

Bits	Name	Description
0–2	SYSA0P	SYSA0 priority order. Defines which interrupt source asserts its request in the SYSA0 priority position. The user should not program the same code to multiple priority positions (0–7). These bits can be changed dynamically. The definition of SYSA0P is as follows: 000 QE High asserts its request in the SYSA0 position. 001 QE Low asserts its request in the SYSA0 position. 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
3–11, 16–27	SYSA1P–SYSA7P	Same as SYSA0P, but for SYSA1P–SYSA7P.
12–15, 28–31	—	Write ignored, read = 0

8.5.5 System Internal Interrupt Group D Priority Register (SIPRR_D)

SIPRR_D, shown in Figure 8-7, defines the priority among the interrupt sources listed in Table 8-13.

Offset 0x1C

Access: Read/write

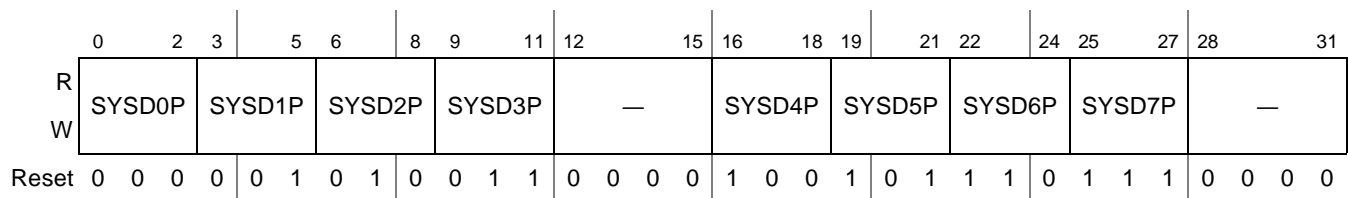


Figure 8-7. System Internal Interrupt Group D Priority Register (SIPRR_D)

Table 8-14 defines the bit fields of SIMSR_H.

Table 8-14. SIMSR_H Field Descriptions

Bits	Name	Description
0–31	INT _n	<p>Each implemented bit (listed in Table 8-8) corresponds to an external interrupt source. The user masks an interrupt by clearing the corresponding SIMSR bit. An interrupt is unmasked (enable) by setting the corresponding SIMSR bit. The SIMSR can be read by the user at any time.</p> <p>Note:</p> <ul style="list-style-type: none"> • SIMSR bit positions do not change according to their relative priority. • The user can clear pending register bits that were set by multiple interrupt events only by clearing all unmasked events in the corresponding event register. • If an SIMSR bit is masked at the same time that the corresponding SIPNR bit causes an interrupt request to the core, the error vector is issued (if no other interrupts are pending). Thus, the user should always include an error vector routine, even if it contains only an rfi instruction. The error vector cannot be masked. <p>Unimplemented bits, shown as reserved in Table 8-8, are ignored on writes; read = 0.</p>

Figure 8-9 shows SIMSR_L.



Figure 8-9. System Internal Interrupt Mask Register (SIMSR_L)

Table 8-15 defines the bit fields of SIMSR_L.

Table 8-15. SIMSR_L Field Descriptions

Bits	Name	Description
0–31	INT _n	<p>Each implemented bit (listed in Table 8-10) corresponds to an external interrupt source. The user masks an interrupt by clearing the corresponding SIMSR bit. An interrupt is unmasked (enabled) by setting the corresponding SIMSR bit. The SIMSR can be read by the user at any time.</p> <p>Note:</p> <ul style="list-style-type: none"> • SIMSR bit positions are not changed according to their relative priority. • The user can clear pending register bits that were set by multiple interrupt events only by clearing all unmasked events in the corresponding event register. • If an SIMSR bit is masked at the same time that the corresponding SIPNR bit causes an interrupt request to the core, the error vector is issued (if no other interrupts are pending). Thus, the user should always include an error <p>Unimplemented bits, shown as reserved in Figure 8-9, are ignored on writes; read = 0.</p>

8.5.7 System Internal Interrupt Control Register (SICNR)

SICNR, shown in Figure 8-10, defines the IPIC output interrupt type ($\overline{\text{int}}$, $\overline{\text{cint}}$, or $\overline{\text{smi}}$) in the SYSA0–SYSA1 and SYSD0–SYSD1 priority positions. All other priority positions assert $\overline{\text{int}}$ to the core.

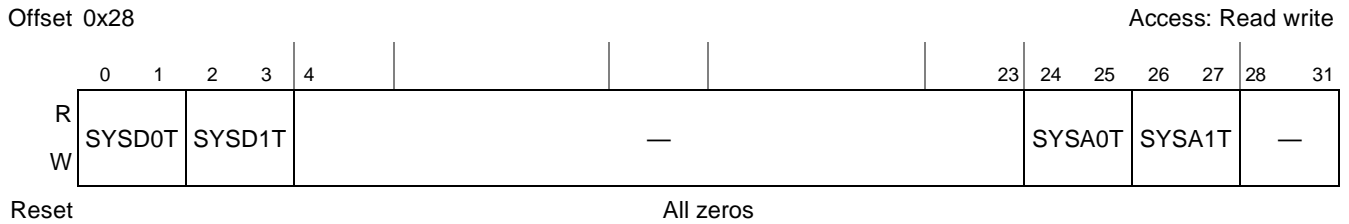


Figure 8-10. System Internal Interrupt Control Register (SICNR)

Table 8-16 defines the bit fields of SICNR.

Table 8-16. SICNR Field Descriptions

Bits	Name	Description
0–1	SYSD0T	<p>SYSD0 priority position IPIC output interrupt type. Defines which type of the IPIC output interrupt signal (\overline{int}, \overline{cint}, or \overline{smi}) asserts its request to the core in the SYSD0 priority position. These bits cannot be changed dynamically. (to change it, software must make sure the corresponding interrupt source is masked or it cannot happen during the change).</p> <p>The definition of SYSD0T is as follows:</p> <ul style="list-style-type: none"> 00 \overline{int} request is asserted to the core for SYSD0. 01 \overline{smi} request is asserted to the core for SYSD0. 10 \overline{cint} request is asserted to the core for SYSD0. 11 Reserved
2–3	SYSD1T	Same as SYSD0T, but for SYSD1T.
4–23	—	Write ignored, read = 0
24–25	SYSA0T	<p>SYSA0 priority position IPIC output interrupt type. Defines which type of the IPIC output interrupt signal (\overline{int}, \overline{smi}, or \overline{cint}) asserts its request to the core in the SYSA0 priority position. These bits can not be changed dynamically. (If s/w really wants to change it, it must ensure the corresponding interrupt source is masked or it does not happen during the change).</p> <p>The definition of SYSA0T is as follows:</p> <ul style="list-style-type: none"> 00 \overline{int} request is asserted to the core for SYSA0. 01 \overline{smi} request is asserted to the core for SYSA0. 10 \overline{cint} request is asserted to the core for SYSA0. 11 Reserved.
26–27	SYSA1T	Same as SYSA0T, but for SYSA1T
28–31	—	Write ignored, read = 0

Table 8-18 defines the bit fields of SMPRR_A.

Table 8-18. SMPRR_A Field Descriptions

Bits	Name	Description
0–2	MIXA0P	MIXA0 priority order. Defines which interrupt source asserts its request in the MIXA0 priority position. The user must not program the same code to more than one priority position (0–7). These bits can be changed dynamically. The definition of MIXA0P is as follows: 000 RTC SEC asserts its request to the MIXA0 position. 001 PIT asserts its request to the MIXA0 position. 010 PCI asserts its request to the MIXA0 position. 011 Reserved. 100 IRQ0 asserts its request to the MIXA0 position. This field for MIXA0 position is valid (must not be ignored) if IRQ0 signal configured as an external maskable interrupt (SEMSR[SIRQ0] = 0). 101 IRQ1 asserts its request to the MIXA0 position. 110 IRQ2 asserts its request to the MIXA0 position. 111 IRQ3 asserts its request to the MIXA0 position.
3–11, 16–27	MIXA1P– MIXA7P	Same as MIXA0P, but for MIXA1P–MIXA7P.
12–15, 28–31	—	Write ignored, read = 0

8.5.10 System Mixed Interrupt Group B Priority Register (SMPRR_B)

SMPRR_B, shown in Figure 8-13, defines the priority among the sources listed in Table 8-19.

Offset 0x34

Access: Read/write

	0	2	3	5	6	8	9	11	12	15	16	18	19	21	22	24	25	27	28	31
R	MIXB0P	MIXB1P	MIXB2P	MIXB3P	—	MIXB4P	MIXB5P	MIXB6P	MIXB7P	—										
W																				
Reset	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	1	0	1	1

Figure 8-13. System Mixed Interrupt Group B Priority Register (SMPRR_B)

Table 8-19 defines the bit fields of SMPRR_B.

Table 8-19. SMPRR_B Field Descriptions

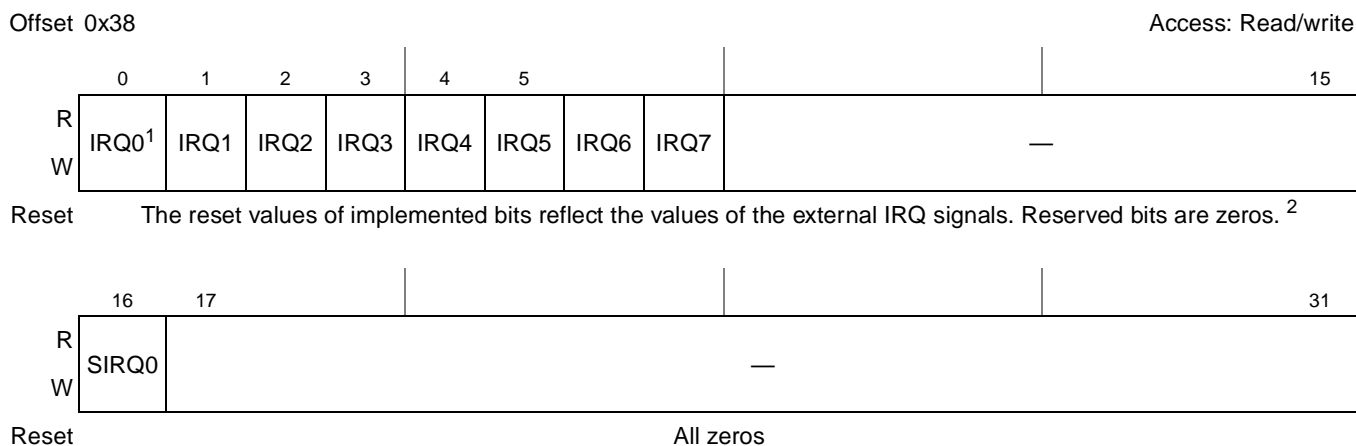
Bits	Name	Description
0–2 3–11, 16–27	MIXB _n P	MIXB _n priority order. Defines which interrupt source asserts its request in the MIXB _n priority position. The user must not program the same code to more than one priority position (0–7). These bits can be changed dynamically. The definition of MIXB _n P is as follows: 000 RTC ALR asserts its request to the MIXB _n position. 001 MU asserts its request to the MIXB _n position. 010 SBA asserts its request to the MIXB _n position. 011 DMA asserts its request to the MIXB _n position. 100 IRQ4 asserts its request to the MIXB _n position. 101 IRQ5 asserts its request to the MIXB _n position. 110 IRQ6 asserts its request to the MIXB _n position. 111 IRQ7 asserts its request to the MIXB _n position.
12–15, 28–31	—	Write ignored, read = 0

8.5.11 System External Interrupt Mask Register (SEMSR)

Each bit in the system external interrupt mask register (SEMSR), shown in Figure 8-11, corresponds to an external interrupt source. The user masks an interrupt by clearing the corresponding SEMSR bit. An interrupt is unmasked (enabled) by setting the corresponding SEMSR bit.

When an external interrupt request occurs, the corresponding SEP_{NR} bit is set regardless of the setting of the corresponding SEMSR bit. However, if the corresponding SEMSR bit is cleared, no interrupt request is passed to the core.

When an SEMSR bit is cleared by the user at the same time that an interrupt source requests an interrupt service, the request stops. If the user sets the SEMSR bit later, a previously pending interrupt request is processed by the core according to its assigned priority. SEMSR can be read by the user at any time.



¹ This bit is valid only if the IRQ0 signal is configured as an external maskable interrupt (SEMSR[SIRQ0] = 0)

² The user should drive all IRQ inputs to an inactive state prior to reset negation

Figure 8-14. System External Interrupt Mask Register (SEMSR)

Table 8-20 defines the bit fields of SEMSR.

Table 8-20. SEMSR Field Descriptions

Bits	Name	Description
0–7	—	Each bit corresponds to an external interrupt source. The user masks an interrupt by clearing the SEMSR bit. An interrupt can be enabled by setting the corresponding SEMSR bit. SEMSR can be read by the user at any time. Note: <ul style="list-style-type: none"> SEMSR bit positions are not affected by their relative priority. The user can clear pending register bits that were set by multiple interrupt events only by clearing all unmasked events in the corresponding event register. If an SEMSR bit is masked at the same time that the corresponding SEPNR bit causes an interrupt request to the core, the error vector is issued (if no other interrupts pending). Thus, the user must always include an error vector routine, even if it contains only an <code>rfi</code> instruction. The error vector cannot be masked.
8–15	—	Write ignored, read = 0
16	SIRQ0	Steer IRQ0. 0 IRQ0 is used as external interrupt request 1 IRQ0 is used as external MCP request
17–31	—	Write ignored, read = 0

8.5.12 System External Interrupt Control Register (SECNR)

SECNR, shown in Figure 8-15, defines the edge detect mode for external \overline{IRQ}_n interrupt signals and determines whether the corresponding \overline{IRQ}_n signal asserts an interrupt request upon either a high-to-low change or assertion on the pin. It also defines the IPIC output interrupt type (\overline{int} , \overline{cint} , or \overline{smi}) in the MIXA0–MIXA1 and MIXB0–MIXB1 priority positions.

Note that in core disabled mode of operation the user should use the \overline{int} output interrupt type (should not use \overline{cint} or \overline{smi} output interrupt types) in order to read an updated SIVCR.

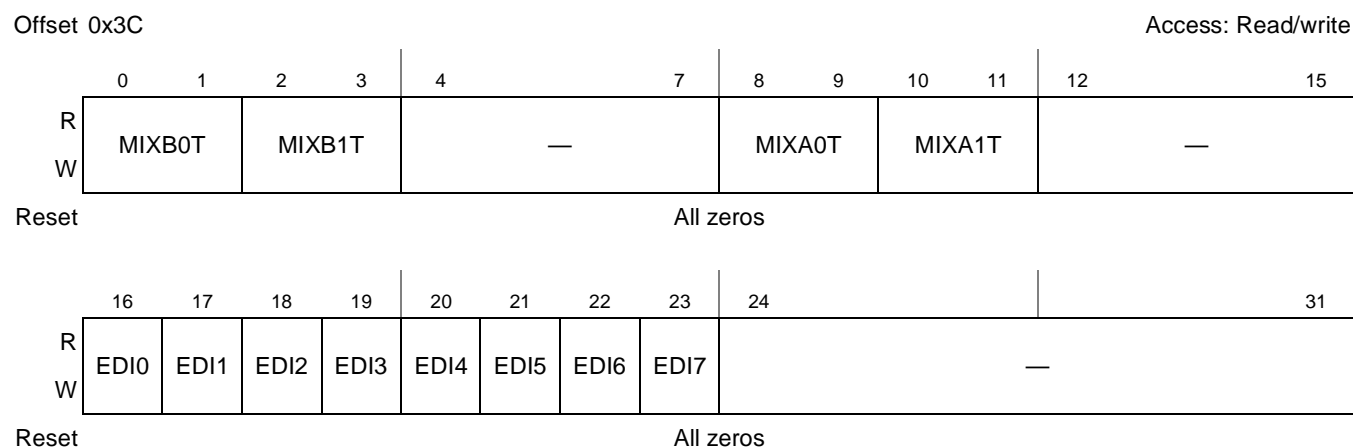


Figure 8-15. System External Interrupt Control Register (SECNR)

Table 8-21 defines the bit fields of SECNR.

Table 8-21. SECNR Field Descriptions

Bits	Name	Description
0–1	MIXB0T	MIXB0 priority position IPIC output interrupt type. Defines which type of the IPIC output interrupt signal (\overline{int} , \overline{cint} , or \overline{smi}) asserts its request to the core in the MIXB0 priority position. These bits can be changed dynamically. The definition of MIXB0T is as follows: 00 \overline{int} request is asserted to the core for MIXB0. 01 \overline{smi} request is asserted to the core for MIXB0. 10 \overline{cint} request is asserted to the core for MIXB0. 11 Reserved
2–3	MIXB1T	Same as MIXB0T, but for MIXB1T.
4–7	—	Write ignored, read = 0
8–9	MIXA0T	MIXA0 priority position IPIC output interrupt Type. Defines which type of the IPIC output interrupt signal (\overline{int} , \overline{cint} , or \overline{smi}) asserts its request to the core in the MIXA0 priority position. These bits can be changed dynamically. The definition of MIXA0T is as follows: 00 \overline{int} request is asserted to the core for MIXA0. 01 \overline{smi} request is asserted to the core for MIXA0. 10 \overline{cint} request is asserted to the core for MIXA0. 11 Reserved
10–11	MIXA1T	Same as MIXA0T, but for MIXA1T.
12–15	—	Write ignored, read = 0
16–23	EDIx	Each bit defines the edge detect mode for the external \overline{IRQn} interrupt signals, determines whether the corresponding \overline{IRQn} signal asserts an interrupt request upon either a high-to-low change or low assertion on the pin. The corresponding \overline{IRQn} signal asserts an interrupt request as follows: 0 Low assertion on \overline{IRQn} generates an interrupt request (level sensitive). 1 High-to-low change on \overline{IRQn} generates an interrupt request (edge sensitive).
24–31	—	Write ignored, read = 0

8.5.13 System Error Status Register (SERSR)

The bits in the SERSR, shown in Figure 8-16, correspond to the external and internal non-maskable error source machine check (mcp) conditions listed in Table 8-22. When an error interrupt signal is received, the interrupt controller sets the corresponding SERSR bit.



Figure 8-16. System Error Status Register (SERSR)

Table 8-22 lists the implemented SERSR bits. Note that these field assignments are valid for SERMR and SERFR.

Table 8-22. SERSR/SERMR/SERFR Bit Assignments

Bits	Field
0	IRQ0 ¹
1	WDT
2	SBA
3	CIEE
4	CMEE
5	PCI
6	—
7	MU
8–14	—
15	—
16–31	—

¹ This bit is valid only if the IRQ0 signal is configured as an external MCP interrupt (SEMSR[SIRQ0] = 1)

Table 8-23 defines the bit fields of SERSR.

Table 8-23. SERSR Field Descriptions

Bits	Name	Description
0–31	INT _n	Each implemented bit in the SERSR, listed in Table 8-22, corresponds to an external and an internal error source (mcp). When an error interrupt signal is received, the interrupt controller sets the corresponding SERSR bit. SERSR bits are cleared by writing ones to them. Unmasked event register bits should be cleared before clearing SERSR bits. Because the user can only clear bits in this register, writing zeros to this register has no effect. SERSR bits are cleared by power-on reset. Subsequent soft and hard resets do not affect SERSR bit states. For unimplemented bits (listed as reserved in Table 8-22), writes are ignored, read = 0

8.5.14 System Error Mask Register (SERMR)

Each implemented bit in SERMR, shown in Figure 8-17, corresponds to an external and an internal \overline{mcp} source (MCP). The user masks an MCP by clearing and enables an interrupt by setting the corresponding SERMR bit. When a masked MCP occurs, the corresponding SERSR bit is set, regardless of the setting of the corresponding SERMR bit although no MCP request is passed to the core in this case. The SERMR can be read by the user at any time.

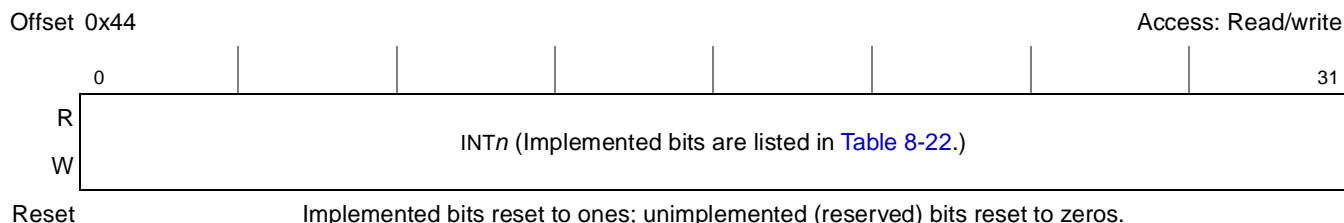


Figure 8-17. System Error Mask Register (SERMR)

Table 8-24 defines the bit fields of SERMR.

Table 8-24. SERMR Field Descriptions

Bits	Name	Description
0–31	INT n	Each implemented SERMR bit, listed in Table 8-22, corresponds to an external and an internal MCP source. The user masks an MCP by clearing and enables an interrupt by setting the corresponding SERMR bit. When a masked MCP occurs, the corresponding SERSR bit is set, regardless of the setting of the SERMR bit although no MCP request is passed to the core. The SERMR can be read by the user at any time. Writes to unimplemented (reserved) bits are ignored; read = 0

8.5.15 System Error Control Register (SERCR)

SERCR, shown in Figure 8-18, defines the control bits that route MCP requests in core disable mode to either MCP_OUT or PCI_INTA in core-disable mode.

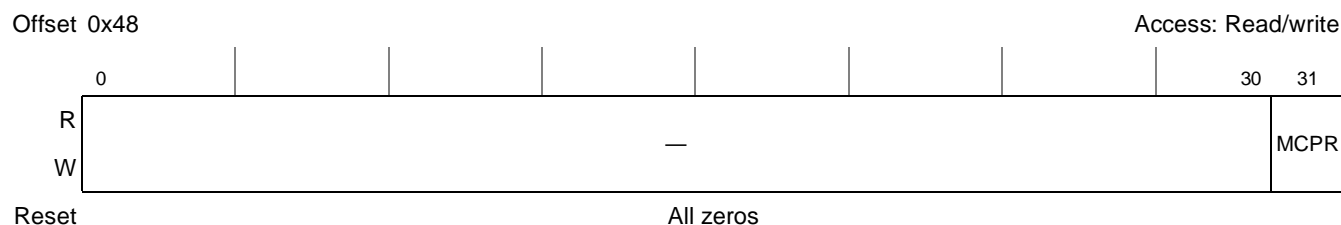


Figure 8-18. System Error Control Register (SERCR)

Table 8-25 defines the bit fields of SERCR.

Table 8-25. SERCR Field Descriptions

Bits	Name	Description
0–30	—	Write ignored, read = 0
31	MCPR	MCP route. Route MCP request to either $\overline{\text{MCP_OUT}}$ or $\overline{\text{PCI_INTA}}$ (in core disable mode). 0 MCP routed to $\overline{\text{PCI_INTA}}$ (in core disable mode). 1 MCP routed to $\overline{\text{MCP_OUT}}$ (in core disable mode).

8.5.16 System Internal Interrupt Force Registers (SIFCR_H and SIFCR_L)

Each implemented bit SIFCR_H and SIFCR_L, shown in [Figure 8-19](#) and [Figure 8-20](#), corresponds to an internal interrupt source. When a bit is set, the interrupt controller generates the corresponding interrupt (sets the corresponding SIPNR bit). The SIFCR can be read by the user at any time.



Figure 8-19. System Internal Interrupt Force Register (SIFCR_H)

[Table 8-26](#) defines the bit fields of SIFCR_H.

Table 8-26. SIFCR_H Field Descriptions

Bits	Name	Description
0-31	INT n	Each implemented bit, listed in Table 8-8 , corresponds to an internal interrupt source. The user forces an interrupt by setting the corresponding SIFCR x bit. SIFCR n bit positions are not changed according to their relative priority. Writes to unimplemented (reserved) bits are ignored; read = 0

SIFCR_L is shown in [Figure 8-20](#).

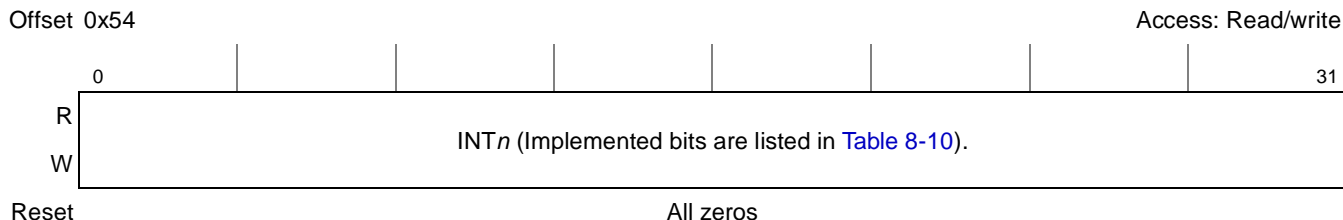


Figure 8-20. System Internal Interrupt Force Register (SIFCR_L)

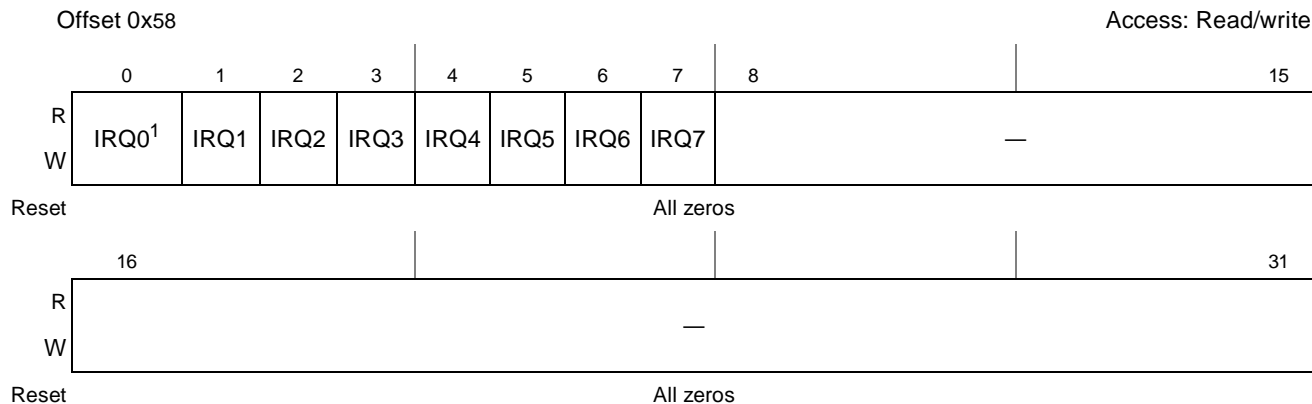
[Table 8-27](#) defines the bit fields of SIFCR_L.

Table 8-27. SIFCR_L Field Descriptions

Bits	Name	Description
0-31	INT n	Each implemented bit, listed in Table 8-10 , corresponds to an internal interrupt source. The user forces an interrupt by setting the corresponding SIFCR x bit. SIFCR x bit positions are not changed according to their relative priority. Writes to unimplemented (reserved) bits are ignored; read = 0

8.5.17 System External Interrupt Force Register (SEFCR)

Each implemented bit in SEFCR, shown in [Figure 8-21](#), corresponds to an external interrupt source. When a bit is set, the interrupt controller generates the corresponding external interrupt (sets the corresponding SEPNR bit). SEFCR can be read by the user at any time.



¹ This bit is valid only if IRQ0 is configured as an external maskable interrupt (SEMSR[SIRQ0] = 0)

Figure 8-21. System External Interrupt Force Register (SEFCR)

[Table 8-28](#) defines the bit fields of SEFCR.

Table 8-28. SEFCR Field Descriptions

Bits	Name	Description
0–7	IRQ _n	Each bit corresponds to an external interrupt source. The user force an interrupt by setting the SEFCR bit. Note: SEFCR bit positions are not affected by their relative priority.
8–31	—	Write ignored, read = 0

8.5.18 System Error Force Register (SERFR)

Each bit in the system error force register (SERFR), shown in [Figure 8-22](#), corresponds to an external MCP source. When a bit is set, the interrupt controller generates the corresponding MCP interrupt (sets the corresponding SERSR bit). The SERFR can be read by the user at any time.



Figure 8-22. System Error Status Register (SERFR)

Table 8-29 defines the bit fields of SERFR.

Table 8-29. SERFR Field Descriptions

Bits	Name	Description
0–31	INT _n	Each implemented bit, listed in Table 8-22, corresponds to an external MCP source. The user forces an MCP by setting the SERFR bit. SERFR bit positions are not affected by their relative priority. Attempts to write to unimplemented (reserved) bits are ignored; read = 0

8.5.19 System Critical Interrupt Vector Register (SCVCR)

SCVCR, shown in Figure 8-23, contains a 7-bit code (Table 8-30) representing the unmasked critical interrupt (CINT) source of the highest priority level.

Note that in core-disabled mode the user should use SIVCR only to read an updated interrupt vector (SCVCR should not be used).

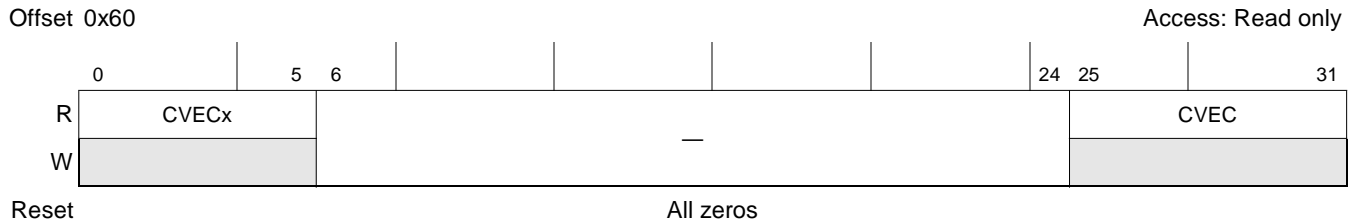


Figure 8-23. System Critical Interrupt Vector Register (SCVCR)

Table 8-30 defines SCVCR bit fields.

Table 8-30. SCVCR Field Descriptions

Bits	Name	Description
0–5	CVECx	Backward (MPC8260) compatible critical interrupt vector. Specifies a 6-bit unique number of the IPIC's highest priority critical interrupt source, pending to the core. When a critical interrupt request occurs, SCVCR can be read. If there are multiple critical interrupt sources, SCVCR latches the highest priority critical interrupt. Note that CVECx field will correctly reflect only first 64 interrupt vectors (See Table 8-7 for details). The value of SCVEC cannot change while it is being read.
6–24	—	Write ignored, read = 0
25–31	CVEC	Critical interrupt vector. Specifies a 7-bit unique number of the IPIC's highest priority critical interrupt source, pending to the core. When a critical interrupt request occurs, SCVCR can be read. If there are multiple critical interrupt sources, SCVCR latches the highest priority critical interrupt. Note that CVEC field correctly reflects all of the interrupt vectors (See Table 8-7 for details). The value of SCVEC cannot change while it is being read.

8.5.20 System Management Interrupt Vector Register (SMVCR)

SMVCR, shown in Figure 8-24, contains a 7-bit code (Table 8-31) representing the unmasked system management interrupt (SMI) source of the highest priority level.

Offset 0x0C¹

Access: w1c

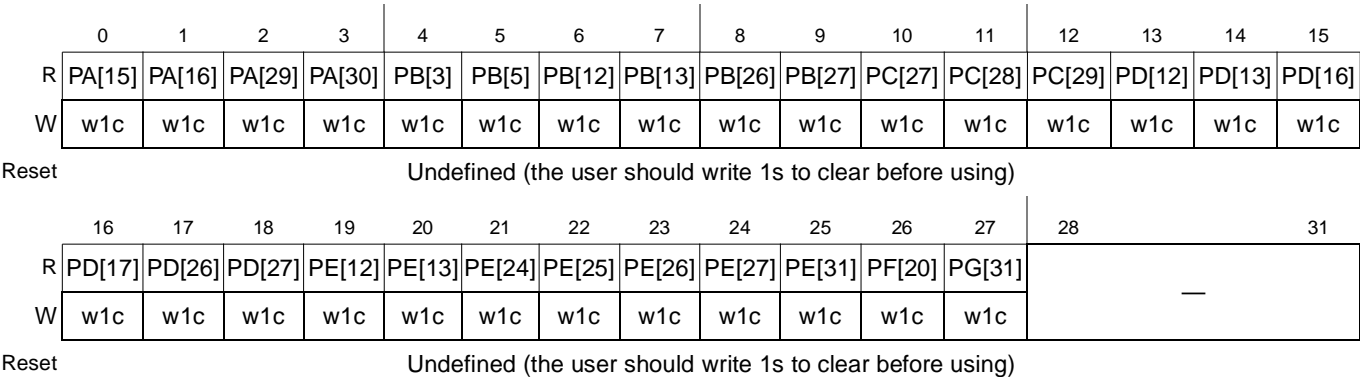


Figure 8-25. QUICC Engine Ports Interrupt Event Register (CEPIER)

¹ Note that the base address for CEPIER is not the same as the base address of other registers in the IPIC. Instead, this register uses a base address which is specific to QUICC engine ports interrupts registers. See [Chapter 2, “Memory Map.”](#)

[Table 8-32](#) defines the bit fields of CEPIER.

Table 8-32. CEPIER Bit Settings

Bits	Name	Description
0–27	Px[n]	QUICC Engine ports interrupt events. Indicates whether interrupt event occurred on the corresponding QUICC Engine port signal. 0 No interrupt event occurred on the corresponding QUICC Engine port signal. 1 Interrupt event occurred on the corresponding QUICC Engine port signal.
28–31	—	Reserved. Should be cleared.

8.5.22 QUICC Engine Ports Interrupt Mask Register (CEPIMR)

The QUICC Engine ports interrupt mask register (CEPIMR), shown in [Figure 8-26](#), defines the interrupt masking for the individual QUICC Engine ports lines. When a masked interrupt request occurs, the corresponding CEPIER bit is set, regardless of the CEPIMR state. When one or more non-masked interrupt events occur, the ‘QE Ports’ internal event is generated. The ‘QE Ports’ internal event is one source in the SIPNR register (See [Section 8.5.3, “System Internal Interrupt Pending Registers \(SIPNR_H and SIPNR_L\).”](#))

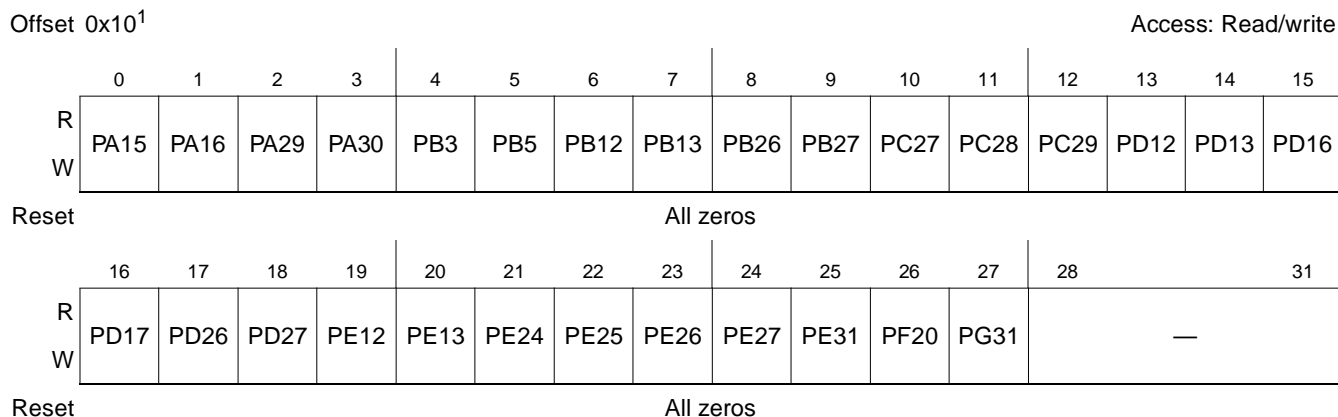


Figure 8-26. QUICC Engine Ports Interrupt Mask Register (CEPIMR)

¹ Note that the base address for CEPIER is not the same as the base address of other registers in the IPIC. Instead, this register uses a base address which is specific to QUICC engine ports interrupts registers. See [Chapter 2, “Memory Map.”](#)

Table 8-33 defines the bit fields of CEPIMR.

Table 8-33. CEPIMR Bit Settings

Bits	Name	Description
0–27	Px[n]	Interrupt mask. Indicates whether an interrupt event is masked or not masked. 0 The input interrupt signal is masked (disabled). 1 The input interrupt signal is not masked (enabled).
28–31	—	Reserved. Should be cleared.

8.5.23 QUICC Engine Ports Interrupt Control Register (CEPICR)

The QUICC Engine ports interrupt control register (CEPICR), shown in [Figure 8-27](#), determines whether the corresponding QUICC Engine port line asserts an interrupt request upon either a high-to-low change or any change on the state of the signal.

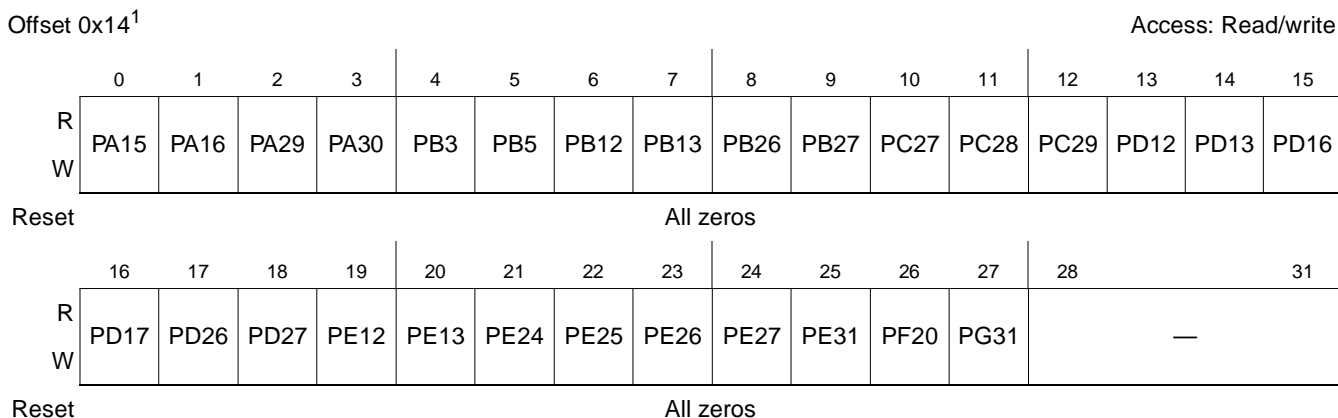


Figure 8-27. QUICC Engine Ports Interrupt Control Register (CEPICR)

¹ Note that the base address for CEPIER is not the same as the base address of other registers in the IPIC. Instead, this register uses a base address which is specific to QUICC engine ports interrupts registers. See [Chapter 2, “Memory Map.”](#)

Table 8-34 defines the bit fields of CEPICR.

Table 8-34. CEPICR Bit Settings

Bits	Name	Description
0–27	Pxn	Edge detection mode. The corresponding QUICC Engine port line asserts an interrupt request according to the following: 0 Any change on the state of the QUICC Engine port generates an interrupt request. 1 High-to-low change on the QUICC Engine port generates an interrupt request.
28–31	—	Reserved. Should be cleared.

8.6 Functional Description

The following sections describe the types of interrupts, interrupt configurations, and their priorities.

8.6.1 Interrupt Types

The IPIC is responsible for receiving hardware-generated interrupts from different sources (both internal and external) along with prioritizing and delivering them to the CPU for servicing. The interrupt sources are controlled by the IPIC unit and may cause three types of exceptions in the processor core. The \overline{int} signal is the main interrupt output from the IPIC to the processor core and causes the external interrupt exception. The \overline{cint} signal is the critical interrupt output from the IPIC to the processor core and causes the critical external interrupt exception. The \overline{smi} signal is the system management interrupt output from the IPIC to the processor core and causes the system management interrupt exception. The machine check exception is caused by the internal \overline{mcp} signal generated by the IPIC, informing the processor of error conditions, assertion of the external MCP request, and other conditions.

8.6.2 Interrupt Configuration

Figure 8-28 shows the interrupt configuration.

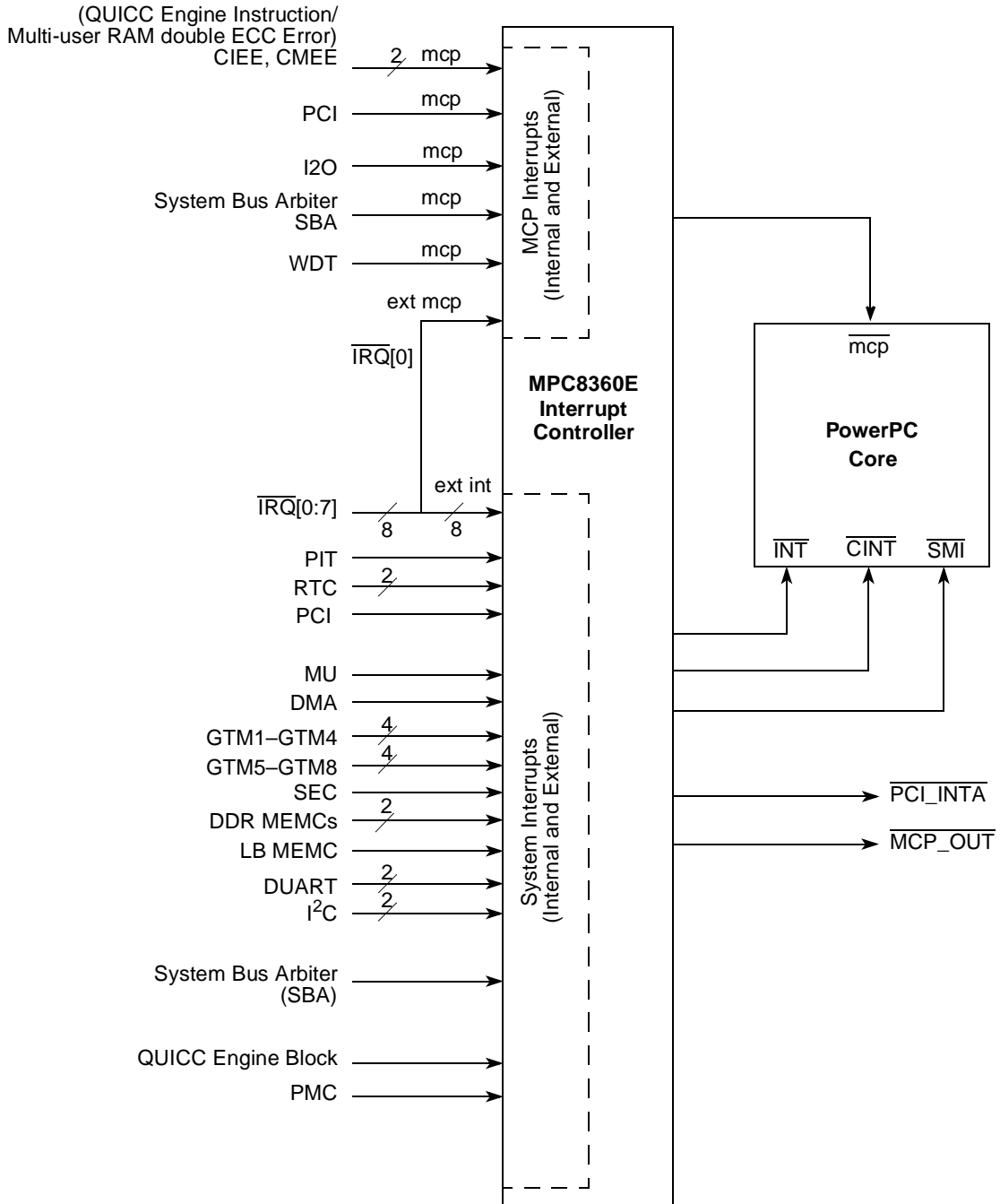


Figure 8-28. Interrupt Structure

The interrupt controller allows masking of each interrupt source. When an unmasked interrupt source is pending in the SIPNR register, the interrupt controller sends an interrupt request to the core. When an interrupt is taken, the interrupt mask bit in the machine state register is cleared to disable further interrupt requests to the PowerPC core until software can handle them.

All interrupt sources are prioritized and bits are set in the system interrupt pending register (SIPNR, SEPNR) as interrupts occur regardless of whether they are masked in the IPIC. The prioritization of the interrupt sources is flexible within the following groups:

- The relative priority of the QE High and QE Low internal interrupt signals can be modified.
- The relative priority of the UART1, UART2, I2C1, I2C2, and SEC internal interrupt signals can be modified.
- The relative priority of the IRQ0, IRQ1, IRQ2, and IRQ3 external interrupts, and RTC SEC internal interrupts can be modified.
- The relative priority of the IRQ4, IRQ5, IRQ6, and IRQ7 external interrupts, and RTC ALR, MU, SBA, and DMA internal interrupts can be modified.
- One interrupt source can be assigned to be the programmable highest priority.

The SIVCR is updated with a 7-bit vector corresponding to the sub-block with the highest current priority.

8.6.3 Internal Interrupts Group Relative Priority

The relative priority in each internal group is programmable and can be changed dynamically. The group priorities are programmed in the IPIC internal interrupt priority registers (SIPRR_x) and can be changed dynamically to implement a rotating priority.

In addition, the grouping of the locations of the interrupt entries has the following two options:

- Grouped. In the group scheme, all interrupts are grouped together at the top of [Table 8-35](#), ahead of most other interrupt sources. This scheme is ideal for applications where all interrupt sources function at a very high data rate and interrupt latency is very important.
- Spread. In the spread scheme, priorities are spread over [Table 8-35](#) so other sources can have lower interrupt latencies. This scheme is also programmed but cannot be changed dynamically.

8.6.4 Mixed Interrupts Group Relative Priority

The relative priority between up to four internal and four external interrupts in each group is programmable and can be changed dynamically. The group priorities are programmed in the IPIC mixed interrupt priority registers (SMPPRR_x) and can be changed dynamically to implement a rotating priority.

In addition, the grouping of the locations of the mixed interrupt entries has the following two options:

- Grouped. In the group scheme, all interrupts are grouped together at the top of the priority table, ahead of most other interrupt sources. See [Table 8-35](#) for more information. This scheme is ideal for applications where all interrupt sources function at a very high data rate and interrupt latency is very important.
- Spread. In the spread scheme, priorities are spread over the table so other sources can have lower interrupt latencies. This scheme is also programmed but cannot be changed dynamically.

8.6.5 Highest Priority Interrupt

In addition to the group relative priority option, SICFR[HPI] can be used to specify one interrupt source as having the highest priority. This interrupt remains within the same interrupt level as the other interrupt controller interrupts, but is serviced before any other interrupt in [Table 8-35](#).

If the highest priority feature is not used, the IPIC selects the interrupt request in MIXA0 to be the highest priority interrupt and the standard interrupt priority order is used from [Table 8-35](#). SICFR[HPI] can be updated dynamically to allow the user to change a normally low priority source into a high priority-source for a period as needed.

8.6.6 Interrupt Source Priorities

Each of the IPIC’s internal and external interrupt sources can independently assert one interrupt request to the core. [Table 8-35](#) shows the prioritization of these interrupt sources. As described in previous sections, flexibility exists in the relative ordering of the interrupts, but, in general, relative priorities are as shown. A single interrupt priority number is associated with each table entry.

Table 8-35. Interrupt Source Priority Levels

Priority	Interrupt Source Description
1	Highest
2	MIXA0 (Grouped/Spread)
3	MIXA1 (Grouped)
4	MIXA2 (Grouped)
5	MIXA3 (Grouped)
6	MIXB0 (Spread)
7–10	Reserved
11	MIXA1 (Spread)
12–15	Reserved
16	MIXB0 (Grouped)
17	MIXB1 (Grouped)
18	MIXB2 (Grouped)
19	MIXB3 (Grouped)
20	MIXB1 (Spread)
21	SYSA0 (Grouped)
22	SYSA1 (Grouped)
23	SYSA2 (Grouped)
24	SYSA3 (Grouped)
25	MIXA2 (Spread)
26	SYSA4 (Grouped)
27	SYSA5 (Grouped)
28	SYSA6 (Grouped)
29	SYSA7 (Grouped)

Table 8-35. Interrupt Source Priority Levels (continued)

Priority	Interrupt Source Description
30	MIXA4 (Grouped)
31	MIXA5 (Grouped)
32	MIXA6 (Grouped)
33	MIXA7 (Grouped)
34	MIXB2 (Spread)
35–38	Reserved
39	MIXA3 (Spread)
40–43	Reserved
44	MIXB4 (Grouped)
45	MIXB5 (Grouped)
46	MIXB6 (Grouped)
47	MIXB7 (Grouped)
48	MIXB3 (Spread)
49	SYSD0 (Grouped)
50	SYSD1 (Grouped)
51	SYSD2 (Grouped)
52	SYSD3 (Grouped)
53	MIXA4 (Spread)
54	SYSD4 (Grouped)
55	SYSD5 (Grouped)
56	SYSD6 (Grouped)
57	SYSD7 (Grouped)
58	MIXB4 (Spread)
59	GTM4
60	Reserved
61	SYSA0 (Spread)
62	GTM8
63	Reserved
64	SYSD0 (Spread)
65	Reserved
66	QE Ports
67	MIXA5 (Spread)
68	SDDR
69	Reserved
70	SYSA1 (Spread)

Table 8-35. Interrupt Source Priority Levels (continued)

Priority	Interrupt Source Description
71	DDR
72	Reserved
73	SYSD1 (Spread)
74	Reserved
75	LBC
76	MIXB5 (Spread)
77	GTM2
78	Reserved
79	SYSA2 (Spread)
80	GTM6
81	Reserved
82	SYSD2 (Spread)
83	Reserved
84	PMC
85	MIXA6 (Spread)
86	Reserved
87	Reserved
88	SYSA3 (Spread)
89	Reserved
90	Reserved
91	SYSD3 (Spread)
92	Reserved
93	Reserved
94	MIXB6 (Spread)
95	GTM3
96	Reserved
97	SYSA4 (Spread)
98	GTM7
99	Reserved
100	SYSD4 (Spread)
101	Reserved
102	Reserved
103	MIXA7 (Spread)
104	Reserved
105	Reserved
106	SYSA5 (Spread)

Table 8-35. Interrupt Source Priority Levels (continued)

Priority	Interrupt Source Description
107	Reserved
108	Reserved
109	SYSD5 (Spread)
110	Reserved
111	Reserved
112	MIXB7 (Spread)
113	GTM1
114	Reserved
115	SYSA6 (Spread)
116	GTM5
117	Reserved
118	SYSD6 (Spread)
119	Reserved
120	Reserved
121	Reserved
122	Reserved
123	SYSA7 (Spread)
124	Reserved
125	Reserved
126	SYSD7 (Spread)
127	Reserved
128	Reserved

8.6.7 Masking Interrupt Sources

By programming the system interrupt mask registers, $SIMSR_x$ and $SEMSR$, the user can mask interrupt requests to the core. Each $SIMSR_x$ and $SEMSR$ bit corresponds to an interrupt source. To enable an interrupt, set the corresponding $SIMSR$ or $SEMSR$ bit. When a masked interrupt source has a pending interrupt request, the corresponding $SIPNR_x$ or $SEMSR$ bit is set, even though the interrupt is not generated to the core. The user can mask all interrupt sources to implement a polling interrupt servicing scheme.

When an interrupt source has multiple interrupting events, the user can individually mask these events by programming a mask register within that particular block. [Table 8-35](#) shows which interrupt sources have multiple interrupting events.

Figure 8-29 shows an example of how the masking occurs, using a DDR block as an example.

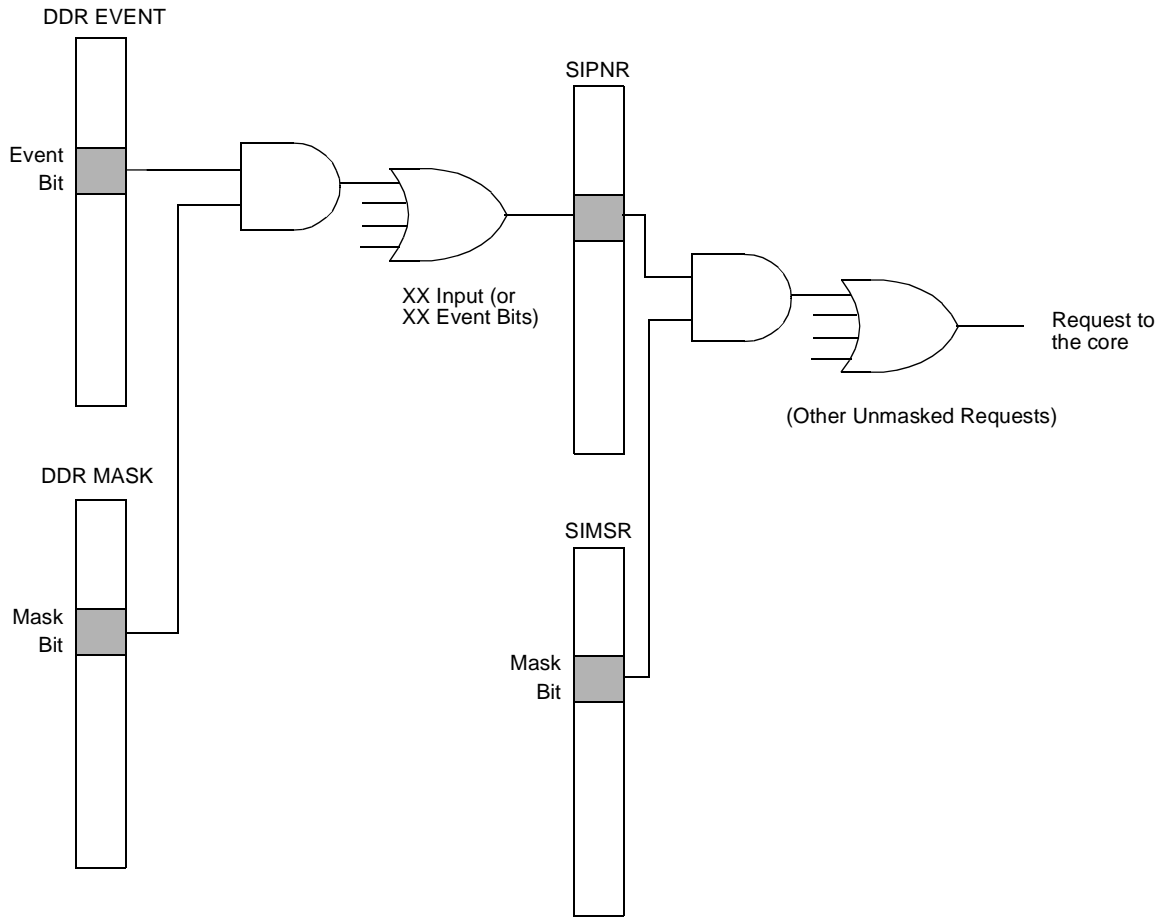


Figure 8-29. DDR Interrupt Request Masking

8.6.8 Interrupt Vector Generation and Calculation

Pending unmasked interrupts are presented to the core in order of priority according to [Table 8-35](#). The interrupt vector that allows the core to locate the interrupt service routine is made available to the core by interrupt handler software reading SIVCR. The interrupt controller passes an interrupt vector corresponding to the highest-priority, unmasked, pending interrupt in response to a read of SIVCR. [Table 8-6](#) lists the encodings for the seven low-order bits of the interrupt vector.

8.6.9 Machine Check Interrupts

The PIC supports the non-maskable machine check interrupts. When an error interrupt signal is received, the interrupt controller indicates the source by setting the corresponding SERSR bit. These sources are listed in [Table 8-22](#).

8.6.10 QUICC Engine Ports Interrupts

The QUICC Engine ports interrupts are a special case of external interrupt requests, which are specifically related to the QUICC Engine block. Each such QUICC Engine ports external interrupt may be generated upon detection of a high-to-low change on the external signal or upon any change on the external signal (note that this is not the same as for normal \overline{IRQ} external interrupt signals). All of the QUICC Engine ports interrupts are managed in three registers CEPIER, CEPIMR & CEPICR. If any of the QUICC Engine ports interrupts is pending and not masked, an internal ‘QE Ports’ event is generated and this is handled through the SIPNR_L register. The specific QUICC Engine ports lines that have this capability are detailed in [Section 8.5.21, “QUICC Engine Ports Interrupt Event Register \(CEPIER\),”](#) and in [Table 8-36](#) below. This feature is specifically useful for pins that can function as modem control signals \overline{CTS} or \overline{CD} .

Table 8-36. QUICC Engine Ports Interrupt Lines

QUICC Engine Port	\overline{CTS} / \overline{CD} Functionality	QUICC Engine Port	\overline{CTS} / \overline{CD} Functionality
PA[15]	UCC1: \overline{CTS}	PD[13]	UCC5: \overline{CD}
PA[16]	UCC1: \overline{CD}	PD[16]	—
PA[29]	UCC2: \overline{CTS}	PD[17]	—
PA[30]	UCC2: \overline{CD}	PD[26]	UCC6: \overline{CTS}
PB[3]	UCC4: \overline{CTS}	PD[27]	UCC6: \overline{CD}
PB[5]	UCC4: \overline{CD}	PE[12]	UCC7: \overline{CTS}
PB[12]	UCC3: \overline{CTS}	PE[13]	UCC7: \overline{CD}
PB[13]	UCC3: \overline{CD}	PE[24]	UCC5: \overline{CTS}
PB[26]	UCC4: \overline{CTS}	PE[25]	UCC5: \overline{CD}
PB[27]	UCC4: \overline{CD}	PE[26]	UCC8: \overline{CTS}
PC[27]	—	PE[27]	UCC8: \overline{CD}
PC[28]	—	PE[31]	—
PC[29]	—	PF[20]	UCC2: \overline{CD}
PD[12]	UCC5: \overline{CTS}	PG[31]	UCC2: \overline{CTS}



Chapter 9

DDR Memory Controller

9.1 DDR Introduction

The fully programmable DDR SDRAM controller supports most JEDEC standard $\times 8$, $\times 16$, or $\times 32$ DDR2 and DDR memories available. In addition, unbuffered and registered DIMMs are supported. However, mixing different memory types or unbuffered and registered DIMMs in the same system is not supported. Built-in error checking and correction (ECC) ensures very low bit-error rates for reliable high-frequency operation. Dynamic power management and auto-precharge modes simplify memory system design. A large set of special features, including ECC error injection, support rapid system debug.

NOTE

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

The MPC8360E has two DDR memory controllers. Each controller supports a 32-bit data bus and two chip selects (physical banks); however, if only the primary controller is used, a 64-bit bus is supported along with four chip selects. This chapter primarily describes a configuration where one controller is used with four chip selects and a 64-bit data bus.

If both controllers are to be used, they must be set to 32-bit bus mode. This is done by setting `DDR_SDRAM_CFG[32_BE]` for both controllers. Note that when using DDR1 in 32-bit mode, burst length should be set to 8 beats. This can be done by setting `DDR_SDRAM_CFG[8_BE]` (for both controllers).

Also note that if 32-bit mode is selected in the primary DDR memory controller, the higher-order data bits (`MDQ[32:63]`) are used for the secondary DDR memory controller. Other than this, all of the board connections and register programming are identical to 64-bit bus mode.

This chapter also describes a single set of external signals. The signals of the two controllers of the MPC8360E are differentiated elsewhere in this document by means of prefixes. For example, the signal that appears in this chapter as `MBA1` may appear elsewhere in this document as either `MEMC1_MBA1` or `MEMC2_MBA1`.

Figure 9-1 is a high-level block diagram of the DDR memory controller with its associated interfaces. Section 9.5, “Functional Description,” contains detailed figures of the controller.

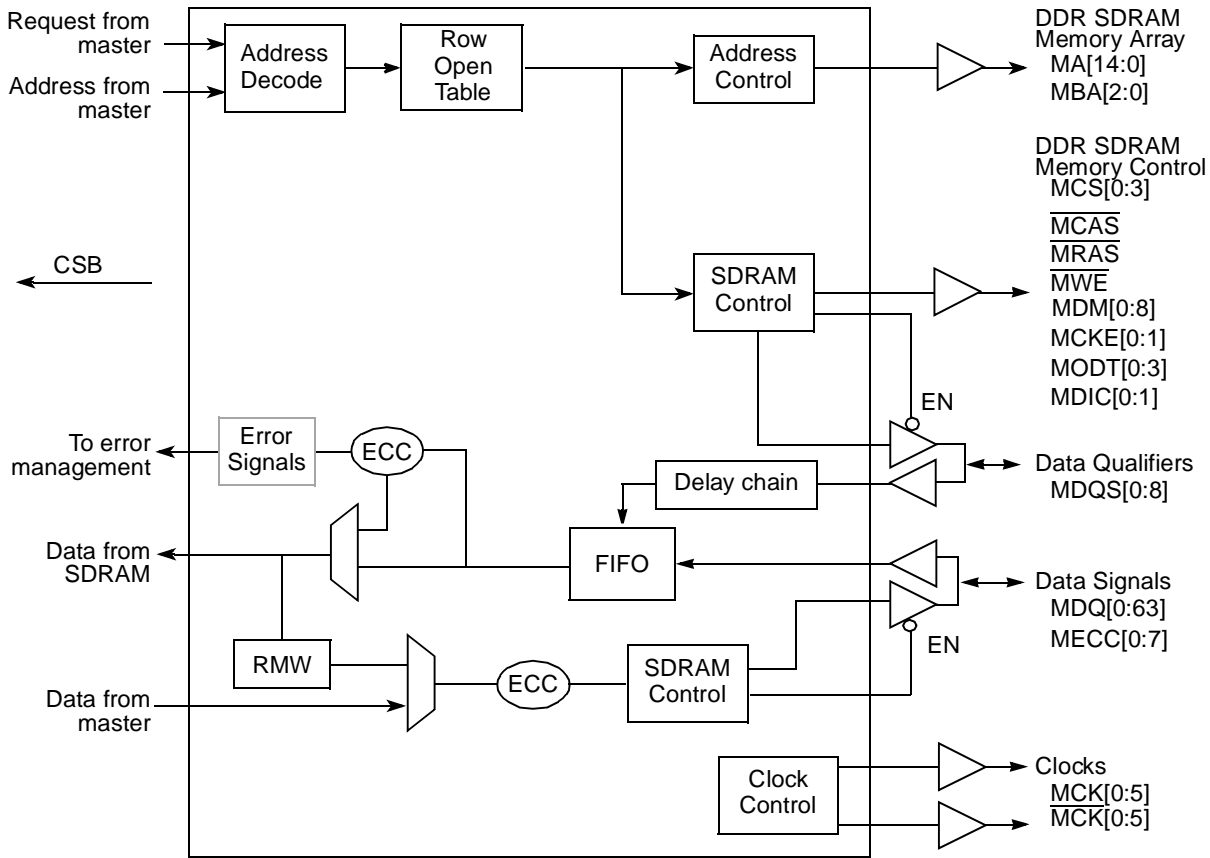


Figure 9-1. DDR Memory Controller Simplified Block Diagram

9.2 DDR Features

The DDR memory controller includes these distinctive features:

- Support for DDR2 and DDR SDRAM
- Dual independent controllers—primary and secondary. Both controllers support 32-bit memories. If only the primary controller is used, it can support 64-bit memories as well.
- 64-/72-bit SDRAM data bus (when using only the primary controller). 32-/40-bit SDRAM (when using both primary and secondary controllers) for DDR and DDR2
- Programmable settings for meeting all SDRAM timing parameters
- Support for the following SDRAM configurations:
 - As many as four physical banks (chip selects), each bank independently addressable
 - When the secondary DDR memory controller is enabled, both primary and secondary DDR memory controllers each support as many as two physical banks (chip selects), each bank independently addressable.

- When the secondary memory controller is disabled, the primary DDR memory controller supports as many as four physical banks (chip selects), each bank independently addressable.
- 64-Mbit to 4-Gbit devices depending on internal device configuration with $\times 8/\times 16/\times 32$ data ports (no direct $\times 4$ support)
- Unbuffered and registered DIMMs
- Chip select interleaving support
- Support for data mask signals and read-modify-write for sub-double-word writes. Note that a read-modify-write sequence is only necessary when ECC is enabled.
- Support for double-bit error detection and single-bit error correction ECC (8-bit check word across 64-bit data)
- Open page management (dedicated entry for each logical bank)
- Automatic DRAM initialization sequence or software-controlled initialization sequence
- Automatic DRAM data initialization
- Support for up to eight posted refreshes
- Memory controller clock frequency of two times the SDRAM clock with support for sleep power management
- Support for error injection

9.2.1 DDR Modes of Operation

The DDR memory controller supports the following modes:

- 64-bit memories are only supported by the primary DDR memory controller and only when the secondary controller is disabled.
In order to use both controllers, they must be set to 32-bit bus mode. This is done by setting `DDR_SDRAM_CFG[32_BE]` for both controllers. Note that when using DDR1 in 32-bit mode, burst length should be set to 8 beats. This can be done by setting `DDR_SDRAM_CFG[8_BE]`.
Also note that if 32-bit mode is selected in the primary DDR memory controller, the higher data bits (`MDQ[32:63]`) are used for the secondary DDR memory controller. Other than this, all of the board connections and register programming are identical to that when 64-bit bus mode is used.
- Dynamic power management mode. The DDR memory controller can reduce power consumption by negating the SDRAM CKE signal when no transactions are pending to the SDRAM.
- Auto-precharge mode. Clearing `DDR_SDRAM_INTERVAL[BSTOPRE]` causes the memory controller to issue an auto-precharge command with every read or write transaction. Auto-precharge mode can be enabled for separate chip selects by setting `CSn_CONFIG[AP_n_EN]`.

9.3 DDR External Signal Descriptions

This section provides descriptions of the DDR memory controller's external signals. It describes each signal's behavior when the signal is asserted or negated and when the signal is an input or an output.

9.3.1 DDR Signals Overview

Memory controller signals are grouped as follows:

- Memory interface signals
- Clock signals
- Debug signals

Table 9-1 shows how DDR memory controller external signals are grouped. The device hardware specification has a pinout diagram showing pin numbers. It also lists all electrical and mechanical specifications.

Table 9-1. DDR Memory Interface Signal Summary

Name	Function/Description	Reset	Pins	I/O
MDQ[0:63]	Data bus	All zeros	64	I/O
MDQS[0:8]	Data strobes	All zeros	9	I/O
MECC[0:7]	Error checking and correcting	All zeros	8	I/O
$\overline{\text{MCAS}}$	Column address strobe	One	1	O
MA[14:0]	Address bus	All zeros	15	O
MBA[2:0]	Logical bank address	All zeros	3	O
$\overline{\text{MCS}}[0:3]$	Chip selects	All ones	4	O
$\overline{\text{MWE}}$	Write enable	One	1	O
$\overline{\text{MRAS}}$	Row address strobe	One	1	O
MDM[0:8]	Data mask	All zeros	9	O
MCK[0:5]	DRAM clock outputs	All zeros	6	O
$\overline{\text{MCK}}[0:5]$	DRAM clock outputs (complement)	All zeros	6	O
MCKE[0:1]	DRAM clock enable	All zeros	2	O
MODT[0:3]	DRAM on-die termination external control.	All zeros	4	O
MDVAL	Memory debug data valid	Zero	1	O
MSRCID[0:4]	Memory debug source ID	All zeros	5	O
MDIC[0:1]	Driver impedance calibration	High Z	2	I/O

Table 9-2 shows the memory address signal mappings.

Table 9-2. Memory Address Signal Mappings

Signal Name (Outputs)		JEDEC DDR DIMM Signals (Inputs)
msb	MA14	A14
	MA13	A13
	MA12	A12
	MA11	A11
	MA10	A10 (AP for DDR) ¹
	MA9	A9
	MA8	A8 (alternate AP for DDR) ²
	MA7	A7
	MA6	A6
	MA5	A5
	MA4	A4
	MA3	A3
	MA2	A2
	MA1	A1
lsb	MA0	A0
msb	MBA2	MBA2
	MBA1	MBA1
lsb	MBA0	MBA0

¹ Auto-precharge for DDR signaled on A10 when DDR_SDRAM_CFG[PCHB8] = 0.

² Auto-precharge for DDR signaled on A8 when DDR_SDRAM_CFG[PCHB8] = 1.

9.3.2 DDR Detailed Signal Descriptions

The following sections describe the DDR SDRAM controller input and output signals, the meaning of their different states, and relative timing information for assertion and negation.

9.3.2.1 Memory Interface Signals

Table 9-3 describes the DDR controller memory interface signals.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
MDQ[0:63]	I/O	Data bus. Both input and output signals on the DDR memory controller.	
	O	As outputs for the bidirectional data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated—Represent the value of data being driven by the DDR memory controller.
		Timing	Assertion/Negation—Driven coincident with corresponding data strobes (MDQS) signal. High impedance—No READ or WRITE command is in progress; data is not being driven by the memory controller or the DRAM.
	I	As inputs for the bidirectional data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated—Represents the state of data being driven by the external DDR SDRAMs.
Timing		Assertion/Negation—The DDR SDRAM drives data during a READ transaction. High impedance—No READ or WRITE command in progress; data is not being driven by the memory controller or the DRAM.	
MDQS[0:8]	I/O	Data strobes. Inputs with read data, outputs with write data.	
		O	As outputs, the data strobes are driven by the DDR memory controller during a write transaction. The memory controller always drives these signals low unless a read has been issued and incoming data strobes are expected. This keeps the data strobes from floating high when there are no transactions on the DRAM interface.
			State Meaning
	Timing	Assertion/Negation—If a WRITE command is registered at clock edge n , data strobes at the DRAM assert centered in the data eye on clock edge $n + 1$. See the JEDEC DDR SDRAM specification for more information.	
	I	As inputs, the data strobes are driven by the external DDR SDRAMs during a read transaction. The data strobes are used by the memory controller to synchronize data latching.	
		State Meaning	Asserted/Negated—Driven high when positive capture data is received and driven low when negative capture data is received. Centered in the data eye for writes; coincident with the data eye for reads. Treated as a clock. Data is valid when signals toggle. See Table 9-36 for byte lane assignments.
Timing		Assertion/Negation—If a READ command is registered at clock edge n , and the latency is programmed in TIMING_CFG_1[CASLAT] to be m clocks, data strobes at the DRAM assert coincident with the data on clock edge $n + m$. See the JEDEC DDR SDRAM specification for more information.	

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
MECC[0:7]	I/O	Error checking and correcting codes. Input and output signals for the DDR controller's bidirectional ECC bus. MECC[0:5] function in both normal and debug modes.	
	O	As normal mode outputs the ECC signals represent the state of ECC driven by the DDR controller on writes. As debug mode outputs MECC[0:5] provide source ID and data-valid information. See Section 9.5.11, "Error Checking and Correcting (ECC)," for more details.	
		State Meaning	Asserted/Negated—Represents the state of ECC being driven by the DDR controller on writes.
		Timing	Assertion/Negation—Same timing as MDQ High impedance—Same timing as MDQ
	I	As inputs, the ECC signals represent the state of ECC driven by the SDRAM devices on reads.	
		State Meaning	Asserted/Negated—Represents the state of ECC being driven by the DDR SDRAMs on reads.
Timing		Assertion/Negation—Same timing as MDQ High impedance—Same timing as MDQ	
MA[14:0]	O	Address bus. Memory controller outputs for the address to the DRAM. MA[14:0] carry 15 of the address bits for the DDR memory interface corresponding to the row and column address bits. MA0 is the lsb of the address output from the memory controller.	
		State Meaning	Asserted/Negated—Represents the address driven by the DDR memory controller. Contains different portions of the address depending on the memory size and the DRAM command being issued by the memory controller. See Table 9-40 and Table 9-2 for a complete description of the mapping of these signals.
		Timing	Assertion/Negation—The address lines are only driven when the controller has a command scheduled to issue on the address/CMD bus; otherwise they will be at high-Z. It is valid when a transaction is driven to DRAM (when \overline{MCS}_n is active). High impedance—When the memory controller is disabled
MBA[2:0]	O	Logical bank address. Outputs that drive the logical (or internal) bank address pins of the SDRAM. Each SDRAM supports four or eight addressable logical sub-banks. Bit zero of the memory controller's output bank address must be connected to bit zero of the SDRAM's input bank address. MBA0, the least-significant bit of the three bank address signals, is asserted during the mode register set command to specify the extended mode register.	
		State Meaning	Asserted/Negated—Selects the DDR SDRAM logical (or internal) bank to be activated during the row address phase and selects the SDRAM internal bank for the read or write operation during the column address phase of the memory access. Table 9-40 and Table 9-2 describes the mapping of these signals in all cases.
		Timing	Assertion/Negation—Same timing as MA_n High impedance—Same timing as MA_n

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
$\overline{\text{MCAS}}$	O	Column address strobe. Active-low SDRAM address multiplexing signal. $\overline{\text{MCAS}}$ is asserted for read or write transactions and for mode register set, refresh, and precharge commands.
		<p>State Meaning Asserted—Indicates that a valid SDRAM column address is on the address bus for read and write transactions. See Table 9-45 for more information on the states required on $\overline{\text{MCAS}}$ for various other SDRAM commands.</p> <p>Negated—The column address is not guaranteed to be valid.</p>
		<p>Timing Assertion/Negation—Assertion and negation timing is directed by the values described in Section 9.4.1.4, “DDR SDRAM Timing Configuration 0 (TIMING_CFG_0),” Section 9.4.1.5, “DDR SDRAM Timing Configuration 1 (TIMING_CFG_1),” Section 9.4.1.6, “DDR SDRAM Timing Configuration 2 (TIMING_CFG_2),” and Section 9.4.1.3, “DDR SDRAM Timing Configuration 3 (TIMING_CFG_3).”</p> <p>High impedance—$\overline{\text{MCAS}}$ is always driven unless the memory controller is disabled.</p>
$\overline{\text{MRAS}}$	O	Row address strobe. Active-low SDRAM address multiplexing signal. Asserted for activate commands. In addition; used for mode register set commands and refresh commands.
		<p>State Meaning Asserted—Indicates that a valid SDRAM row address is on the address bus for read and write transactions. See Table 9-45 for more information on the states required on $\overline{\text{MRAS}}$ for various other SDRAM commands.</p> <p>Negated—The row address is not guaranteed to be valid.</p>
		<p>Timing Assertion/Negation—Assertion and negation timing is directed by the values described in Section 9.4.1.4, “DDR SDRAM Timing Configuration 0 (TIMING_CFG_0),” Section 9.4.1.5, “DDR SDRAM Timing Configuration 1 (TIMING_CFG_1),” Section 9.4.1.6, “DDR SDRAM Timing Configuration 2 (TIMING_CFG_2),” and Section 9.4.1.3, “DDR SDRAM Timing Configuration 3 (TIMING_CFG_3).”</p> <p>High impedance—$\overline{\text{MRAS}}$ is always driven unless the memory controller is disabled.</p>
$\overline{\text{MCS}}[0:3]$	O	Chip selects. Four chip selects supported by the memory controller.
		<p>State Meaning Asserted—Selects a physical SDRAM bank to perform a memory operation as described in Section 9.4.1.1, “Chip Select Memory Bounds (CSn_BNDS),” and Section 9.4.1.2, “Chip Select Configuration (CSn_CONFIG).” The DDR controller asserts one of the $\overline{\text{MCS}}[0:3]$ signals to begin a memory cycle.</p> <p>Negated—Indicates no SDRAM action during the current cycle.</p>
		<p>Timing Assertion/Negation—Asserted to signal any new transaction to the SDRAM. The transaction must adhere to the timing constraints set in TIMING_CFG_0–TIMING_CFG_3.</p> <p>High impedance—Always driven unless the memory controller is disabled.</p>
$\overline{\text{MWE}}$	O	Write enable. Asserted when a write transaction is issued to the SDRAM. This is also used for mode registers set commands and precharge commands.
		<p>State Meaning Asserted—Indicates a memory write operation. See Table 9-45 for more information on the states required on $\overline{\text{MWE}}$ for various other SDRAM commands.</p> <p>Negated—Indicates a memory read operation.</p>
		<p>Timing Assertion/Negation—Similar timing as $\overline{\text{MRAS}}$ and $\overline{\text{MCAS}}$. Used for write commands.</p> <p>High impedance—$\overline{\text{MWE}}$ is always driven unless the memory controller is disabled.</p>

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
MDM[0:8]	O	DDR SDRAM data output mask. Masks unwanted bytes of data transferred during a write. They are needed to support sub-burst-size transactions (such as single-byte writes) on SDRAM where all I/O occurs in multi-byte bursts. MDM0 corresponds to the most significant byte (MSB) and MDM7 corresponds to the LSB, while MDM8 corresponds to the ECC byte. Table 9-36 shows byte lane encodings.	
		State Meaning	Asserted—Prevents writing to DDR SDRAM. Asserted when data is written to DRAM if the corresponding byte(s) should be masked for the write. Note that the MDM n signals are active-high for the DDR controller. MDM n is part of the DDR command encoding. Negated—Allows the corresponding byte to be read from or written to the SDRAM.
		Timing	Assertion/Negation—Same timing as MDQ x as outputs. High impedance—Always driven unless the memory controller is disabled.
MODT[0:3]	O	On-Die termination. Memory controller outputs for the ODT to the DRAM. MODT[0:3] represents the on-die termination for the associated data, data masks, ECC, and data strobes.	
		State Meaning	Asserted/Negated—Represents the ODT driven by the DDR memory controller.
		Timing	Assertion/Negation—Driven in accordance with JEDEC DRAM specifications for on-die termination timings. It is configured through the CS n _CONFIG[ODT_RD_CFG] and CS n _CONFIG[ODT_WR_CFG] fields. High impedance—Always driven.
MDIC[0:1]	I/O	Driver impedance calibration. Note that the MDIC signals require the use of 18.2- Ω precision 1% resistors; MDIC0 must be pulled to GND, while MDIC1 must be pulled to GV _{DD} .	
		State Meaning	These pins are used for automatic calibration of the DDR IOs.
		Timing	These are driven for four DRAM cycles at a time while the DDR controller is executing the automatic driver compensation.

9.3.2.2 Clock Interface Signals

[Table 9-4](#) contains the detailed descriptions of the clock signals of the DDR controller.

Table 9-4. Clock Signals—Detailed Signal Descriptions

Signal	I/O	Description	
MCK[0:5], $\overline{\text{MCK}}$ [0:5]	O	DRAM clock outputs and their complements. See Section 9.5.4.1, “Clock Distribution.” Enabled by the MCKENR n register. See Section 4.5.3.1, “MCK Enable Register (MCKENRn),” for details.	
		State Meaning	Asserted/Negated—The JEDEC DDR SDRAM specifications require true and complement clocks. A clock edge is seen by the SDRAM when the true and complement cross.
		Timing	Assertion/Negation—Timing is controlled by the DDR_CLK_CNTL register at offset 0x130.

Table 9-4. Clock Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
MCKE[0:1]	O	Clock enable. Output signals used as the clock enables to the SDRAM. MCKE[0:1] can be negated to stop clocking the DDR SDRAM. The MCKE signals should be connected to the same rank of memory as the corresponding \overline{MCS} and \overline{MODT} signals. For example, MCKE[0] should be connected to the same rank of memory as $\overline{MCS}[0]$ and $\overline{MODT}[0]$.
		State Meaning Asserted—Clocking to the SDRAM is enabled. Negated—Clocking to the SDRAM is disabled and the SDRAM should ignore signal transitions on MCK or \overline{MCK} . MCK/ \overline{MCK} are don't cares while MCKE[0:1] are negated.
		Timing Assertion/Negation—Asserted when DDR_SDRAM_CFG[MEM_EN] is set. Can be negated when entering dynamic power management or self refresh. Are asserted again when exiting dynamic power management or self refresh. High impedance—Always driven.

9.3.2.3 Debug Signals

The debug signals MSRCID[0:4] and MDVAL have no function in normal DDR controller operation. A detailed description of these signals can be found in [Section 5.4.3.7, “Debug Configuration.”](#)

9.4 DDR Memory Map/Register Definition

[Table 9-5](#) shows the register memory map for the DDR memory controller.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 9-5. DDR Memory Controller Memory Map

Offset	Register	Access	Reset	Section/Page
DDR Memory Controller—Block Base Address 0x0_2000				
0x000	CS0_BNDS—Chip select 0 memory bounds	R/W	All zeros	9.4.1.1/9-12
0x008	CS1_BNDS—Chip select 1 memory bounds	R/W	All zeros	9.4.1.1/9-12
0x010	CS2_BNDS—Chip select 2 memory bounds	R/W	All zeros	9.4.1.1/9-12
0x018	CS3_BNDS—Chip select 3 memory bounds	R/W	All zeros	9.4.1.1/9-12
0x080	CS0_CONFIG—Chip select 0 configuration	R/W	All zeros	9.4.1.2/9-12
0x084	CS1_CONFIG—Chip select 1 configuration	R/W	All zeros	9.4.1.2/9-12
0x088	CS2_CONFIG—Chip select 2 configuration	R/W	All zeros	9.4.1.2/9-12
0x08C	CS3_CONFIG—Chip select 3 configuration	R/W	All zeros	9.4.1.2/9-12

Table 9-5. DDR Memory Controller Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	All zeros	9.4.1.3/9-14
0x104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	9.4.1.4/9-15
0x108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	All zeros	9.4.1.5/9-17
0x10C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	All zeros	9.4.1.6/9-19
0x110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.7/9-21
0x114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	All zeros	9.4.1.8/9-23
0x118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	All zeros	9.4.1.9/9-25
0x11C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	All zeros	9.4.1.10/9-26
0x120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	All zeros	9.4.1.11/9-26
0x124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	All zeros	9.4.1.12/9-29
0x128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	All zeros	9.4.1.13/9-29
0x130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	9.4.1.14/9-30
0x140– 0x144	Reserved	—	—	—
0x148	DDR_INIT_ADDR—DDR training initialization address	R/W	All zeros	9.4.1.15/9-30
0x150– 0xBF4	Reserved	—	—	—
0xBF8	DDR_IP_REV1—DDR IP block revision 1	R	0xnnnn_nnnn ¹	9.4.1.16/9-31
0xBFC	DDR_IP_REV2—DDR IP block revision 2	R	0x00nn_00nn ¹	9.4.1.17/9-31
0xE00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	All zeros	9.4.1.18/9-32
0xE04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	All zeros	9.4.1.19/9-32
0xE08	ERR_INJECT—Memory data path error injection mask ECC	R/W	All zeros	9.4.1.20/9-33
0xE20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	All zeros	9.4.1.21/9-33
0xE24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	All zeros	9.4.1.22/9-34
0xE28	CAPTURE_ECC—Memory data path read capture ECC	R/W	All zeros	9.4.1.23/9-34
0xE40	ERR_DETECT—Memory error detect	w1c	All zeros	9.4.1.24/9-34
0xE44	ERR_DISABLE—Memory error disable	R/W	All zeros	9.4.1.25/9-35
0xE48	ERR_INT_EN—Memory error interrupt enable	R/W	All zeros	9.4.1.26/9-36
0xE4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	All zeros	9.4.1.27/9-37
0xE50	CAPTURE_ADDRESS—Memory error address capture	R/W	All zeros	9.4.1.28/9-38
0xE58	ERR_SBE—Single-Bit ECC memory error management	R/W	All zeros	9.4.1.29/9-39

¹ Implementation-dependent reset values are listed in specified section/page.

9.4.1 DDR Register Descriptions

This section describes the DDR memory controller registers. Shading indicates reserved fields that should not be written.

9.4.1.1 Chip Select Memory Bounds (CS_n_BNDS)

The chip select bounds registers (CS_n_BNDS) define the starting and ending address of the memory space that corresponds to the individual chip selects. Note that the size specified in CS_n_BNDS should equal the size of physical DRAM. Also, note that EAn must be greater than or equal to SAn .

If chip select interleaving is enabled, all fields in the lower interleaved chip select are used, and the other chip selects' bounds registers are unused. For example, if chip selects 0 and 1 are interleaved, all fields in $CS0_BNDS$ are used, and all fields in $CS1_BNDS$ are unused.

CS_n_BNDS are shown in [Figure 9-2](#).



Figure 9-2. Chip Select Bounds Registers (CS_n_BNDS)

[Table 9-6](#) describes the CS_n_BNDS register fields.

Table 9-6. CS_n_BNDS Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	SAn	Starting address for chip select (bank) n . This value is compared against the 8 msbs of the 32-bit address.
16–23	—	Reserved
24–31	EAn	Ending address for chip select (bank) n . This value is compared against the 8 msbs of the 32-bit address.

9.4.1.2 Chip Select Configuration (CS_n_CONFIG)

The chip select configuration (CS_n_CONFIG) registers shown in [Figure 9-3](#) enable the DDR chip selects and set the number of row and column bits used for each chip select. These registers should be loaded with the correct number of row and column bits for each SDRAM. Because $CS_n_CONFIG[ROW_BITS_CS_n, COL_BITS_CS_n]$ establish address multiplexing, the user should take great care to set these values correctly.

If chip select interleaving is enabled, then all fields in the lower interleaved chip select are used, and the other registers' fields are unused, with the exception of the ODT_RD_CFG and ODT_WR_CFG fields. For example, if chip selects 0 and 1 are interleaved, all fields in $CS0_CONFIG$ are used, but only the ODT_RD_CFG and ODT_WR_CFG fields in $CS1_CONFIG$ are used.

Offset 0x080, 0x084, 0x088, 0x08C

Access: Read/Write

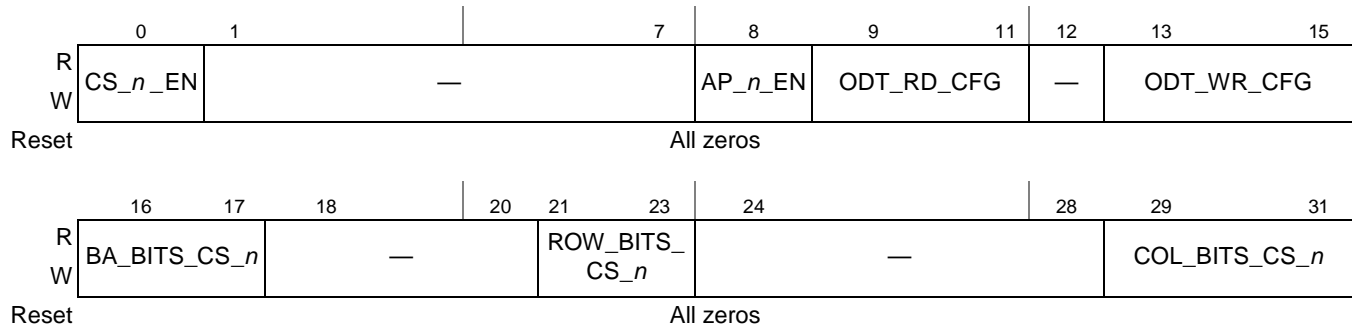

Figure 9-3. Chip Select Configuration Register (CS_n_CONFIG)

 Table 9-7 describes the CS_n_CONFIG register fields.

Table 9-7. CS_n_CONFIG Field Descriptions

Bits	Name	Description
0	CS _n _EN	Chip select <i>n</i> enable 0 Chip select <i>n</i> is not active 1 Chip select <i>n</i> is active and assumes the state set in CS _n _BNDS.
1–7	—	Reserved
8	AP _n _EN	Chip select <i>n</i> auto-precharge enable 0 Chip select <i>n</i> is only auto-precharged if global auto-precharge mode is enabled (DDR_SDRAM_INTERVAL[BSTOPRE] = 0). 1 Chip select <i>n</i> always issues an auto-precharge for read and write transactions.
9–11	ODT_RD_CFG	ODT for reads configuration. Note that CAS latency plus additive latency must be at least 3 cycles for ODT_RD_CFG to be enabled. ODT should only be used with DDR2 memories. 000 Never assert ODT for reads 001 Assert ODT only during reads to CS _n 010 Assert ODT only during reads to other chip selects 011 Assert ODT only during reads to other DIMM modules. It is assumed that CS0 and CS1 are on the same DIMM module, whereas CS2 and CS3 are on a separate DIMM module. 100 Assert ODT for all reads 101–111 Reserved
12	—	Reserved
13–15	ODT_WR_CFG	ODT for writes configuration. Note that write latency plus additive latency must be at least 3 cycles for ODT_WR_CFG to be enabled. ODT should only be used with DDR2 memories. 000 Never assert ODT for writes 001 Assert ODT only during writes to CS _n 010 Assert ODT only during writes to other chip selects 011 Assert ODT only during writes to other DIMM modules. It is assumed that CS0 and CS1 are on the same DIMM module, whereas CS2 and CS3 are on a separate DIMM module. 100 Assert ODT for all writes 101–111 Reserved
16–17	BA_BITS_CS _n	Number of bank bits for SDRAM on chip select <i>n</i> . These bits correspond to the sub-bank bits driven on MBA _n in Table 9-42. 00 2 logical bank bits 01 3 logical bank bits 10–11 Reserved

Table 9-7. CS_n_CONFIG Field Descriptions (continued)

Bits	Name	Description
18–20	—	Reserved
21–23	ROW_BITS_CS _n	Number of row bits for SDRAM on chip select <i>n</i> . See Table 9-42 for details. 000 12 row bits 001 13 row bits 010 14 row bits 011 15 row bits 000–111 Reserved
24–28	—	Reserved
29–31	COL_BITS_CS _n	Number of column bits for SDRAM on chip select <i>n</i> . For DDR, the decoding is as follows: 000 8 column bits 001 9 column bits 010 10 column bits 011 11 column bits 100–111 Reserved

9.4.1.3 DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)

DDR SDRAM timing configuration register 3, shown in [Figure 9-4](#), sets the extended refresh recovery time, which is combined with TIMING_CFG_1[REFREC] to determine the full refresh recovery time.

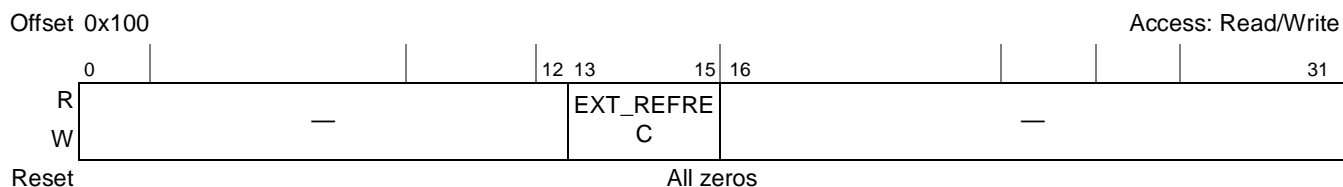


Figure 9-4. DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)

[Table 9-8](#) describes TIMING_CFG_3 fields.

Table 9-8. TIMING_CFG_3 Field Descriptions

Bits	Name	Description
0–12	—	Reserved, should be cleared.
13–15	EXT_REFREC	Extended refresh recovery time (t_{RFC}). Controls the number of clock cycles from a refresh command until an activate command is allowed. This field is concatenated with TIMING_CFG_1[REFREC] to obtain 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. $t_{RFC} = \{EXT_REFREC \parallel REFREC\} + 8$, such that t_{RFC} is calculated as follows: 000 0 clocks 001 16 clocks 010 32 clocks 011 48 clocks 100 64 clocks 101 80 clocks 110 96 clocks 111 112 clocks
16–31	—	Reserved, should be cleared.

9.4.1.4 DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)

DDR SDRAM timing configuration register 0, shown in Figure 9-5, sets the number of clock cycles between various SDRAM control commands.

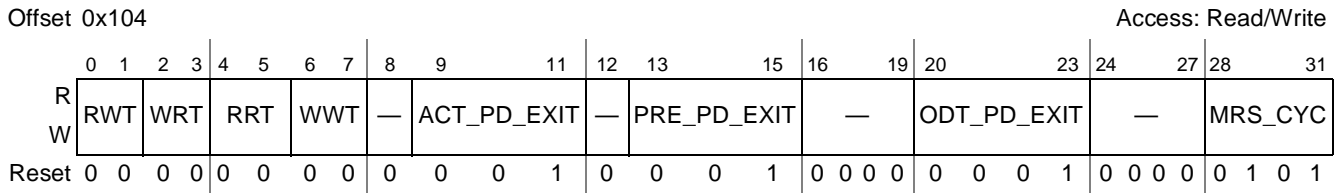


Figure 9-5. DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)

Table 9-9 describes TIMING_CFG_0 fields.

Table 9-9. TIMING_CFG_0 Field Descriptions

Bits	Name	Description
0–1	RWT	Read-to-write turnaround (t_{RTW}). Specifies how many extra cycles are added between a read to write turnaround. If 0 clocks is chosen, then the DDR controller uses a fixed number based on the CAS latency and write latency. Choosing a value other than 0 adds extra cycles past this default calculation. As a default the DDR controller determines the read-to-write turnaround as $CL - WL + BL \div 2 + 2$. In this equation, CL is the CAS latency rounded up to the next integer, WL is the programmed write latency, and BL is the burst length. 00 0 clocks 10 2 clocks 01 1 clock 11 3 clocks
2–3	WRT	Write-to-read turnaround. Specifies how many extra cycles are added between a write to read turnaround. If 0 clocks is chosen, then the DDR controller uses a fixed number based on the, read latency, and write latency. Choosing a value other than 0 adds extra cycles past this default calculation. As a default, the DDR controller determines the write-to-read turnaround as $WL - CL + BL \div 2 + 1$. In this equation, CL is the CAS latency rounded down to the next integer, WL is the programmed write latency, and BL is the burst length. 00 0 clocks 10 2 clocks 01 1 clock 11 3 clocks
4–5	RRT	Read-to-read turnaround. Specifies how many extra cycles are added between reads to different chip selects. As a default, 3 cycles are required between read commands to different chip selects. Extra cycles may be added with this field. Note: If 8-beat bursts are enabled, then 5 cycles are the default. Note that DDR2 does not support 8-beat bursts. 00 0 clocks 10 2 clocks 01 1 clock 11 3 clocks
6–7	WWT	Write-to-write turnaround. Specifies how many extra cycles are added between writes to different chip selects. As a default, 2 cycles are required between write commands to different chip selects. Extra cycles may be added with this field. Note: If 8-beat bursts are enabled, then 4 cycles are the default. Note that DDR2 does not support 8-beat bursts. 00 0 clocks 10 2 clocks 01 1 clock 11 3 clocks
8	—	Reserved, should be cleared.

Table 9-9. TIMING_CFG_0 Field Descriptions (continued)

Bits	Name	Description
9–11	ACT_PD_EXIT	Active powerdown exit timing (t_{XARD} and t_{XARDS}). Specifies how many clock cycles to wait after exiting active powerdown before issuing any command. 000 Reserved 100 4 clocks 001 1 clock 101 5 clocks 010 2 clocks 110 6 clocks 011 3 clocks 111 7 clocks
12	—	Reserved, should be cleared.
13–15	PRE_PD_EXIT	Precharge powerdown exit timing (t_{XP}). Specifies how many clock cycles to wait after exiting precharge powerdown before issuing any command. 000 Reserved 100 4 clocks 001 1 clock 101 5 clocks 010 2 clocks 110 6 clocks 011 3 clocks 111 7 clocks
16–19	—	Reserved, should be cleared.
20–23	ODT_PD_EXIT	ODT powerdown exit timing (t_{AXPD}). Specifies how many clocks must pass after exiting powerdown before ODT may be asserted. 0000 0 clock 1000 8 clocks 0001 1 clock 1001 9 clocks 0010 2 clocks 1010 10 clocks 0011 3 clocks 1011 11 clocks 0100 4 clocks 1100 12 clocks 0101 5 clocks 1101 13 clocks 0110 6 clocks 1110 14 clocks 0111 7 clocks 1111 15 clocks
24–27	—	Reserved, should be cleared.
28–31	MRS_CYC	Mode register set cycle time (t_{MRD}). Specifies the number of cycles that must pass after a Mode Register Set command until any other command. 0000 Reserved 1000 8 clocks 0001 1 clock 1001 9 clocks 0010 2 clocks 1010 10 clocks 0011 3 clocks 1011 11 clocks 0100 4 clocks 1100 12 clocks 0101 5 clocks 1101 13 clocks 0110 6 clocks 1110 14 clocks 0111 7 clocks 1111 15 clocks

9.4.1.5 DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)

DDR SDRAM timing configuration register 1, shown in [Figure 9-6](#), sets the number of clock cycles between various SDRAM control commands.

Offset 0x108

Access: Read/Write

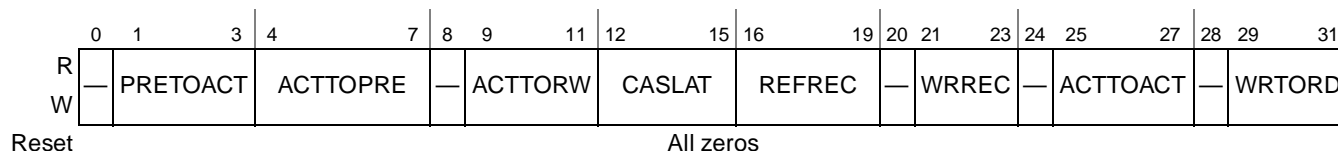


Figure 9-6. DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)

[Table 9-10](#) describes TIMING_CFG_1 fields.

Table 9-10. TIMING_CFG_1 Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared.
1–3	PRETOACT	Precharge-to-activate interval (t_{RP}). Determines the number of clock cycles from a precharge command until an activate or refresh command is allowed. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
4–7	ACTTOPRE	Activate to precharge interval (t_{RAS}). Determines the number of clock cycles from an activate command until a precharge command is allowed. 0000 16 clocks 0101 5 clocks 0001 17 clocks 0110 6 clocks 0010 18 clocks 0111 7 clocks 0011 19 clocks ... 0100 4 clocks 1111 15 clocks
8	—	Reserved, should be cleared.
9–11	ACTTORW	Activate to read/write interval for SDRAM (t_{RCD}). Controls the number of clock cycles from an activate command until a read or write command is allowed. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks

Table 9-10. TIMING_CFG_1 Field Descriptions (continued)

Bits	Name	Description																																
12–15	CASLAT	<p>MCAS latency from READ command. Number of clock cycles between registration of a READ command by the SDRAM and the availability of the first output data. If a READ command is registered at clock edge n and the latency is m clocks, data is available nominally coincident with clock edge $n + m$. This value must be programmed at initialization as described in Section 9.4.1.8, “DDR SDRAM Control Configuration 2 (DDR_SDRAM_CFG_2).”</p> <table> <tr> <td>0000</td> <td>Reserved</td> <td>1000</td> <td>4.5 clocks</td> </tr> <tr> <td>0001</td> <td>1 clock</td> <td>1001</td> <td>5 clocks</td> </tr> <tr> <td>0010</td> <td>1.5 clocks</td> <td>1010</td> <td>5.5 clocks</td> </tr> <tr> <td>0011</td> <td>2 clocks</td> <td>1011</td> <td>6 clocks</td> </tr> <tr> <td>0100</td> <td>2.5 clocks</td> <td>1100</td> <td>6.5 clocks</td> </tr> <tr> <td>0101</td> <td>3 clocks</td> <td>1101</td> <td>7 clocks</td> </tr> <tr> <td>0110</td> <td>3.5 clocks</td> <td>1110</td> <td>7.5 clocks</td> </tr> <tr> <td>0111</td> <td>4 clocks</td> <td>1111</td> <td>8 clocks</td> </tr> </table>	0000	Reserved	1000	4.5 clocks	0001	1 clock	1001	5 clocks	0010	1.5 clocks	1010	5.5 clocks	0011	2 clocks	1011	6 clocks	0100	2.5 clocks	1100	6.5 clocks	0101	3 clocks	1101	7 clocks	0110	3.5 clocks	1110	7.5 clocks	0111	4 clocks	1111	8 clocks
0000	Reserved	1000	4.5 clocks																															
0001	1 clock	1001	5 clocks																															
0010	1.5 clocks	1010	5.5 clocks																															
0011	2 clocks	1011	6 clocks																															
0100	2.5 clocks	1100	6.5 clocks																															
0101	3 clocks	1101	7 clocks																															
0110	3.5 clocks	1110	7.5 clocks																															
0111	4 clocks	1111	8 clocks																															
16–19	REFREC	<p>Refresh recovery time (t_{RFC}). Controls the number of clock cycles from a refresh command until an activate command is allowed. This field is concatenated with TIMING_CFG_3[EXTREFREC] to obtain a 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, such that t_{RFC} is calculated as follows: $t_{RFC} = \{EXT_REFREC REFREC\} + 8$.</p> <table> <tr> <td>0000</td> <td>8 clocks</td> <td>0011</td> <td>11 clocks</td> </tr> <tr> <td>0001</td> <td>9 clocks</td> <td>...</td> <td></td> </tr> <tr> <td>0010</td> <td>10 clocks</td> <td>1111</td> <td>23 clocks</td> </tr> </table>	0000	8 clocks	0011	11 clocks	0001	9 clocks	...		0010	10 clocks	1111	23 clocks																				
0000	8 clocks	0011	11 clocks																															
0001	9 clocks	...																																
0010	10 clocks	1111	23 clocks																															
20	—	Reserved, should be cleared.																																
21–23	WRREC	<p>Last data to precharge minimum interval (t_{WR}). Determines the number of clock cycles from the last data associated with a write command until a precharge command is allowed.</p> <table> <tr> <td>000</td> <td>Reserved</td> </tr> <tr> <td>001</td> <td>1 clock</td> </tr> <tr> <td>010</td> <td>2 clocks</td> </tr> <tr> <td>011</td> <td>3 clocks</td> </tr> <tr> <td>100</td> <td>4 clocks</td> </tr> <tr> <td>101</td> <td>5 clocks</td> </tr> <tr> <td>110</td> <td>6 clocks</td> </tr> <tr> <td>111</td> <td>7 clocks</td> </tr> </table>	000	Reserved	001	1 clock	010	2 clocks	011	3 clocks	100	4 clocks	101	5 clocks	110	6 clocks	111	7 clocks																
000	Reserved																																	
001	1 clock																																	
010	2 clocks																																	
011	3 clocks																																	
100	4 clocks																																	
101	5 clocks																																	
110	6 clocks																																	
111	7 clocks																																	
24	—	Reserved, should be cleared.																																
25–27	ACTTOACT	<p>Activate-to-activate interval (t_{RRD}). Number of clock cycles from an activate command until another activate command is allowed for a different logical bank in the same physical bank (chip select).</p> <table> <tr> <td>000</td> <td>Reserved</td> <td>100</td> <td>4 clocks</td> </tr> <tr> <td>001</td> <td>1 clock</td> <td>101</td> <td>5 clocks</td> </tr> <tr> <td>010</td> <td>2 clocks</td> <td>110</td> <td>6 clocks</td> </tr> <tr> <td>011</td> <td>3 clocks</td> <td>111</td> <td>7 clocks</td> </tr> </table>	000	Reserved	100	4 clocks	001	1 clock	101	5 clocks	010	2 clocks	110	6 clocks	011	3 clocks	111	7 clocks																
000	Reserved	100	4 clocks																															
001	1 clock	101	5 clocks																															
010	2 clocks	110	6 clocks																															
011	3 clocks	111	7 clocks																															
28	—	Reserved, should be cleared.																																
29–31	WRTORD	<p>Last write data pair to read command issue interval (t_{WTR}). Number of clock cycles between the last write data pair and the subsequent read command to the same physical bank.</p> <table> <tr> <td>000</td> <td>Reserved</td> <td>100</td> <td>4 clocks</td> </tr> <tr> <td>001</td> <td>1 clock</td> <td>101</td> <td>5 clocks</td> </tr> <tr> <td>010</td> <td>2 clocks</td> <td>110</td> <td>6 clocks</td> </tr> <tr> <td>011</td> <td>3 clocks</td> <td>111</td> <td>7 clocks</td> </tr> </table>	000	Reserved	100	4 clocks	001	1 clock	101	5 clocks	010	2 clocks	110	6 clocks	011	3 clocks	111	7 clocks																
000	Reserved	100	4 clocks																															
001	1 clock	101	5 clocks																															
010	2 clocks	110	6 clocks																															
011	3 clocks	111	7 clocks																															

9.4.1.6 DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)

DDR SDRAM timing configuration 2, shown in [Figure 9-7](#), sets the clock delay to data for writes.

Offset 0x10C

Access: Read/Write

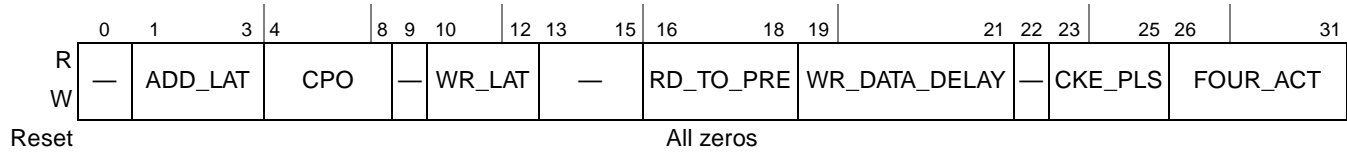


Figure 9-7. DDR SDRAM Timing Configuration 2 Register (TIMING_CFG_2)

[Table 9-11](#) describes the TIMING_CFG_2 fields.

Table 9-11. TIMING_CFG_2 Field Descriptions

Bits	Name	Description
0	—	Reserved
1–3	ADD_LAT	Additive latency. The additive latency must be set to a value less than TIMING_CFG_1[ACTTORW]. (DDR2-specific) 000 0 clocks 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 Reserved 111 Reserved
4–8	CPO ¹	MCAS-to-preamble override. Defines the number of DRAM cycles between when a read is issued and when the corresponding DQS preamble is valid for the memory controller. For these decodings, “READ_LAT” is equal to the CAS latency plus the additive latency. 00000READ_LAT + 1 01100READ_LAT + 5/2 00001Reserved 01101READ_LAT + 11/4 00010READ_LAT 01110READ_LAT + 3 00011READ_LAT + 1/4 01111READ_LAT + 13/4 00100READ_LAT + 1/2 10000READ_LAT + 7/2 00101READ_LAT + 3/4 10001READ_LAT + 15/4 00110READ_LAT + 1 10010READ_LAT + 4 00111READ_LAT + 5/4 10011READ_LAT + 17/4 01000READ_LAT + 3/2 10100READ_LAT + 9/2 01001READ_LAT + 7/4 10101READ_LAT + 19/4 01010READ_LAT + 2 10110–11111 Reserved 01011READ_LAT + 9/4
9	—	Reserved
10–12	WR_LAT	Write latency. Note that the total write latency for DDR2 is equal to WR_LAT + ADD_LAT; the write latency for DDR1 is 1. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks

9.4.1.7 DDR SDRAM Control Configuration (DDR_SDRAM_CFG)

The DDR SDRAM control configuration register, shown in [Figure 9-8](#), enables the interface logic and specifies certain operating features such as self refreshing, error checking and correcting, registered DIMMs, and dynamic power management.

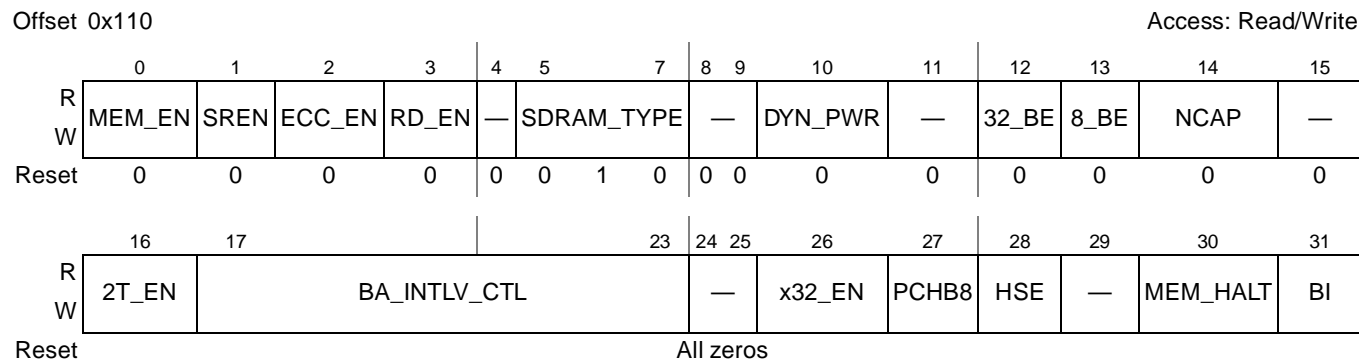


Figure 9-8. DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG)

[Table 9-12](#) describes the DDR_SDRAM_CFG fields.

Table 9-12. DDR_SDRAM_CFG Field Descriptions

Bits	Name	Description
0	MEM_EN	DDR SDRAM interface logic enable. 0 SDRAM interface logic is disabled. 1 SDRAM interface logic is enabled. Must not be set until all other memory configuration parameters have been appropriately configured by initialization code.
1	SREN	Self refresh enable (during sleep). 0 SDRAM self refresh is disabled during sleep. Whenever self-refresh is disabled, the system is responsible for preserving the integrity of SDRAM during sleep. 1 SDRAM self refresh is enabled during sleep.
2	ECC_EN	ECC enable. Note that non-correctable read errors may cause an interrupt. 0 No ECC errors are reported. No ECC interrupts are generated. 1 ECC is enabled.
3	RD_EN	Registered DIMM enable. Specifies the type of DIMM used in the system. 0 Indicates unbuffered DIMMs. 1 Indicates registered DIMMs. Note: RD_EN and 2T_EN must not both be set at the same time.
4	—	Reserved
5–7	SDRAM_TYPE	Type of SDRAM device to be used. This field is used when issuing the automatic hardware initialization sequence to DRAM through Mode Register Set and Extended Mode Register Set commands. Default value is 010 designating DDR1 SDRAM. 000–001 Reserved 010 DDR1 SDRAM 011 DDR2 SDRAM 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 9-12. DDR_SDRAM_CFG Field Descriptions (continued)

Bits	Name	Description
8–9	—	Reserved
10	DYN_PWR	Dynamic power management mode 0 Dynamic power management mode is disabled. 1 Dynamic power management mode is enabled. If there is no ongoing memory activity, the SDRAM CKE signal is negated.
11	—	Reserved
12	32_BE	32-bit bus enable. 0 64-bit bus is used. 1 32-bit bus is used.
13	8_BE	8-beat burst enable. 0 4-beat bursts are used on the DRAM interface. 1 8-beat bursts are used on the DRAM interface. Note: DDR1 (SDRAM_TYPE = 010) must use 8-beat bursts when using 32-bit bus mode (32_BE = 1) and 4-beat bursts when using 64-bit bus mode; DDR2 (SDRAM_TYPE = 011) must use 4-beat bursts, even when using 32-bit bus mode
14	NCAP	Non-concurrent auto-precharge. Some older DDR DRAMs do not support concurrent auto precharge. If one of these devices is used, then this bit needs to 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.
15	—	Reserved
16	2T_EN	Enable 2T timing. 0 1T timing is enabled. The DRAM command/address is held for only 1 cycle on the DRAM bus. 1 2T timing is enabled. The DRAM command/address are held for 2 full cycles on the DRAM bus for every DRAM transaction. However, the chip select is only held for the second cycle. Note: RD_EN and 2T_EN must not both be set at the same time.
17–23	BA_INTLV_CTL	Bank (chip select) interleaving control. Set this field only if you wish to use bank interleaving. ('x' denotes a don't care bit value. All unlisted field values are reserved.) 0000000No external memory banks are interleaved 1000000External memory banks 0 and 1 are interleaved 0100000External memory banks 2 and 3 are interleaved 1100000External memory banks 0 and 1 are interleaved together and banks 2 and 3 are interleaved together xx00100External memory banks 0 through 3 are all interleaved together
24–25	—	Reserved
26	x32_EN	x32 enable. 0 Either x8 or x16 discrete DRAM chips are used. In this mode, each data byte has a dedicated corresponding data strobe. 1 x32 discrete DRAM chips are used. In this mode, DQS0 is used to capture DQ[0:31], DQS4 is used to capture DQ[32:63] and DQS8 is used to capture ECC[0:7].
27	PCHB8	Precharge bit 8 enable. 0 MA[10] is used to indicate the auto-precharge and precharge all commands. 1 MA[8] is used to indicate the auto-precharge and precharge all commands. If x32_EN is cleared, then PCHB8 should be cleared as well.

Table 9-12. DDR_SDRAM_CFG Field Descriptions (continued)

Bits	Name	Description
28	HSE	Global half-strength override Sets I/O driver impedance to half strength. This impedance is used by the MDIC, address/command, data, and clock impedance values, but only if automatic hardware calibration is disabled and the corresponding group's software override is disabled in the DDR control driver register(s) described in Section 5.4.3.8, "DDR Control Driver Register (DDRCDR)." This bit should be cleared if using automatic hardware calibration. 0 I/O driver impedance is configured to full strength. 1 I/O driver impedance is configured to half strength.
29	—	Reserved
30	MEM_HALT	DDR memory controller halt. When this bit is set, the memory controller does not accept any new data read/write transactions to DDR SDRAM until the bit is cleared again. This can be used when bypassing initialization and forcing MODE REGISTER SET commands through software. 0 DDR controller accepts new transactions. 1 DDR controller finishes any remaining transactions, and then it remains halted until this bit is cleared by software.
31	BI	Bypass initialization 0 DDR controller cycles through initialization routine based on SDRAM_TYPE 1 Initialization routine is bypassed. Software is responsible for initializing memory through DDR_SDRAM_MODE2 register. If software is initializing memory, then the MEM_HALT bit can be set to prevent the DDR controller from issuing transactions during the initialization sequence. Note that the DDR controller does not issue a DLL reset to the DRAMs when bypassing the initialization routine, regardless of the value of DDR_SDRAM_CFG[DLL_RST_DIS]. If a DLL reset is required, then the controller should be forced to enter and exit self-refresh after the controller is enabled. See Section 9.4.1.15, "DDR Initialization Address (DDR_INIT_ADDR)," for details on avoiding ECC errors in this mode.

9.4.1.8 DDR SDRAM Control Configuration 2 (DDR_SDRAM_CFG_2)

The DDR SDRAM control configuration register 2, shown in [Figure 9-9](#), provides more control configuration for the DDR controller.

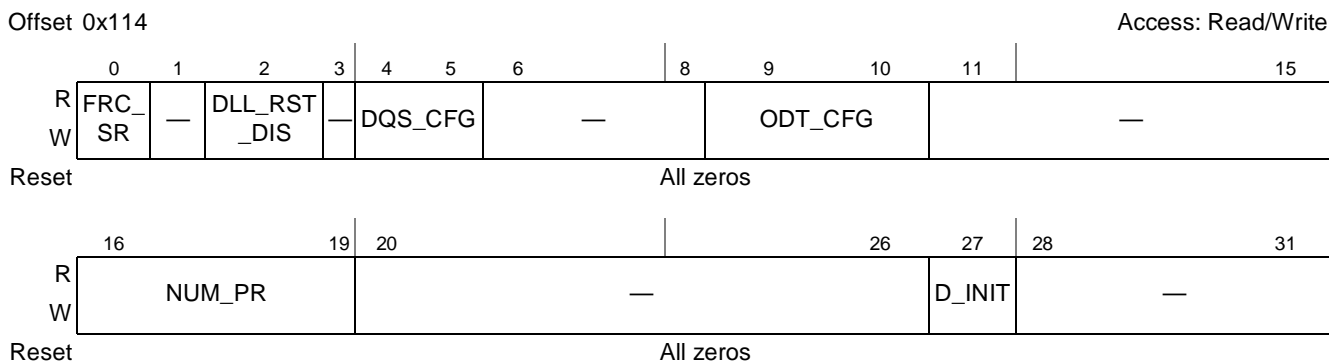


Figure 9-9. DDR SDRAM Control Configuration Register 2 (DDR_SDRAM_CFG_2)

Table 9-13 describes the DDR_SDRAM_CFG_2 fields.

Table 9-13. DDR_SDRAM_CFG_2 Field Descriptions

Bits	Name	Description
0	FRC_SR	Force self-refresh 0 DDR controller operates in normal mode. 1 DDR controller enters self-refresh mode.
1	—	Reserved. Should be cleared.
2	DLL_RST_DIS	DLL reset disable The DDR controller typically issues a DLL reset to the DRAMs when exiting self refresh. However, this function may be disabled by setting this bit during initialization. 0 DDR controller issues a DLL reset to the DRAMs when exiting self refresh. 1 DDR controller does not issue a DLL reset to the DRAMs when exiting self refresh.
3	—	Reserved
4–5	DQS_CFG	DQS configuration 00 Only true DQS signals are used. 01 Reserved 10 Reserved 11 Reserved
6–8	—	Reserved
9–10	ODT_CFG	ODT configuration This field defines how ODT is driven to the on-chip IOs. See Section 5.4.3.8, “DDR Control Driver Register (DDRCDR),” which defines the termination value that is used. (DDR2-specific, must be cleared for DDR1) 00 Never assert ODT to internal IOs 01 Assert ODT to internal IOs only during writes to DRAM 10 Assert ODT to internal IOs only during reads to DRAM 11 Always keep ODT asserted to internal IOs
11–15	—	Reserved.
16–19	NUM_PR	Number of posted refreshes This determines how many posted refreshes, if any, can be issued at one time. Note that if posted refreshes are used, then this field, along with DDR_SDRAM_INTERVAL[REFINT], must be programmed such that the maximum t_{ras} specification cannot be violated. For example, some DDR1 SDRAMs are not able to use more than 3 posted refreshes because the required refresh interval could then exceed the maximum constraint for t_{ras} . 0000 Reserved 0001 1 refresh is issued at a time 0010 2 refreshes is issued at a time 0011 3 refreshes is issued at a time ... 1000 8 refreshes is issued at a time 1001–1111 Reserved
20–24	—	Reserved
20–26	—	Reserved, should be cleared.

Table 9-13. DDR_SDRAM_CFG_2 Field Descriptions (continued)

Bits	Name	Description
27	D_INIT	<p>DRAM data initialization</p> <p>This bit is set by software, and it is cleared by hardware. If software sets this bit before the memory controller is enabled, the controller automatically initializes DRAM after it is enabled. This bit is automatically cleared by hardware once the initialization is completed. This data initialization bit should only be set when the controller is idle.</p> <p>0 There is not data initialization in progress, and no data initialization is scheduled</p> <p>1 The memory controller initializes memory once it is enabled. This bit remains asserted until the initialization is complete. The value in DDR_DATA_INIT register is used to initialize memory.</p>
28–31	—	Reserved

9.4.1.9 DDR SDRAM Mode Configuration (DDR_SDRAM_MODE)

The DDR SDRAM mode configuration register, shown in [Figure 9-10](#), sets the values loaded into the DDR’s mode registers.



Figure 9-10. DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE)

[Table 9-14](#) describes the DDR_SDRAM_MODE fields.

Table 9-14. DDR_SDRAM_MODE Field Descriptions

Bits	Name	Description
0–15	ESDMODE	<p>Extended SDRAM mode</p> <p>Specifies the initial value loaded into the DDR SDRAM extended mode register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer.</p> <p>When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb of ESDMODE, which, in the big-endian convention shown in Figure 9-10, corresponds to ESDMODE[15]. The msb of the SDRAM extended mode register value must be stored at ESDMODE[0].</p>
16–31	SDMODE	<p>SDRAM mode</p> <p>Specifies the initial value loaded into the DDR SDRAM mode register. The range of legal values is specified by the DDR SDRAM manufacturer.</p> <p>When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of SDMODE, which, in the big-endian convention shown in Figure 9-10, corresponds to SDMODE[15]. The msb of the SDRAM mode register value must be stored at SDMODE[0]. Because the memory controller forces SDMODE[7] to certain values depending on the state of the initialization sequence, (for resetting the SDRAM’s DLL) the corresponding bits of this field are ignored by the memory controller. Note that SDMODE[7] is mapped to MA[8].</p>

9.4.1.10 DDR SDRAM Mode 2 Configuration (DDR_SDRAM_MODE_2)

The DDR SDRAM mode 2 configuration register, shown in [Figure 9-11](#), sets the values loaded into the DDR's extended mode 2 and 3 registers (for DDR2).

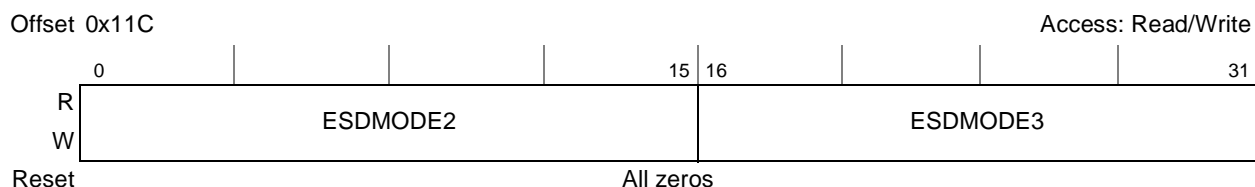


Figure 9-11. DDR SDRAM Mode 2 Configuration Register (DDR_SDRAM_MODE_2)

[Table 9-15](#) describes the DDR_SDRAM_MODE_2 fields.

Table 9-15. DDR_SDRAM_MODE_2 Field Descriptions

Bits	Name	Description
0–15	ESDMODE2	Extended SDRAM mode 2 Specifies the initial value loaded into the DDR SDRAM extended 2 mode register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb bit of ESDMODE2, which, in the big-endian convention shown in Figure 9-11 , corresponds to ESDMODE2[15]. The msb of the SDRAM extended mode 2 register value must be stored at ESDMODE2[0].
16–31	ESDMODE3	Extended SDRAM mode 3 Specifies the initial value loaded into the DDR SDRAM extended 3 mode register. The range of legal values of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of ESDMODE3, which, in the big-endian convention shown in Figure 9-11 , corresponds to ESDMODE3[15]. The msb of the SDRAM extended mode 3 register value must be stored at ESDMODE3[0].

9.4.1.11 DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL)

The DDR SDRAM mode control register, shown in [Figure 9-12](#), allows the user to carry out the following tasks:

- Issue a mode register set command to a particular chip select
- Issue an immediate refresh to a particular chip select
- Issue an immediate precharge or precharge all command to a particular chip select
- Force the CKE signals to a specific value

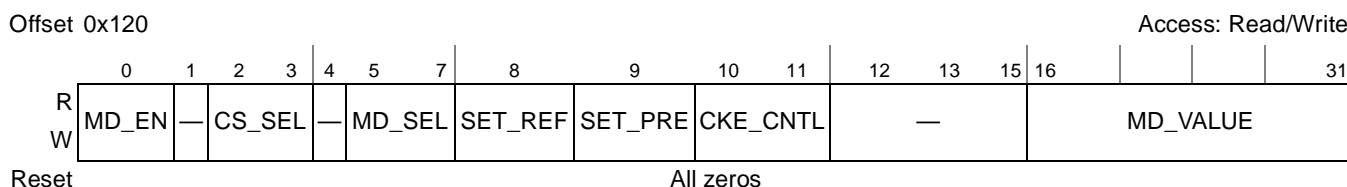


Figure 9-12. DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL)

Table 9-16 describes the fields of this register. Table 9-17 shows the user how to set the fields of this register to accomplish the above tasks.

NOTE

Note that MD_EN, SET_REF, and SET_PRE are mutually exclusive; only one of these fields can be set at a time.

Table 9-16. DDR_SDRAM_MD_CNTL Field Descriptions

Bits	Name	Description
0	MD_EN	Mode enable Setting this bit specifies that valid data in MD_VALUE is ready to be written to DRAM as one of the following commands: <ul style="list-style-type: none"> • MODE REGISTER SET • EXTENDED MODE REGISTER SET • EXTENDED MODE REGISTER SET 2 • EXTENDED MODE REGISTER SET 3 The specific command to be executed is selected by setting MD_SEL. In addition, the chip select must be chosen by setting CS_SEL. MD_EN is set by software and cleared by hardware once the command has been issued. 0 Indicates that no mode register set command needs to be issued. 1 Indicates that valid data contained in the register is ready to be issued as a mode register set command.
1	—	Reserved
2-3	CS_SEL	Select chip select Specifies the chip select that is driven active due to any command forced by software in DDR_SDRAM_MD_CNTL. <ul style="list-style-type: none"> 00 Chip select 0 is active 01 Chip select 1 is active 10 Chip select 2 is active 11 Chip select 3 is active
4	—	Reserved
5-7	MD_SEL	Mode register select MD_SEL specifies one of the following: <ul style="list-style-type: none"> • During a mode select command, selects the SDRAM mode register to be changed • During a precharge command, selects the SDRAM logical bank to be precharged. A precharge all command ignores this field. • During a refresh command, this field is ignored. Note that MD_SEL contains the value that is presented onto the memory bank address pins (MBA _n) of the DDR controller. <ul style="list-style-type: none"> 000 MR 001 EMR 010 EMR2 011 EMR3
8	SET_REF	Set refresh Forces an immediate refresh to be issued to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. This bit is set by software and cleared by hardware once the command has been issued. 0 Indicates that no refresh command needs to be issued. 1 Indicates that a refresh command is ready to be issued.

Table 9-16. DDR_SDRAM_MD_CNTL Field Descriptions (continued)

Bits	Name	Description
9	SET_PRE	Set precharge Forces a precharge or precharge all to be issued to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. This bit is set by software and cleared by hardware once the command has been issued. 0 Indicates that no precharge all command needs to be issued. 1 Indicates that a precharge all command is ready to be issued.
10–11	CKE_CNTL	Clock enable control Allows software to globally clear or set all CKE signals issued to DRAM. Once software has forced the value driven on CKE, that value continues to be forced until software clears the CKE_CNTL bits. At that time, the DDR controller continues to drive the CKE signals to the same value forced by software until another event causes the CKE signals to change (such as, self refresh entry/exit, power down entry/exit). 00 CKE signals are not forced by software. 01 CKE signals are forced to a low value by software. 10 CKE signals are forced to a high value by software. 11 Reserved
12–15	—	Reserved
16–31	MD_VALUE	Mode register value This field, which specifies the value that is presented on the memory address pins of the DDR controller during a mode register set command, is significant only when this register is used to issue a mode register set command or a precharge or precharge all command. For a mode register set command, this field contains the data to be written to the selected mode register. For a precharge command, only bit five is significant: 0 Issue a precharge command; MD_SEL selects the logical bank to be precharged 1 Issue a precharge all command; all logical banks are precharged

Table 9-17 shows how DDR_SDRAM_MD_CNTL fields should be set for each of the tasks described above.

Table 9-17. Settings of DDR_SDRAM_MD_CNTL Fields

Field	Mode Register Set	Refresh	Precharge	Clock Enable Signals Control
MD_EN	1	0	0	—
SET_REF	0	1	0	—
SET_PRE	0	0	1	—
CS_SEL	Chooses chip select (CS)			—
MD_SEL	Select mode register. See Table 9-16.	—	Selects logical bank	—
MD_VALUE	Value written to mode register	—	Only bit five is significant. See Table 9-16.	—
CKE_CNTL	0	0	0	See Table 9-16.

9.4.1.12 DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)

The DDR SDRAM interval configuration register, shown in [Figure 9-13](#), sets the number of DRAM clock cycles between bank refreshes issued to the DDR SDRAMs. In addition, the number of DRAM cycles that a page is maintained after it is accessed is provided here.



Figure 9-13. DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL)

[Table 9-18](#) describes the DDR_SDRAM_INTERVAL fields.

Table 9-18. DDR_SDRAM_INTERVAL Field Descriptions

Bits	Name	Description
0–15	REFINT	Refresh interval Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR_SDRAM_CFG_2[NUM_PR], some number of rows are refreshed in each DDR SDRAM physical bank during each refresh cycle. The value for REFINT depends on the specific SDRAMs used and the interface clock frequency. Refreshes are not issued when the REFINT is set to all 0s.
16–17	—	Reserved
18–31	BSTOPRE	Precharge interval Sets the duration (in memory bus clocks) that a page is retained after a DDR SDRAM access. If BSTOPRE is zero, the DDR memory controller uses auto-precharge read and write commands rather than operating in page mode. This is called global auto-precharge mode.

9.4.1.13 DDR SDRAM Data Initialization (DDR_DATA_INIT)

The DDR SDRAM data initialization register, shown in [Figure 9-14](#), provides the value that is used to initialize memory if DDR_SDRAM_CFG2[D_INIT] is set.

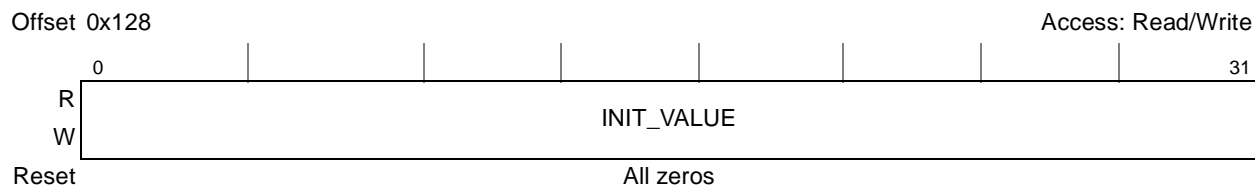


Figure 9-14. DDR SDRAM Data Initialization Configuration Register (DDR_DATA_INIT)

[Table 9-19](#) describes the DDR_DATA_INIT fields.

Table 9-19. DDR_DATA_INIT Field Descriptions

Bits	Name	Description
0–31	INIT_VALUE	Initialization value. Represents the value that DRAM is initialized with if DDR_SDRAM_CFG2[D_INIT] is set.

9.4.1.14 DDR SDRAM Clock Control (DDR_SDRAM_CLK_CNTL)

The DDR SDRAM clock control configuration register, shown in [Figure 9-15](#), provides a 1/4-cycle clock adjustment.

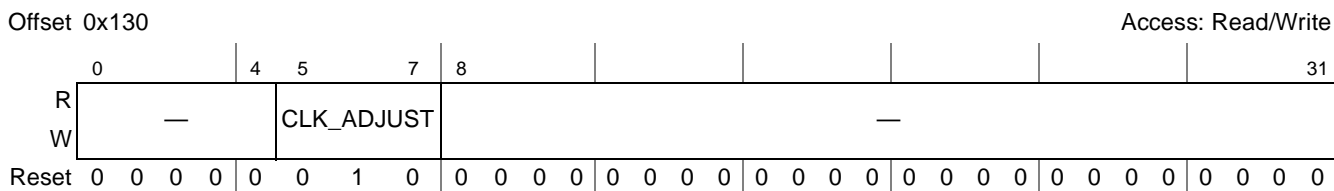


Figure 9-15. DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL)

[Table 9-20](#) describes the DDR_SDRAM_CLK_CNTL fields.

Table 9-20. DDR_SDRAM_CLK_CNTL Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	CLK_ADJUST	Clock adjust. 000 Clock is launched aligned with address/command 001 Clock is launched 1/4 applied cycle after address/command 010 Clock is launched 1/2 applied cycle after address/command 011 Clock is launched 3/4 applied cycle after address/command 100 Clock is launched 1 applied cycle after address/command 101–111 Reserved
8	—	Reserved, should be cleared.
9–31	—	Reserved

9.4.1.15 DDR Initialization Address (DDR_INIT_ADDR)

The DDR SDRAM initialization address register, shown in [Figure 9-16](#), provides the address that is used for the automatic CAS to preamble calibration after POR.

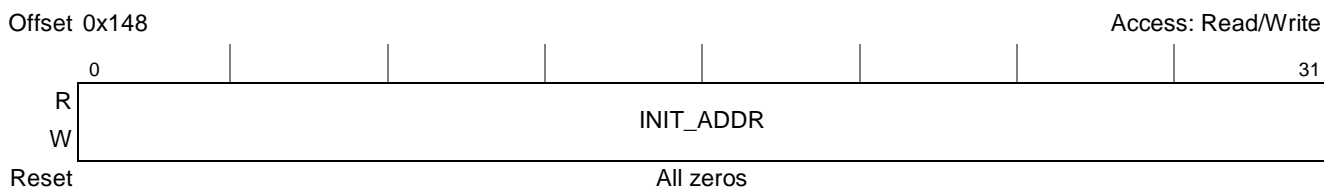


Figure 9-16. DDR Initialization Address Configuration Register (DDR_INIT_ADDR)

[Table 9-21](#) describes the DDR_INIT_ADDR fields.

Table 9-21. DDR_INIT_ADDR Field Descriptions

Bits	Name	Description
0–31	INIT_ADDR	Initialization address. Represents the address that is used for the automatic CAS to preamble calibration at POR.

9.4.1.16 DDR IP Block Revision 1 (DDR_IP_REV1)

The DDR IP block revision 1 register, shown in Figure 9-17, provides read-only fields with the IP block ID, along with major and minor revision information.

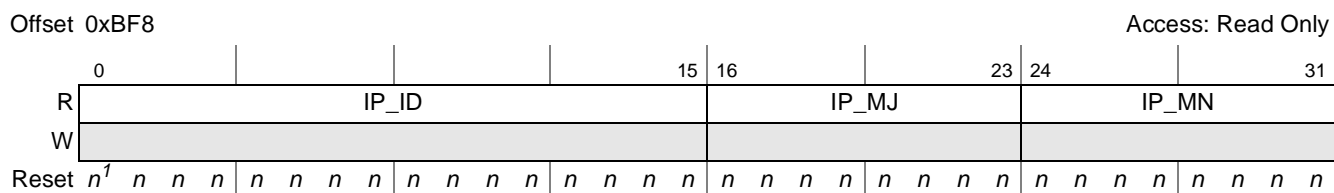


Figure 9-17. DDR IP Block Revision 1 (DDR_IP_REV1)

¹ For reset values, see Table 9-22.

Table 9-22 describes the DDR_IP_REV1 fields.

Table 9-22. DDR_IP_REV1 Field Descriptions

Bits	Name	Description
0–15	IP_ID	IP block ID. For the DDR controller, this value is 0x0002.

9.4.1.17 DDR IP Block Revision 2 (DDR_IP_REV2)

The DDR IP block revision 2 register, shown in Figure 9-18, provides read-only fields with the IP block integration and configuration options.

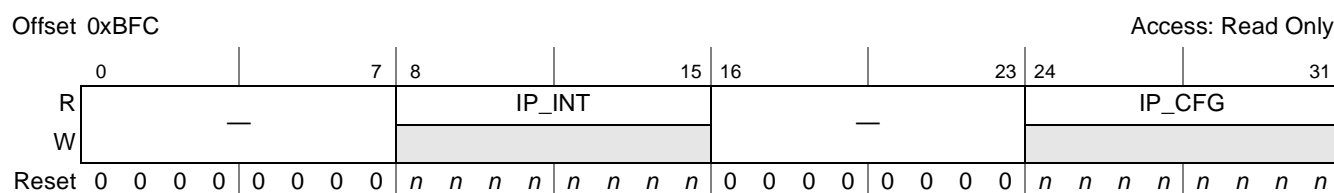


Figure 9-18. DDR IP Block Revision 2 (DDR_IP_REV2)

Table 9-23 describes the DDR_IP_REV2 fields.

Table 9-23. DDR_IP_REV2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IP_INT	IP block integration options
16–23	—	Reserved
24–31	IP_CFG	IP block configuration options

9.4.1.18 Memory Data Path Error Injection Mask High (DATA_ERR_INJECT_HI)

The memory data path error injection mask high register is shown in [Figure 9-19](#).

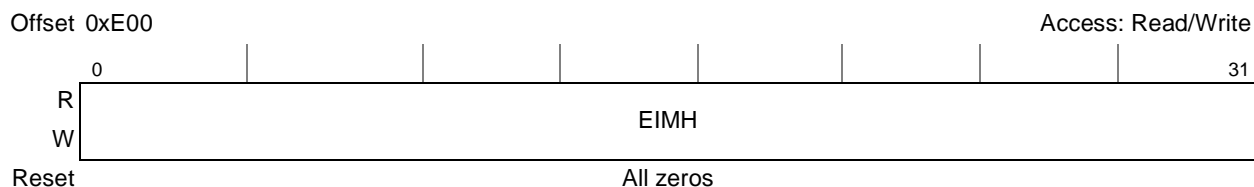


Figure 9-19. Memory Data Path Error Injection Mask High Register (DATA_ERR_INJECT_HI)

[Table 9-24](#) describes the DATA_ERR_INJECT_HI fields.

Table 9-24. DATA_ERR_INJECT_HI Field Descriptions

Bits	Name	Description
0–31	EIMH	Error injection mask high data path Used to test ECC by forcing errors on the high word of the data path. Setting a bit causes the corresponding data path bit to be inverted on memory bus writes.

9.4.1.19 Memory Data Path Error Injection Mask Low (DATA_ERR_INJECT_LO)

The memory data path error injection mask low register is shown in [Figure 9-20](#).

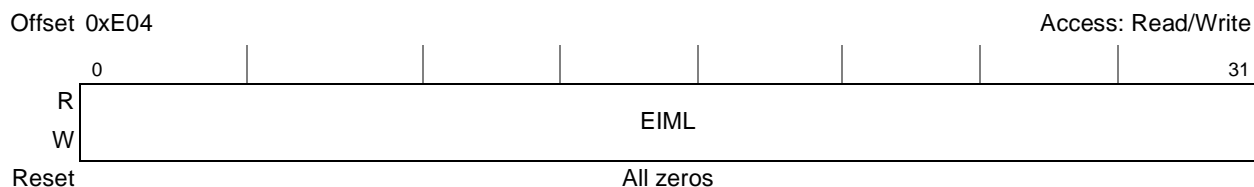


Figure 9-20. Memory Data Path Error Injection Mask Low Register (DATA_ERR_INJECT_LO)

[Table 9-25](#) describes the DATA_ERR_INJECT_LO fields.

Table 9-25. DATA_ERR_INJECT_LO Field Descriptions

Bits	Name	Description
0–31	EIML	Error injection mask low data path Used to test ECC by forcing errors on the low word of the data path. Setting a bit causes the corresponding data path bit to be inverted on memory bus writes.

9.4.1.22 Memory Data Path Read Capture Low (CAPTURE_DATA_LO)

The memory data path read capture low register, shown in [Figure 9-23](#), stores the low word of the read data path during error capture.

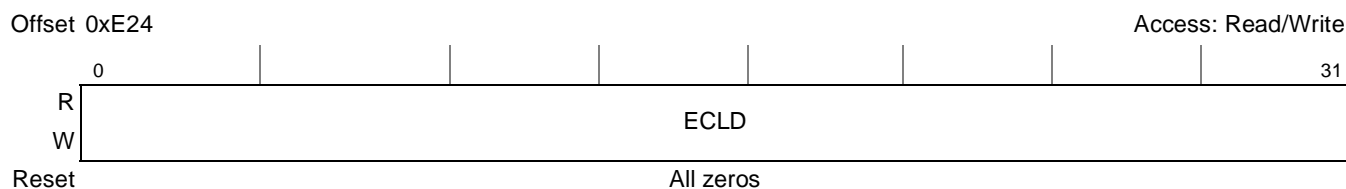


Figure 9-23. Memory Data Path Read Capture Low Register (CAPTURE_DATA_LO)

[Table 9-28](#) describes the CAPTURE_DATA_LO fields.

Table 9-28. CAPTURE_DATA_LO Field Descriptions

Bits	Name	Description
0–31	ECLD	Error capture low data path. Captures the low word of the data path when errors are detected.

9.4.1.23 Memory Data Path Read Capture ECC (CAPTURE_ECC)

The memory data path read capture ECC register, shown in [Figure 9-24](#), stores the ECC syndrome bits that were on the data bus when an error was detected.

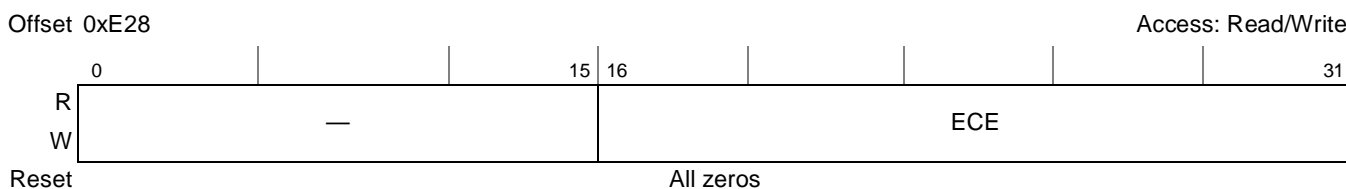


Figure 9-24. Memory Data Path Read Capture ECC Register (CAPTURE_ECC)

[Table 9-29](#) describes the CAPTURE_ECC fields.

Table 9-29. CAPTURE_ECC Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	ECE	Error capture ECC. Captures the ECC bits on the data path whenever errors are detected. 16:23—8-bit ECC code for 1st 32 bits 24:31—8-bit ECC code for 2nd 32 bits Note: In 64-bit mode, only 24:31 should be used, although 16:23 shows the 8-bit ECC code replicated.

9.4.1.24 Memory Error Detect (ERR_DETECT)

The memory error detect register stores the detection bits for multiple memory errors, single- and multiple-bit ECC errors, and memory select errors. It is a read/write register. A bit can be cleared by writing a one to the bit. System software can determine the type of memory error by examining the

Table 9-33. CAPTURE_ATTRIBUTES Field Descriptions (continued)

Bits	Name	Description
11–15	TSRC	Transaction source for the error. 00000 e300 core data transaction 00001 Reserved 00010 e300 core instruction fetch 00011–00111 Reserved 01000 Encryption core 01001 I ² C (boot sequencer) 01010 JTAG 01011 Reserved 01100 Reserved 01101 PCI 01110 Reserved 01111 DMA 10000–11111 QUICC Engine block as follows: 100xy are used by the QUICC Engine block as follows: x = TSRC[3] Least significant bit of SNUM y = TSCRC[4] CETM bit from RBMR/TBMR registers. RBMR/TBMR registers are described in protocol chapters. 101xx Reserved 11xxx Reserved Note: TSRC reflects the source of transaction and is used for debug purposes.
16–17	—	Reserved
18–19	TTYP	Transaction type for the error. 00 Reserved 01 Write 10 Read 11 Read-modify-write
20–30	—	Reserved
31	VLD	Valid. Set as soon as valid information is captured in the error capture registers.

9.4.1.28 Memory Error Address Capture (CAPTURE_ADDRESS)

The memory error address capture register, shown in [Figure 9-29](#), holds the 32 lsbs of a transaction when a DDR ECC error is detected.

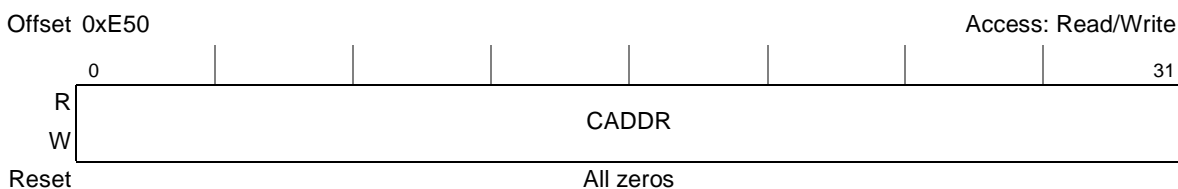


Figure 9-29. Memory Error Address Capture Register (CAPTURE_ADDRESS)

[Table 9-34](#) describes the CAPTURE_ADDRESS fields.

Table 9-34. CAPTURE_ADDRESS Field Descriptions

Bits	Name	Description
0–31	CADDR	Captured address. Captures the 32 lsbs of the transaction address when an error is detected.

9.4.1.29 Single-Bit ECC Memory Error Management (ERR_SBE)

The single-bit ECC memory error management register, shown in [Figure 9-30](#), 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.

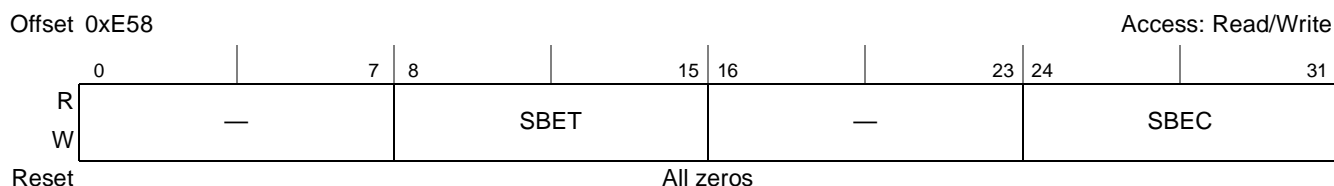


Figure 9-30. Single-Bit ECC Memory Error Management Register (ERR_SBE)

[Table 9-35](#) describes the ERR_SBE fields.

Table 9-35. ERR_SBE Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	SBET	Single-bit error threshold. Establishes the number of single-bit errors that must be detected before an error condition is reported.
16–23	—	Reserved
24–31	SBEC	Single-bit error counter. Indicates the number of single-bit errors detected and corrected since the last error report. If single-bit error reporting is enabled, an error is reported and an interrupt is generated when this value equals SBET. SBEC is automatically cleared when the threshold value is reached.

9.5 Functional Description

The DDR SDRAM controller controls processor and I/O interactions with system memory. It provides support for JEDEC-compliant DDR2 and DDR SDRAMs. The memory system allows a wide range of memory devices to be mapped to any arbitrary chip select, and support is provided for registered DIMMs and unbuffered DIMMs. However, registered DIMMs cannot be mixed with unbuffered DIMMs.

[Figure 9-31](#) is a high-level block diagram of the DDR memory controller. Requests are received from the internal mastering device and the address is decoded to generate the physical bank, logical bank, row, and column addresses. The transaction is compared with values in the row open table to determine if the address maps to an open page. If the transaction does not map to an open page, an active command is issued.

The memory interface supports as many as four physical banks of 64-/72-bit wide or 32-/40-bit wide memory. Bank sizes up to 2 Gbits (maximum total physical memory size of 4 Gbytes) are supported, providing up to a maximum of 4 Gbits of DDR main memory per chip select.

Programmable parameters allow for a variety of memory organizations and timings. Optional error checking and correcting (ECC) protection is provided for the DDR SDRAM data bus. Using ECC, the DDR memory controller detects and corrects all single-bit errors within the 64- or 32-bit data bus, detects

all double-bit errors within the 64- or 32-bit data bus, and detects all errors within a nibble. The controller allows as many as 32 pages to be open simultaneously. The amount of time (in clock cycles) the pages remain open is programmable with `DDR_SDRAM_INTERVAL[BSTOPRE]`.

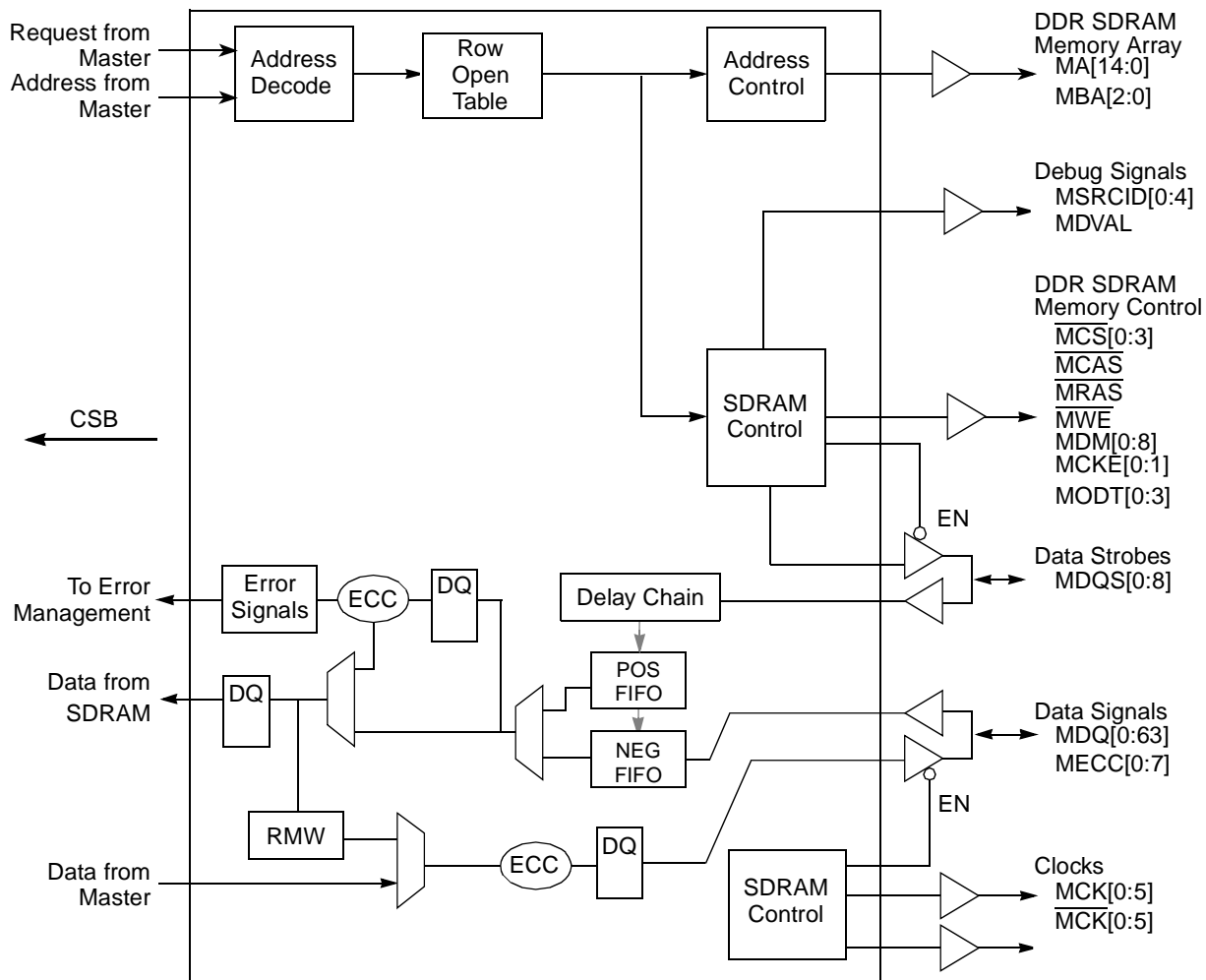


Figure 9-31. DDR Memory Controller Block Diagram

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

The data interface is source synchronous, meaning whatever sources the data also provides a clocking signal to synchronize data reception. These bidirectional data strobes (MDQS[0:8]) are inputs to the controller during reads and outputs during writes. The DDR SDRAM specification requires the data strobe signals to be centered within the data tenure during writes and to be offset by the controller to the center of the data tenure during reads. This delay is implemented in the controller for both reads and writes.

When ECC is enabled, 1 clock cycle is added to the read path to check ECC and correct single-bit errors. ECC generation does not add a cycle to the write path.

The address and command interface is also source synchronous, although 1/8 cycle adjustments are provided for adjusting the clock alignment.

Figure 9-32 shows an example DDR SDRAM configuration with four logical banks.

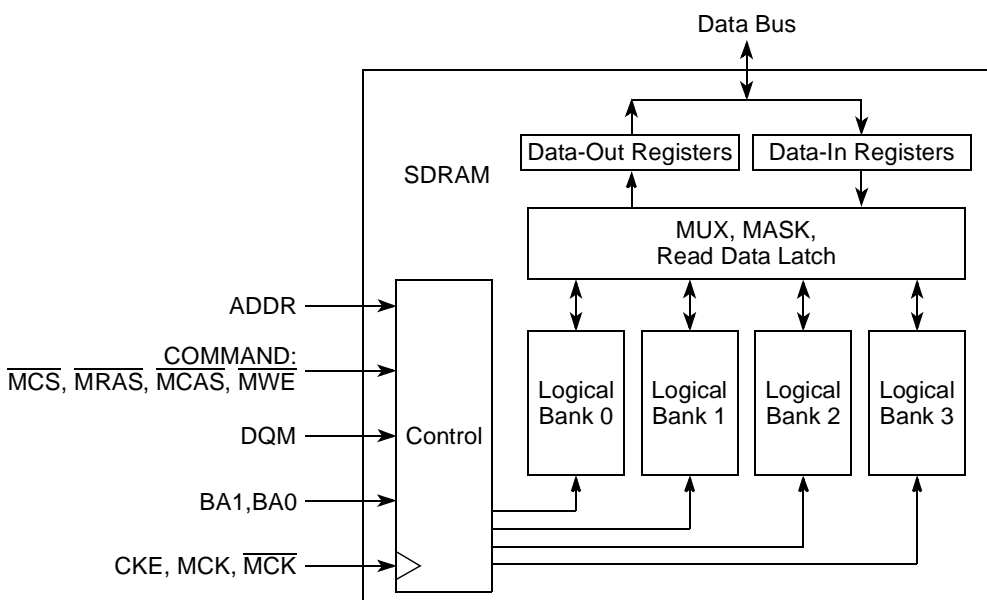


Figure 9-32. Typical Dual Data Rate SDRAM Internal Organization

Figure 9-33 shows some typical signal connections.

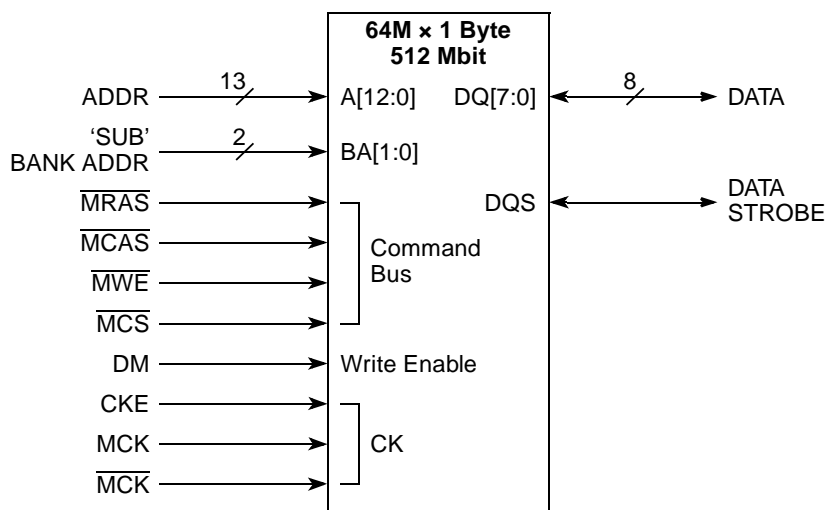
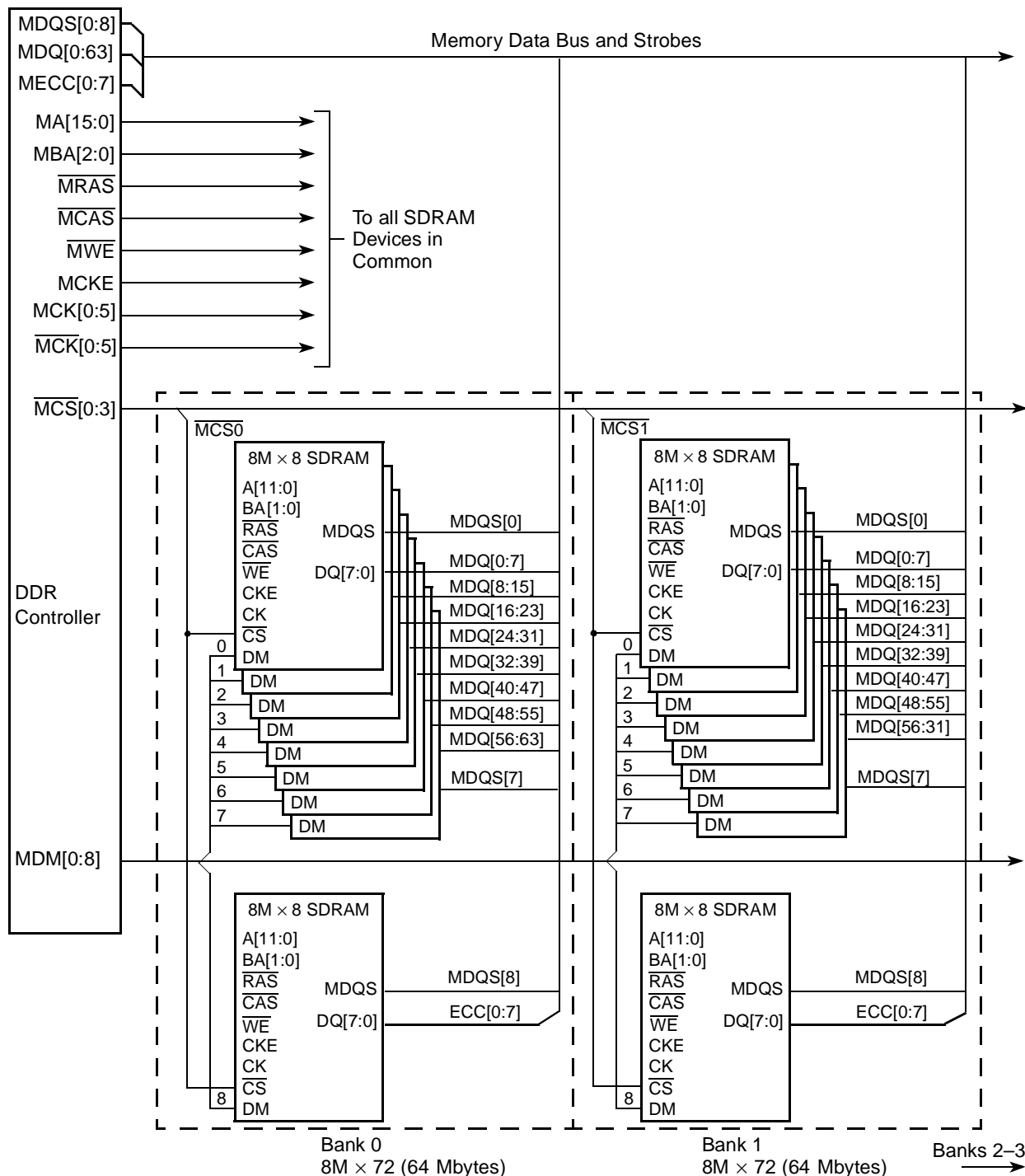


Figure 9-33. Typical DDR SDRAM Interface Signals

Figure 9-34 shows an example DDR SDRAM configuration with four physical banks each comprised of nine 8M x 8 DDR modules for a total of 256 Mbytes of system memory. One of the nine modules is used for the memory's ECC checking function. Certain address and control lines may require buffering.

DDR Memory Controller

Analysis of the device's AC timing specifications, desired memory operating frequency, capacitive loads, and board routing loads can assist the system designer in deciding signal buffering requirements. The DDR memory controller drives 15 address pins, but in this example the DDR SDRAM devices use only 12 bits.



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

Figure 9-34. Example 256-Mbyte DDR SDRAM Configuration With ECC

Section 9.5.12, “Error Management,” explains how the DDR memory controller handles errors.

9.5.1 DDR SDRAM Interface Operation

The DDR memory controller supports many different DDR SDRAM configurations. SDRAMs with different sizes can be used in the same system. Fifteen multiplexed address signals and three logical bank select signals support device densities from 64 Mbits to 4 Gbits. Four chip select (\overline{CS}) signals support up to two DIMMs of memory. The DDR SDRAM physical banks can be built from standard memory modules or directly-attached memory devices. The data path to individual physical banks is 64 or 32 bits wide, 72 or 40 bits with ECC. The DDR memory controller supports physical bank sizes from 16 Mbytes to 4 Gbytes. The physical banks can be constructed using $\times 8$, $\times 16$, or $\times 32$ memory devices. The memory technologies supported are 64 Mbits, 128 Mbits, 256 Mbits, 512 Mbits, 1 Gbit, 2 Gbits, and 4 Gbits. Nine data qualifier (DQM) signals provide byte selection for memory accesses.

NOTE

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

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

Table 9-36 shows the DDR memory controller’s relationships between data byte lane 0–7, MDM[0:7], MDQS[0:7], and MDQ[0:63] when DDR SDRAM memories are used with $\times 8$ or $\times 16$ devices.

Table 9-36. Byte Lane to Data Relationship

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

9.5.1.1 Supported DDR SDRAM Organizations

Although the DDR memory controller multiplexes row and column address bits onto 15 memory address signals and 3 logical bank select signals, a physical bank may be implemented with memory devices requiring fewer than 31 address bits. The physical bank may be configured to provide from 12 to 15 row address bits, plus 2 or 3 logical bank-select bits and from 8–11 column address bits.

Table 9-37 and Table 9-38 describe DDR SDRAM device configurations supported by the DDR memory controller.

NOTE

DDR SDRAM is limited to 30 total address bits.

Table 9-37. Supported DDR1 SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column x Sub-Bank Bits	64-Bit Bank Size	Four Banks of Memory
64 Mbits	8 Mbits x 8	12 x 9 x 2	64 Mbytes	256 Mbytes
64 Mbits ¹	4 Mbits x 16	12 x 8 x 2	32 Mbytes	128 Mbytes
128 Mbits	16 Mbits x 8	12 x 10 x 2	128 Mbytes	512 Mbytes
128 Mbits	8 Mbits x 16	12 x 9 x 2	64 Mbytes	256 Mbytes
256 Mbits	32 Mbits x 8	13 x 10 x 2	256 Mbytes	1 Gbyte
256 Mbits	16 Mbits x 16	13 x 9 x 2	128 Mbytes	512 Mbytes
512 Mbits	64 Mbits x 8	13 x 11 x 2	512 Mbytes	2 Gbytes
512 Mbits	32 Mbits x 16	13 x 10 x 2	256 Mbytes	1 Gbyte
1 Gbit	128 Mbits x 8	14 x 11 x 2	1 Gbyte	4 Gbytes
1 Gbit	64 Mbits x 16	14 x 10 x 2	512 Mbytes	2 Gbytes
2 Gbits	256 Mbits x 8	15 x 11 x 2	2 Gbytes	8 Gbytes (split into two banks)
2 Gbits	128 Mbits x 16	15 x 10 x 2	1 Gbyte	4 Gbytes

¹ This configuration is not supported in 16-bit bus mode.

If a transaction request is issued to the DDR memory controller and the address does not lie within any of **Table 9-38. Supported DDR2 SDRAM Device Configurations**

SDRAM Device	Device Configuration	Row x Column x Sub-Bank Bits	64-Bit Bank Size	Four Banks of Memory
256 Mbits	32 Mbits x 8	13 x 10 x 2	256 Mbytes	1 Gbyte
256 Mbits	16 Mbits x 16	13 x 9 x 2	128 Mbytes	512 Mbytes
512 Mbits	64 Mbits x 8	14 x 10 x 2	512 Mbytes	2 Gbytes
512 Mbits	32 Mbits x 16	13 x 10 x 2	256 Mbytes	1 Gbyte
1 Gbit	128 Mbits x 8	14 x 10 x 3	1 Gbyte	4 Gbytes
1 Gbit	64 Mbits x 16	13 x 10 x 3	512 Mbytes	2 Gbytes
2 Gbits	256 Mbits x 8	15 x 10 x 3	2 Gbytes	8 Gbytes (split into two banks)
2 Gbits	128 Mbits x 16	14 x 10 x 3	1 Gbyte	4 Gbytes
4 Gbits	256 Mbits x 16	15 x 10 x 3	2 Gbytes	8 Gbytes (split into two banks)

the programmed address ranges for an enabled chip select, a memory select error is flagged. Errors are described in detail in [Section 9.5.12, “Error Management.”](#)

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

9.5.2 DDR SDRAM Address Multiplexing

[Table 9-39](#), [Table 9-40](#), [Table 9-41](#), [Table 9-42](#) show the address bit encodings for each DDR SDRAM configuration. The address presented at the memory controller signals MA[14:0] use MA[14] as the msb and MA[0] as the lsb. Also, MA[10] is used as the auto-precharge bit in DDR1/DDR2 modes for reads and writes, so the column address can never use MA[10].

Table 9-39. DDR1 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled

Row x Col	msb	Address from Core Master																												lsb		
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27		28	29-31
15 x 11 x 2	MRAS		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																	1	0													
	\overline{MCAS}																			11	9	8	7	6	5	4	3	2	1	0		
15 x 10 x 2	MRAS			14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		1	0												
	\overline{MCAS}																				9	8	7	6	5	4	3	2	1	0		

Table 9-39. DDR1 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled (continued)

Row x Col	msb	Address from Core Master																												lsb			
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27		28	29-31	
14 x 11 x 2	$\overline{\text{MRAS}}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																	1	0														
	$\overline{\text{MCAS}}$																			11	9	8	7	6	5	4	3	2	1	0			
14 x 10 x 2	$\overline{\text{MRAS}}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																		1	0													
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0			
13 x 11 x 2	$\overline{\text{MRAS}}$					12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																		1	0													
	$\overline{\text{MCAS}}$																				11	9	8	7	6	5	4	3	2	1	0		
13 x 10 x 2	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																			1	0												
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0		
13 x 9 x 2	$\overline{\text{MRAS}}$							12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																				1	0											
	$\overline{\text{MCAS}}$																						8	7	6	5	4	3	2	1	0		
12 x 10 x 2	$\overline{\text{MRAS}}$						11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																				1	0											
	$\overline{\text{MCAS}}$																						9	8	7	6	5	4	3	2	1	0	
12 x 9 x 2	$\overline{\text{MRAS}}$							11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																					1	0										
	$\overline{\text{MCAS}}$																							8	7	6	5	4	3	2	1	0	
12 x 8 x 2	$\overline{\text{MRAS}}$								11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																						1	0									
	$\overline{\text{MCAS}}$																								7	6	5	4	3	2	1	0	

Table 9-40. DDR1 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled

Row x Col	msb	Address from Core Master																													lsb		
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28		29	30-31
15 x 11 x 2	$\overline{\text{MRAS}}$			14	13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																			1	0												
	$\overline{\text{MCAS}}$																					11	9	8	7	6	5	4	3	2	1	0	
15 x 10 x 2	$\overline{\text{MRAS}}$				14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																					1	0										
	$\overline{\text{MCAS}}$																							9	8	7	6	5	4	3	2	1	0

Table 9-40. DDR1 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled (continued)

Row x Col	msb	Address from Core Master																												lsb				
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27		28	29	30-31	
14 x 11 x 2	MRAS				13	12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																		1	0														
	MCAS																					11	9	8	7	6	5	4	3	2	1	0		
14 x 10 x 2	MRAS				13	12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																			1	0													
	MCAS																					9	8	7	6	5	4	3	2	1	0			
13 x 11 x 2	MRAS				12	11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																			1	0													
	MCAS																					11	9	8	7	6	5	4	3	2	1	0		
13 x 10 x 2	MRAS				12	11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																				1	0												
	MCAS																						9	8	7	6	5	4	3	2	1	0		
13 x 9 x 2	MRAS				12	11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																					1	0											
	MCAS																							8	7	6	5	4	3	2	1	0		
12 x 10 x 2	MRAS				11	10	9	8	7	6	5	4	3	2	1	0																		
	MBA																					1	0											
	MCAS																						9	8	7	6	5	4	3	2	1	0		
12 x 9 x 2	MRAS				11	10	9	8	7	6	5	4	3	2	1	0																		
	MBA																						1	0										
	MCAS																							8	7	6	5	4	3	2	1	0		
12 x 8 x 2	MRAS				11	10	9	8	7	6	5	4	3	2	1	0																		
	MBA																						1	0										
	MCAS																								7	6	5	4	3	2	1	0		

Table 9-41. DDR2 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled

Row x Col	msb	Address from Core Master																												lsb				
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27		28	29-31		
15 x 10 x 3	MRAS		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																			2	1	0												
	MCAS																						9	8	7	6	5	4	3	2	1	0		
14 x 10 x 3	MRAS			13	12	11	10	9	8	7	6	5	4	3	2	1	0																	
	MBA																				2	1	0											
	MCAS																							9	8	7	6	5	4	3	2	1	0	

Table 9-41. DDR2 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled (continued)

Row x Col	msb	Address from Core Master																												lsb	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27		28
14 x 10 x 2	$\overline{\text{MRAS}}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{\text{MRAS}}$				12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																			9	8	7	6	5	4	3	2	1	0		
13 x 10 x 2	$\overline{\text{MRAS}}$				12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		1	0											
	$\overline{\text{MCAS}}$																			9	8	7	6	5	4	3	2	1	0		
13 x 9 x 2	$\overline{\text{MRAS}}$				12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																			1	0										
	$\overline{\text{MCAS}}$																				8	7	6	5	4	3	2	1	0		

Table 9-42. DDR2 Address Multiplexing for 32-Bit Data Bus with Interleaving and Partial Array Self-Refresh Disabled

Row x Col	msb	Address from Core Master																													lsb
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	
15 x 10 x 3	$\overline{\text{MRAS}}$			14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		2	1	0										
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 3	$\overline{\text{MRAS}}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		2	1	0										
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{\text{MRAS}}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																			1	0										
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{\text{MRAS}}$			12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																		2	1	0										
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	

Table 9-42. DDR2 Address Multiplexing for 32-Bit Data Bus with Interleaving and Partial Array Self-Refresh Disabled (continued)

Row x Col	msb	Address from Core Master																												lsb			
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27		28	29	30-31
13 x 10 x 2	MRAS					12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																			1	0												
	MCAS																						9	8	7	6	5	4	3	2	1	0	
13 x 9 x 2	MRAS					12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																				1	0											
	MCAS																						8	7	6	5	4	3	2	1	0		

Chip select interleaving is supported for the memory controller, and is programmed in DDR_SDRAM_CFG[BA_INTLV_CTL]. Interleaving is supported between chip selects 0 and 1 or chip selects 2 and 3. In addition, interleaving between all four chip selects can be enabled. When interleaving is enabled, the chip selects being interleaved must use the same size of memory. If two chip selects are interleaved, then 1 extra bit in the address decode is used for the interleaving to determine which chip select to access. If four chip selects are interleaved, then two extra bits are required in the address decode.

Table 9-43 illustrates examples of address decode when interleaving between two chip selects, and Table 9-44 shows examples of address decode when interleaving between four chip selects.

Table 9-43. Example of Address Multiplexing for 64-Bit Data Bus Interleaving between Two Banks with Partial Array Self Refresh Disabled

Row x Col	msb	Address from Core Master																												lsb		
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27		28	29-31
14 x 10 x 3	MRAS		13	12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																		CS SEL	2	1	0										
	MCAS																						9	8	7	6	5	4	3	2	1	0
14 x 10 x 2	MRAS			13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																			CS SEL	1	0										
	MCAS																						9	8	7	6	5	4	3	2	1	0
13 x 10 x 3	MRAS			12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																			CS SEL	2	1	0									
	MCAS																						9	8	7	6	5	4	3	2	1	0
13 x 10 x 2	MRAS			12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																			CS SEL	1	0										
	MCAS																						9	8	7	6	5	4	3	2	1	0

9.5.3 JEDEC Standard DDR SDRAM Interface Commands

Table 9-44. Example of Address Multiplexing for 64-Bit Data Bus Interleaving between Four Banks with Partial Array Self Refresh Disabled

Row x Col	msb	Address from Core Master																												lsb																
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27		28	29–31														
14 x 10 x 3	MRAS	13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL																														
	MBA																	2	1	0																										
	MCAS																					9	8	7	6	5	4	3	2	1	0															
14 x 10 x 2	MRAS		13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL																													
	MBA																		1	0																										
	MCAS																					9	8	7	6	5	4	3	2	1	0															
13 x 10 x 3	MRAS		12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL																														
	MBA																		2	1	0																									
	MCAS																					9	8	7	6	5	4	3	2	1	0															
13 x 10 x 2	MRAS		12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL																														
	MBA																		1	0																										
	MCAS																					9	8	7	6	5	4	3	2	1	0															

The following section describes the commands and timings the controller uses when operating in DDR2 or DDR modes.

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

The following DDR SDRAM interface commands (summarized in [Table 9-45](#)) are provided by the DDR controller. All actions for these commands are described from the perspective of the SDRAM device.

- Row activate—Latches row address and initiates memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored by a precharge command before another row activate occurs.
- Precharge—Restores data from the sense amplifiers to the appropriate row. Also initializes the sense amplifiers in preparation for reading another row in the memory array (performing another activate command). Precharge must occur after read or write, if the row address changes on the next open page mode access.
- Read—Latches column address and transfers data from the selected sense amplifier to the output buffer as determined by the column address. During each succeeding clock edge, additional data is driven without additional read commands. The amount of data transferred is determined by the burst size which defaults to 4.
- Write—Latches column address and transfers data from the data pins to the selected sense amplifier as determined by the column address. During each succeeding clock edge, additional data

is transferred to the sense amplifiers from the data pins without additional write commands. The amount of data transferred is determined by the data masks and the burst size, which is set to four by the DDR memory controller.

- Refresh (similar to \overline{MCAS} before \overline{MRAS})—Causes a row to be read in all logical banks (JEDEC SDRAM) as determined by the refresh row address counter. This refresh row address counter is internal to the SDRAM. After being read, the row is automatically rewritten in the memory array. All logical banks must be in a precharged state before executing a refresh. The memory controller also supports posted refreshes, where several refreshes may be executed at once, and the refresh interval may be extended.
- Mode register set (for configuration)—Allows setting of DDR SDRAM options. These options are: \overline{MCAS} latency, additive latency (for DDR2), write recovery (for DDR2), burst type, and burst length. \overline{MCAS} latency may be chosen as provided by the preferred SDRAM (some SDRAMs provide \overline{MCAS} latency {1,2,3}, some provide \overline{MCAS} latency {1,2,3,4,5}, 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 a burst length of 4. A burst length of 8 is supported for DDR1 memory only. For DDR2 in 32-bit bus mode, all 32-byte burst accesses from the platform are split into two 16-byte (that is, 4-beat) accesses to the SDRAMs in the memory controller. The mode register set command is performed by the DDR memory controller during system initialization. Parameters such as mode register data, \overline{MCAS} latency, burst length, and burst type, are set by software in `DDR_SDRAM_MODE[SDMODE]` and transferred to the SDRAM array by the DDR memory controller after `DDR_SDRAM_CFG[MEM_EN]` is set. If `DDR_SDRAM_CFG[BI]` is set to bypass the automatic initialization, then the MODE registers can be configured through software through use of the `DDR_SDRAM_MD_CNTL` register.
- Self refresh (for long periods of standby)—Used when the device is in standby for very long periods of time. Automatically generates internal refresh cycles to keep the data in all memory banks refreshed. Before execution of this command, the DDR controller places all logical banks in a precharged state.

Table 9-45. DDR SDRAM Command Table

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

Table 9-45. DDR SDRAM Command Table (continued)

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

9.5.4 DDR SDRAM Interface Timing

The DDR memory controller supports four-beat bursts to SDRAM. For single-beat reads, the DDR memory controller performs a four- (or eight-) beat burst read, but ignores the last three (or seven) beats. Single-beat writes are performed by masking the last three (or seven) beats of the four- (or eight-) beat burst using the data mask MDM[0:8]. If ECC is disabled, writes smaller than double words are performed by appropriately activating the data mask. If ECC is enabled, the controller performs a read-modify write.

NOTE

If a second read or write is pending, reads shorter than four beats are not terminated early even if some data is irrelevant.

To accommodate available memory technologies across a wide spectrum of operating frequencies, the DDR memory controller allows the setting of the intervals defined in [Table 9-46](#) with granularity of one memory clock cycle, except for CASLAT, which can be programmed with $\frac{1}{2}$ clock granularity.

Table 9-46. DDR SDRAM Interface Timing Intervals

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

Table 9-46. DDR SDRAM Interface Timing Intervals (continued)

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

The value of the above parameters (in whole clock cycles) must be set by boot code at system start-up (in the TIMING_CFG_0, TIMING_CFG_1, TIMING_CFG_2, and TIMING_CFG_3 registers as described in Section 9.4.1.4, “DDR SDRAM Timing Configuration 0 (TIMING_CFG_0),” Section 9.4.1.5, “DDR SDRAM Timing Configuration 1 (TIMING_CFG_1),” Section 9.4.1.6, “DDR SDRAM Timing Configuration 2 (TIMING_CFG_2),” and Section 9.4.1.3, “DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)”) and be kept in the DDR memory controller configuration register space.

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

Figure 9-35 through Figure 9-37 show DDR SDRAM timing for various types of accesses; see Figure 9-35 for a single-beat read operation, Figure 9-36 for a single-beat write operation, and Figure 9-37 for a double word write operation. Note that all signal transitions occur on the rising edge of the memory bus clock and that single-beat read operations are identical to burst-reads. These figures assume the CLK_ADJUST is

set to 1/2 DRAM cycle, an additive latency of 0 DRAM cycles is used, and the write latency is 1 DRAM cycle (for DDR1).

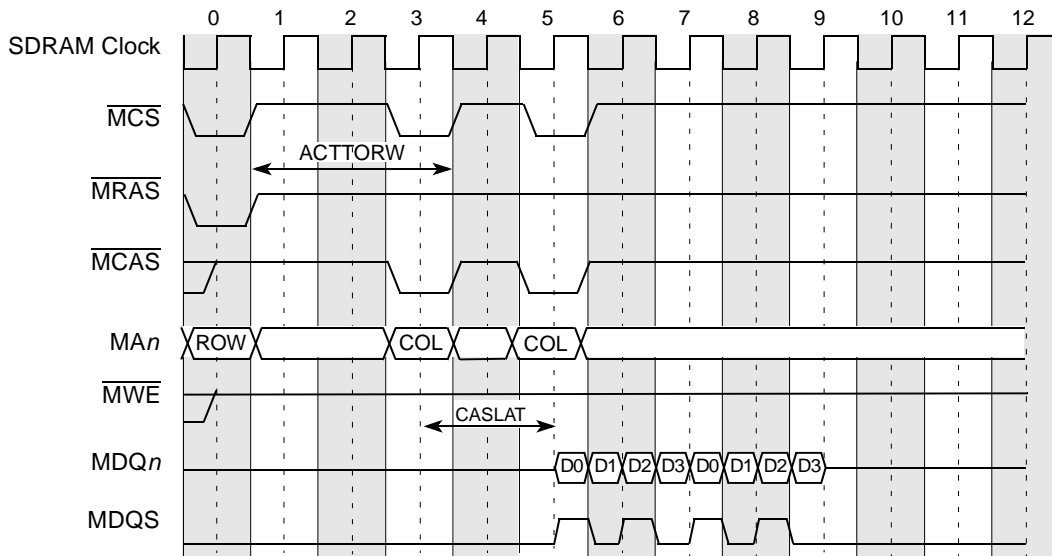


Figure 9-35. DDR SDRAM Burst Read Timing—ACTTORW = 3, MCAS Latency = 2

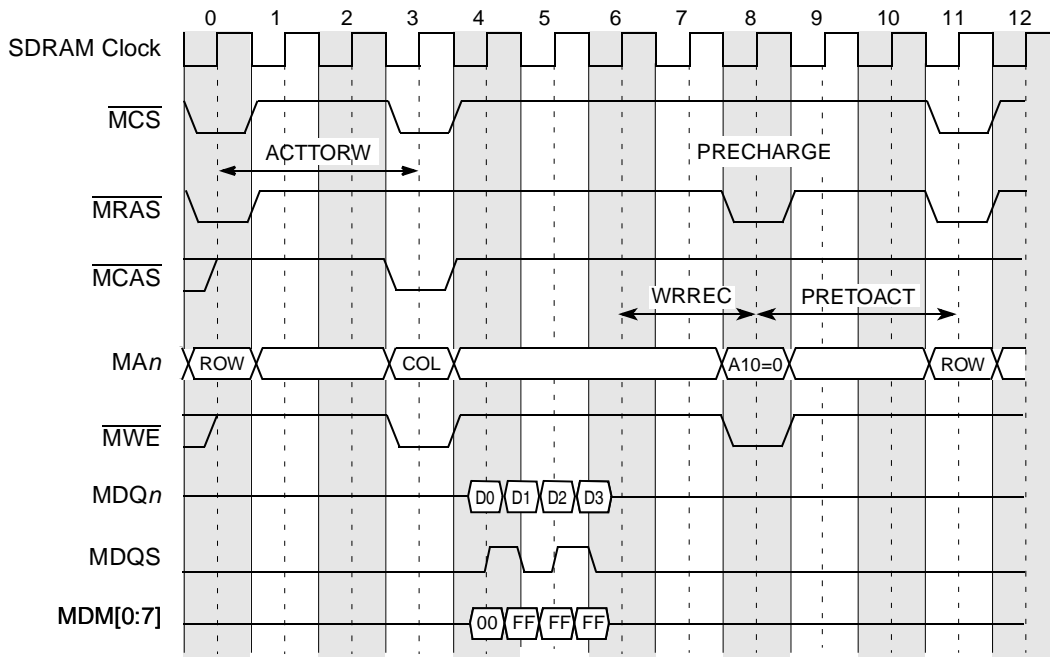


Figure 9-36. DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTOR

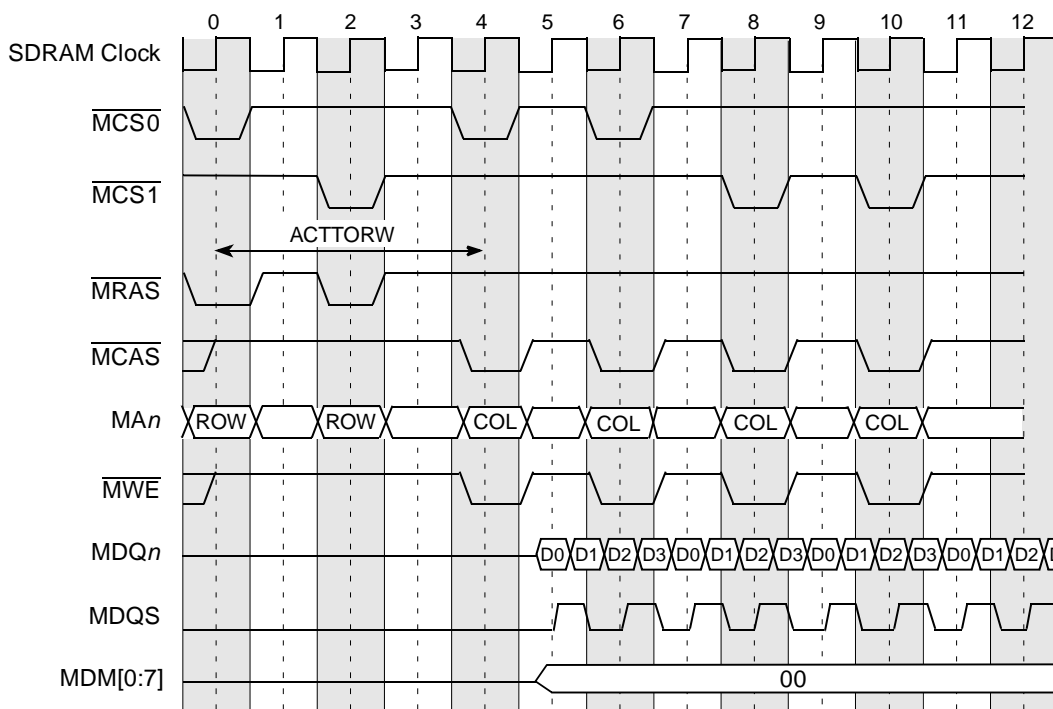


Figure 9-37. DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTORW = 3

9.5.4.1 Clock Distribution

Clock distribution has the following recommendations:

- If running with many devices, zero-delay PLL clock buffers, JEDEC-JESD82 standard, should be used. These buffers were designed for DDR applications.
- A 72-bit x 64 Mbytes DDR bank has 9-byte-wide DDR chips, resulting in 18 DDR chips in a two-bank system. In this case, each MCK/ $\overline{\text{MCK}}$ signal pair should drive exactly three devices.
- PCB traces for DDR clock signals should be short, all on the same layer, and of equal length and loading.

DDR SDRAM manufacturers provide detailed information on PCB layout and termination issues.

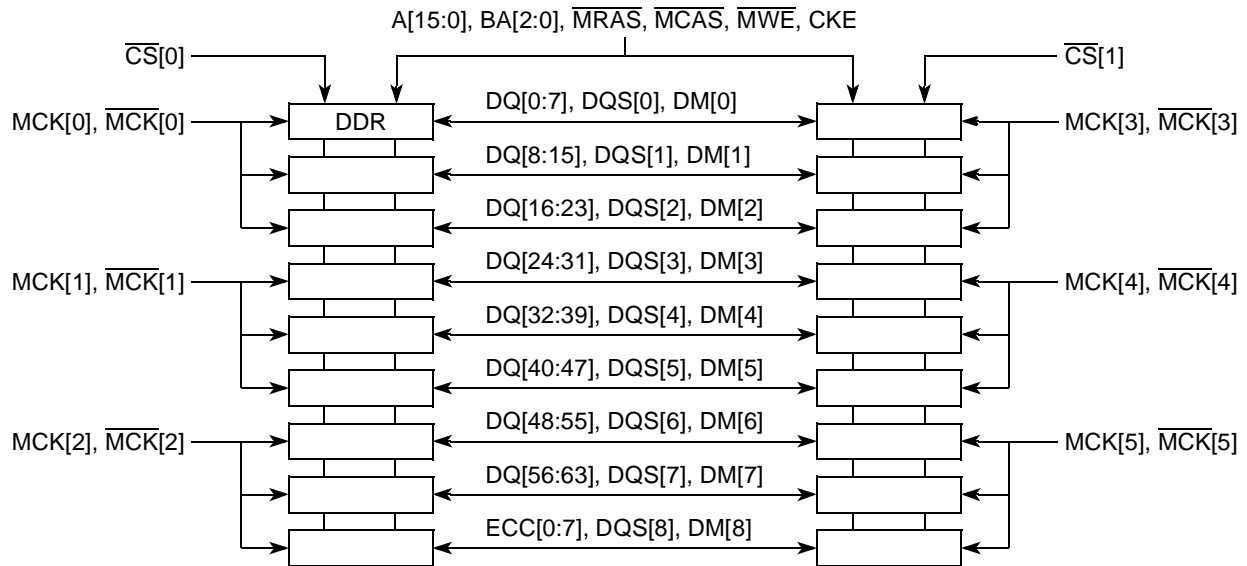


Figure 9-38. DDR SDRAM Clock Distribution Example for x8 DDR SDRAMs

9.5.5 DDR SDRAM Mode-Set Command Timing

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

Figure 9-39 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.

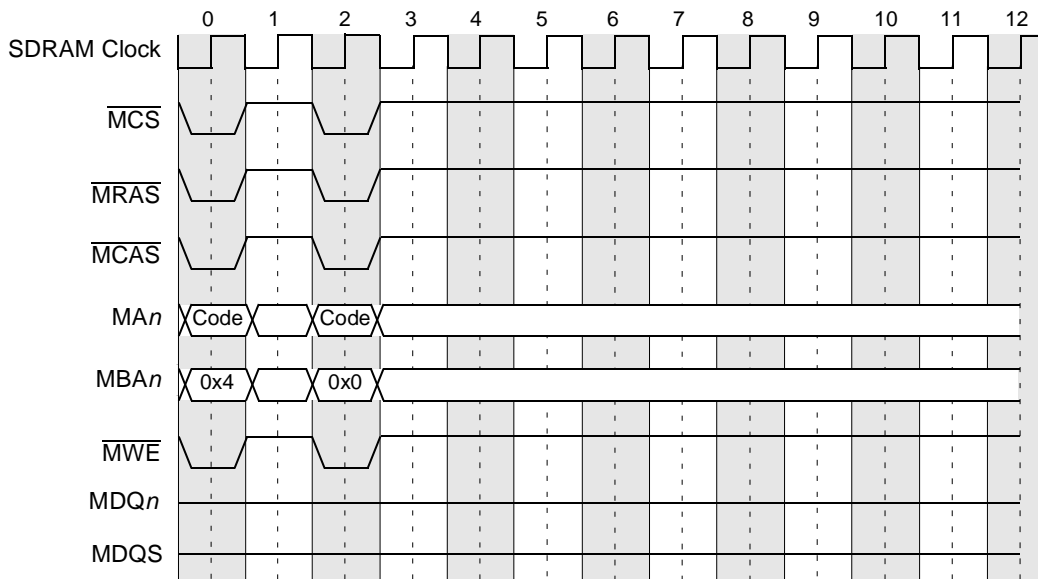


Figure 9-39. DDR SDRAM Mode-Set Command Timing

9.5.6 DDR SDRAM Registered DIMM Mode

To reduce loading, registered DIMMs latch the DDR SDRAM control signals internally before using them to access the array. Setting `DDR_SDRAM_CFG[RD_EN]` compensates for this delay on the DIMMs' control bus by delaying the data and data mask writes (on SDRAM buses) by an extra SDRAM clock cycle.

NOTE

Application system board must assert the reset signal on DDR memory devices until software is able to program the DDR memory controller configuration registers, and must deassert the reset signal on DDR memory devices before `DDR_SDRAM_CFG[MEM_EN]` is set. This ensures that the DDR memory devices are held in reset until a stable clock is provided and, further, that a stable clock is provided before memory devices are released from reset.

Figure 9-40 shows the registered DDR SDRAM DIMM single-beat write timing.

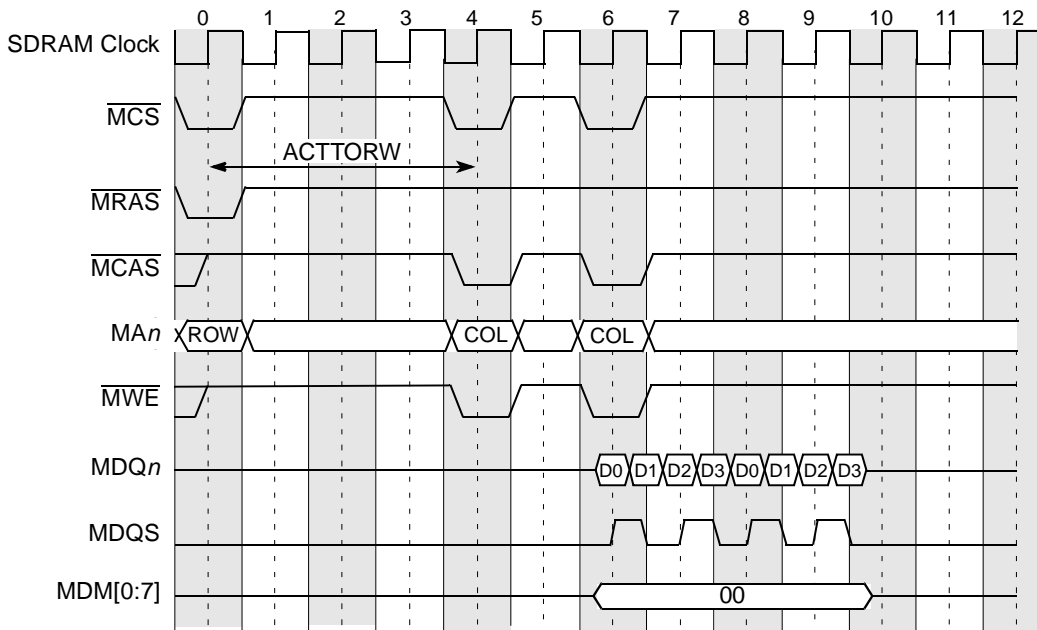


Figure 9-40. Registered DDR SDRAM DIMM Burst Write Timing

9.5.7 DDR SDRAM Write Timing Adjustments

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

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

Figure 9-41 shows the use of the WR_DATA_DELAY parameter.

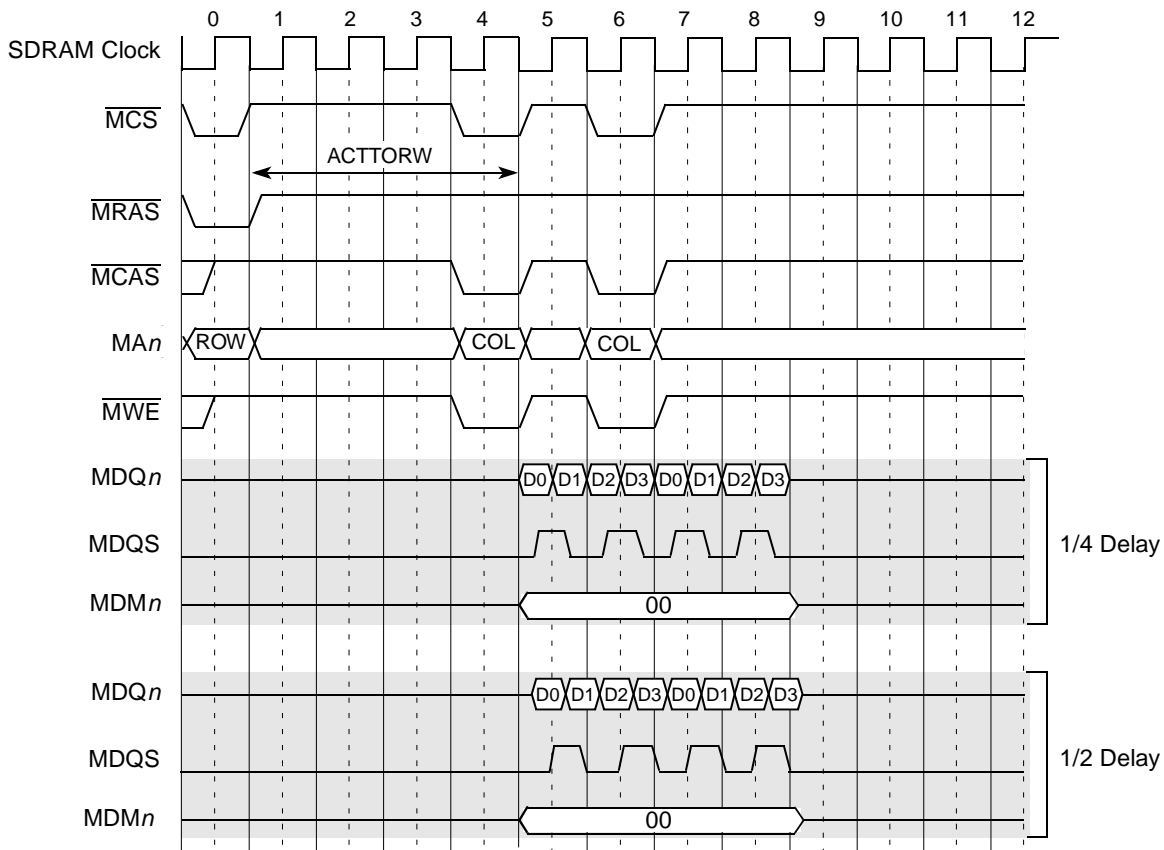


Figure 9-41. Write Timing Adjustments Example for Write Latency = 1

9.5.8 DDR SDRAM Refresh

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

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

1. Completes all current memory requests.

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

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

9.5.8.1 DDR SDRAM Refresh Timing

Refresh timing for the DDR SDRAM is controlled by the programmable timing parameter TIMING_CFG_1 [REFREC], which specifies the number of memory bus clock cycles from the refresh command until a logical bank activate command is allowed. The DDR memory controller implements bank staggering for refreshes, as shown in Figure 9-42 (TIMING_CFG_1 [REFREC] = 10 in this example).

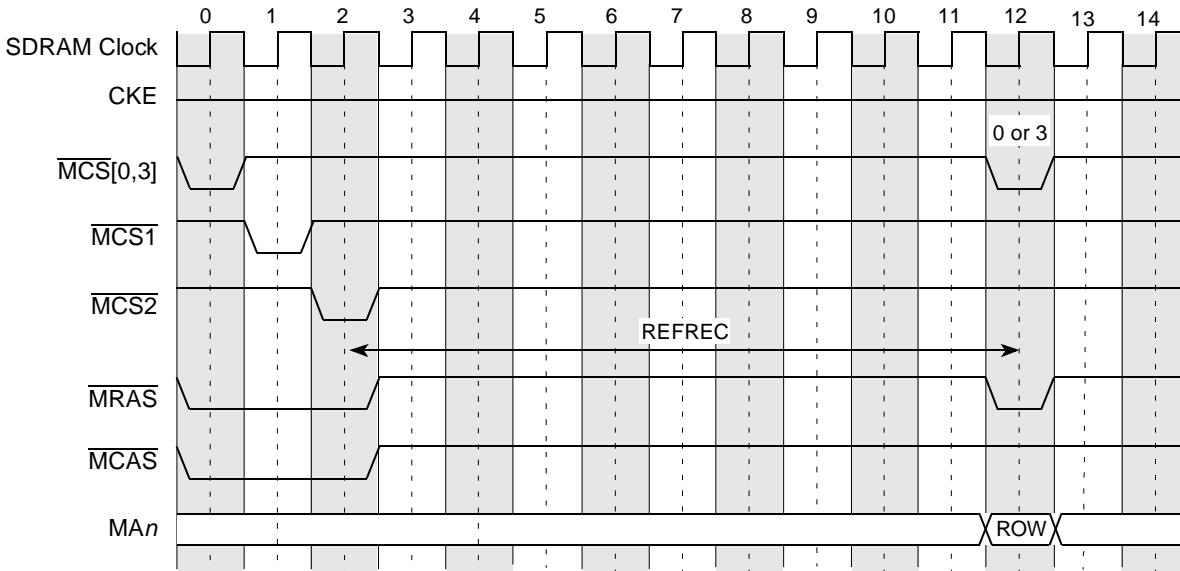


Figure 9-42. DDR SDRAM Bank Staggered Auto Refresh Timing

System software is responsible for optimal configuration of TIMING_CFG_1 [REFREC] and TIMING_CFG_3[EXT_REFREC] at reset. Configuration must be completed before DDR SDRAM accesses are attempted.

9.5.8.2 DDR SDRAM Refresh and Power-Saving Modes

In full-on mode, the DDR memory controller supplies the normal auto refresh to SDRAM. In sleep mode, the DDR memory controller can be configured to take advantage of self-refreshing SDRAMs or to provide no refresh support. Self-refresh support is enabled with the SREN memory control parameter.

Table 9-47 summarizes the refresh types available in each power-saving mode.

Table 9-47. DDR SDRAM Power-Saving Modes Refresh Configuration

Power Saving Mode	Refresh Type	SREN
Sleep	Self	1
	None	—

Note that in the absence of refresh support, system software must preserve DDR SDRAM data (such as by copying the data to disk) before entering the power-saving mode.

The dynamic power-saving mode uses the CKE DDR SDRAM pin to dynamically power down when there is no system memory activity. The CKE pin is negated when both of the following conditions are met:

- No memory refreshes are scheduled
- No memory accesses are scheduled

CKE is reasserted when a new access or refresh is scheduled or the dynamic power mode is disabled. This mode is controlled with DDR_SDRAM_CFG[DYN_PWR_MGMT].

Dynamic power management mode offers tight control of the memory system's power consumption by trading power for performance through the use of CKE. Powering up the DDR SDRAM when a new memory reference is scheduled causes an access latency penalty, depending on whether active or precharge powerdown is used, along with the settings of TIMING_CFG_0[ACT_PD_EXIT] and TIMING_CFG_0[PRE_PD_EXIT]. A penalty of 1 cycle is shown in Figure 9-43.

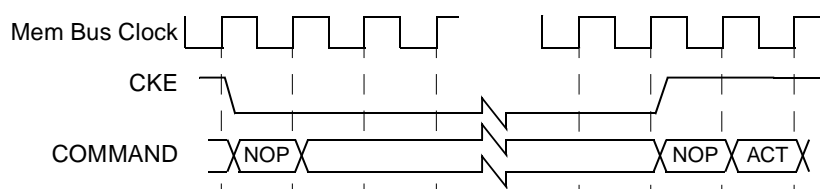


Figure 9-43. DDR SDRAM Power-Down Mode

9.5.8.2.1 Self-Refresh in Sleep Mode

The entry and exit timing for self-refreshing SDRAMs is shown in [Figure 9-44](#) and [Figure 9-45](#).

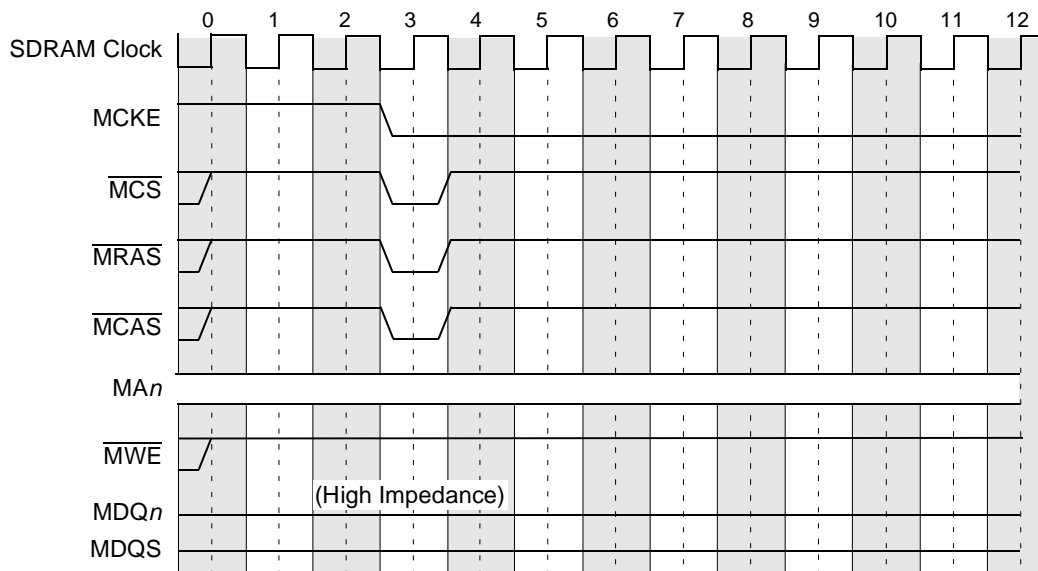


Figure 9-44. DDR SDRAM Self-Refresh Entry Timing

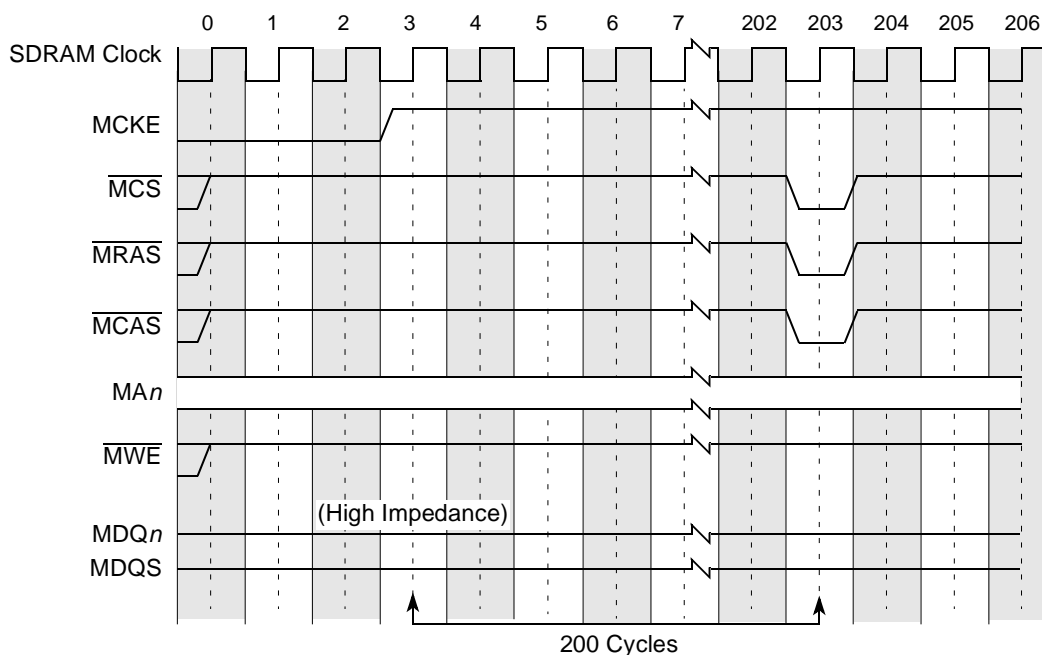


Figure 9-45. DDR SDRAM Self-Refresh Exit Timing

9.5.9 DDR Data Beat Ordering

Transfers to and from memory are always performed in four- or eight-beat bursts (four beats = 32 bytes when a 64-bit bus is used). For transfer sizes other than four or eight beats, the data transfers are still operated as four- or eight-beat bursts. If ECC is enabled and either the access is not double-word aligned or the size is not a multiple of a double word, a full read-modify-write is performed for a write to SDRAM. If ECC is disabled or both the access is double-word aligned with a size that is a multiple of a double word, the data masks (MDM[0:8] (MDM[0:] for 32-bit bus) can be used to prevent the writing of unwanted data to SDRAM. The DDR memory controller also uses data masks to prevent all unintended full double words from writing to SDRAM. For example, if a write transaction is desired with a size of one double word (8 bytes), then the second, third, and fourth beats of data are not written to DRAM, as the width of the data bus is 64 bits.

Table 9-48 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 9-48. Memory Controller—Data Beat Ordering

Transfer Size	Starting Double-Word Offset	Double-Word Sequence ¹ to/from DRAM and Queues
1 double word	0	<u>0</u> - 1 - 2 - 3
	1	1 - 2 - 3 - 0
	2	<u>2</u> - 3 - 0 - 1
	3	3 - 0 - 1 - 2
2 double words	0	<u>0</u> - <u>1</u> - 2 - 3
	1	1 - <u>2</u> - 3 - 0
	2	<u>2</u> - <u>3</u> - 0 - 1
3 double words	0	<u>0</u> - <u>1</u> - <u>2</u> - 3
	1	1 - <u>2</u> - <u>3</u> - 0

¹ All underlined **Double**-word offsets are valid for the transaction.

9.5.10 Page Mode and Logical Bank Retention

The DDR memory controller supports an open/closed page mode with an allowable open page for each logical bank of DRAM used. In closed page mode for DDR SDRAMs, the DDR memory controller uses the SDRAM auto-precharge feature, which allows the controller to indicate that the page must be automatically closed by the DDR SDRAM after the READ or WRITE access. This is performed using MA[10] of the address during the COMMAND phase of the access to enable auto-precharge. Auto-precharge is non-persistent in that it is either enabled or disabled for each individual READ or WRITE command. It can, however, be enabled or disabled separately for each chip select.

When the DDR memory controller operates in open page mode, it retains the currently active SDRAM page by not issuing a precharge command. The page remains open 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 using more banks, especially in systems which use many different channels. Page mode is disabled by clearing `DDR_SDRAM_INTERVAL[BSTOPRE]` or setting `CSn_CONFIG[AP_nEN]`.

9.5.11 Error Checking and Correcting (ECC)

The DDR memory controller supports error checking and correcting (ECC) for the data path between the core master and system memory. The memory detects all double-bit errors, detects all multi-bit errors within a nibble, and corrects all single-bit errors. Other errors may be detected, but are not guaranteed to be corrected or detected. Multi-bit errors are always reported when error reporting is enabled. When a single-bit error occurs, the single-bit error counter register is incremented, and its value compared to the single-bit error trigger register. An error is reported when these values are equal. The single-bit error registers can be programmed such that minor memory faults are corrected and ignored, but a catastrophic memory failure generates an interrupt.

For writes that are smaller than 64 bits, the DDR memory controller performs a double-word read from system memory of the address for the write (checking for errors), and merges the write data with the data read from memory. Then, a new ECC code is generated for the merged double word. The data and ECC code is then written to memory. If a multi-bit error is detected on the read, the transaction completes the read-modify-write to keep the DDR memory controller from hanging. However, the corrupt data is masked on the write, so the original contents in SDRAM remain unchanged.

The syndrome encodings for the ECC code are shown in [Table 9-49](#) and [Table 9-50](#).

In 32-bit mode, [Table 9-49](#) is split into 2 halves. The first half, consisting of rows 0–31, is used to calculate the ECC bits for the first 32 data bits of any 64-bit granule of data. This always applies to the odd data beats on the DDR data bus. The second half of the table, consisting of rows 32–63, is used to calculate the ECC bits for the second 32 bits of any 64-bit granule of data. This always applies to the even data beats on the DDR data bus.

Table 9-49. DDR SDRAM ECC Syndrome Encoding

Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•	•						•
1	•		•					•
2	•			•				•
3	•				•			•
4	•	•				•		
5	•		•			•		
6	•			•		•		
7	•				•	•		

Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
32			•	•				•
33			•		•			•
34	•		•		•			
35		•	•		•			
36			•	•		•		
37			•		•	•		
38	•		•		•	•		•
39		•	•		•	•		•

Table 9-49. DDR SDRAM ECC Syndrome Encoding (continued)

Data Bit	Syndrome Bit								Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
8	•	•					•		40			•	•			•	
9	•		•				•		41			•		•		•	
10	•			•			•		42	•		•		•		•	•
11	•				•		•		43		•	•		•		•	•
12	•	•				•	•	•	44			•	•		•	•	•
13	•		•			•	•	•	45			•		•	•	•	•
14	•			•		•	•	•	46	•		•		•	•	•	
15	•				•	•	•	•	47		•	•		•	•	•	
16		•	•					•	48		•			•	•		
17		•		•				•	49			•			•	•	
18		•			•			•	50				•		•	•	
19	•	•			•				51	•					•	•	
20		•	•			•			52		•				•		•
21		•		•		•			53			•			•		•
22		•			•	•			54				•		•		•
23	•	•			•	•		•	55	•					•		•
24		•	•				•		56		•				•	•	
25		•		•			•		57			•			•	•	
26		•			•		•		58				•		•	•	
27	•	•			•		•	•	59	•					•	•	
28		•	•			•	•	•	60				•	•		•	
29		•		•		•	•	•	61	•			•	•		•	•
30		•			•	•	•	•	62		•		•	•		•	•
31	•	•			•	•	•		63			•	•	•		•	•

Table 9-50. DDR SDRAM ECC Syndrome Encoding (Check Bits)

Check Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•							
1		•						
2			•					
3				•				
4					•			
5						•		

Table 9-50. DDR SDRAM ECC Syndrome Encoding (Check Bits) (continued)

Check Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
6							•	
7								•

9.5.12 Error Management

The DDR memory controller detects four different kinds of errors: training, single-bit, multi-bit, and memory select errors. The following discussion assumes all the relevant error detection, correction, and reporting functions are enabled as described in [Section 9.4.1.26, “Memory Error Interrupt Enable \(ERR_INT_EN\),”](#) [Section 9.4.1.25, “Memory Error Disable \(ERR_DISABLE\),”](#) and [Section 9.4.1.24, “Memory Error Detect \(ERR_DETECT\).”](#)

Single-bit errors are counted and reported based on the ERR_SBE value. When a single-bit error is detected, the DDR memory controller does the following:

- Corrects the data
- Increments the single-bit error counter ERR_SBE[SBEC]
- Generates a critical interrupt if the counter value ERR_SBE[SBEC] equals the programmable threshold ERR_SBE[SBET]
- Completes the transaction normally

If a multi-bit error is detected for a read, the DDR memory controller logs the error and generates the interrupt, and transfer error acknowledge (TEA) is asserted internally on the CSB bus (if enabled, as described in [Section 9.4.1.25, “Memory Error Disable \(ERR_DISABLE\)”](#)). Another error the DDR memory controller detects is a memory select error, which causes the DDR memory controller to log the error and generate a critical interrupt (if enabled, as described in [Section 9.4.1.24, “Memory Error Detect \(ERR_DETECT\)”](#)). This error is detected if the address from the memory request does not fall into any of the enabled, programmed chip select address ranges. For all memory select errors, the DDR memory controller does not issue any transactions onto the pins after the first read has returned data strobes. If the DDR memory controller is not using sample points, then a dummy transaction is issued to DDR SDRAM with the first enabled chip select. In this case, the source port on the pins is forced to 0x1F to show the transaction is not real. [Table 9-51](#) shows the errors with their descriptions. The final error the memory controller detects is the automatic calibration error. This error is set if the memory controller detects an error during its training sequence.

Table 9-51. 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 through interrupt if enabled.	The error control register only logs read versus write, not full type
Access Error	Multi-bit ECC error	A multi-bit ECC error is detected during a read, or read-modify-write memory operation.		
	Memory select error	Read, or write, address does not fall within the address range of any of the memory banks.		

9.6 Initialization/Application Information

System software must configure the DDR memory controller, using a memory polling algorithm at system start-up, to correctly map the size of each bank in memory. Then, the DDR memory controller uses its bank map to assert the appropriate \overline{MCS}_n signal for memory accesses according to the provided bank depths. System software must also configure the DDR memory controller at system start-up to appropriately multiplex the row and column address bits for each bank. Refer to row-address configuration in [Section 9.4.1.2, “Chip Select Configuration \(CS_n_CONFIG\).”](#) Address multiplexing occurs according to these configuration bits.

At system reset, initialization software (boot code) must set up the programmable parameters in the memory interface configuration registers. See [Section 9.4.1, “DDR Register Descriptions,”](#) for more detailed descriptions of the configuration registers. These parameters are shown in [Table 9-52.](#)

Table 9-52. Memory Interface Configuration Register Initialization Parameters

Name	Description	Parameter	Section/page
CS _n _BNDS	Chip select memory bounds	SA _n EA _n	9.4.1.1/9-12
CS _n _CONFIG	Chip select configuration	CS _n _EN AP _n _EN ODT_RD_CFG ODT_WR_CFG BA_BITS_CS _n ROW_BITS_CS _n COL_BITS_CS _n	9.4.1.2/9-12
TIMING_CFG_3	Extended timing parameters for fields in TIMING_CFG_1	EXT_REFREC	9.4.1.3/9-14
TIMING_CFG_0	Timing configuration	RWT WRT RRT WWT ACT_PD_EXIT PRE_PD_EXIT ODT_PD_EXIT MRS_CYC	9.4.1.4/9-15
TIMING_CFG_1	Timing configuration	PRETOACT ACTTOPRE ACTTORW CASLAT REFREC WRREC ACTTOACT WRTORD	9.4.1.5/9-17
TIMING_CFG_2	Timing configuration	ADD_LAT CPO WR_LAT RD_TO_PRE WR_DATA_DELAY CKE_PLS FOUR_ACT	9.4.1.6/9-19

Table 9-52. Memory Interface Configuration Register Initialization Parameters (continued)

Name	Description	Parameter	Section/page	
DDR_SDRAM_CFG	Control configuration	SREN ECC_EN RD_EN SDRAM_TYPE DYN_PWR 32_BE 8_BE DBW	NCAP 2T_EN BA_INTLV_CTL x32_EN HSE BI	9.4.1.7/9-21
DDR_SDRAM_CFG_2	Control configuration	DLL_RST_DIS DQS_CFG ODT_CFG	NUM_PR D_INIT	9.4.1.8/9-23
DDR_SDRAM_MODE	Mode configuration	ESDMODE SDMODE		9.4.1.9/9-25
DDR_SDRAM_MODE_2	Mode configuration	ESDMODE2 ESDMODE3		9.4.1.10/9-26
DDR_SDRAM_INTERVAL	Interval configuration	REFINT BSTOPRE		9.4.1.12/9-29
DDR_DATA_INIT	Data initialization configuration register	INIT_VALUE		9.4.1.13/9-29
DDR_SDRAM_CLK_CNTL	Clock adjust	CLK_ADJUST		9.4.1.14/9-30
DDR_INIT_ADDR	Initialization address	INIT_ADDR		9.4.1.15/9-30

9.6.1 Programming Differences between Memory Types

Depending on the memory type used, certain fields must be programmed differently. [Table 9-53](#) illustrates the differences in certain fields for different memory types. Note that this table does not list all fields that must be programmed.

Table 9-53. Programming Differences between Memory Types

Parameter	Description	Differences		Section/page
AP _n _EN	Chip Select <i>n</i> Auto Precharge Enable	DDR1	Can be used to place chip select <i>n</i> in auto precharge mode	9.4.1.2/9-12
		DDR2	Can be used to place chip select <i>n</i> in auto precharge mode	
ODT_RD_CFG	Chip Select ODT Read Configuration	DDR1	Should always be set to 000	9.4.1.2/9-12
		DDR2	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, systems with only 1 chip select typically not uses ODT when issuing reads to the memory.	

Table 9-53. Programming Differences between Memory Types (continued)

Parameter	Description	Differences		Section/page
ODT_WR_CFG	Chip Select ODT Write Configuration	DDR1	Should always be set to 000	9.4.1.2/9-12
		DDR2	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, ODT typically is set to assert for the chip select that is getting written to (value would be set to 001).	
ODT_PD_EXIT	ODT Powerdown Exit	DDR1	Should be set to 0001	9.4.1.4/9-15
		DDR2	Should be set according to the DDR2 specifications for the memory used. The JEDEC parameter this applies to is t_{AXPD} .	
PRETOACT	Precharge to Activate Timing	DDR1	Should be set according to the specifications for the memory used (t_{RP})	9.4.1.5/9-17
		DDR2	Should be set according to the specifications for the memory used (t_{RP})	
ACTTOPRE	Activate to Precharge Timing	DDR1	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used (t_{RAS})	9.4.1.5/9-17
		DDR2	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used (t_{RAS})	
ACTTORW	Activate to Read/Write Timing	DDR1	Should be set according to the specifications for the memory used (t_{RCD})	9.4.1.5/9-17
		DDR2	Should be set according to the specifications for the memory used (t_{RCD})	
CASLAT	CAS Latency	DDR1	Should be set, along with the Extended CAS Latency, to the desired CAS latency	9.4.1.5/9-17
		DDR2	Should be set, along with the Extended CAS Latency, to the desired CAS latency	
REFREC	Refresh Recovery	DDR1	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used (t_{RFC})	9.4.1.5/9-17
		DDR2	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used (T_{RFC})	
WRREC	Write Recovery	DDR1	Should be set according to the specifications for the memory used (t_{WR})	9.4.1.5/9-17
		DDR2	Should be set according to the specifications for the memory used (t_{WR})	
ACTTOACT	Activate A to Activate B	DDR1	Should be set according to the specifications for the memory used (t_{RRD})	9.4.1.5/9-17
		DDR2	Should be set according to the specifications for the memory used (t_{RRD})	

Table 9-53. Programming Differences between Memory Types (continued)

Parameter	Description	Differences		Section/page
WRTORD	Write to Read Timing	DDR1	Should be set according to the specifications for the memory used (t_{WTR})	9.4.1.5/9-17
		DDR2	Should be set according to the specifications for the memory used (t_{WTR})	
ADD_LAT	Additive Latency	DDR1	Should be set to 000	9.4.1.6/9-19
		DDR2	Should be set to the desired additive latency. This must be set to a value less than TIMING_CFG_1[ACTTORW]	
WR_LAT	Write Latency	DDR1	Should be set to 001	9.4.1.6/9-19
		DDR2	Should be set to CAS latency – 1 cycle. For example, if the CAS latency is 5 cycles, then this field should be set to 100 (4 cycles).	
RD_TO_PRE	Read to Precharge Timing	DDR1	Should be set to 010 if burst length is 4 and 100 if burst length is 8	9.4.1.6/9-19
		DDR2	Should be set according to the specifications for the memory used (t_{RTP}). Time between read and precharge for non-zero value of additive latency (AL) is a minimum of $AL + t_{RTP}$ cycles.	
CKE_PLS	Minimum CKE Pulse Width	DDR1	Can be set to 001	9.4.1.6/9-19
		DDR2	Should be set according to the specifications for the memory used (t_{CKE})	
FOUR_ACT	Four Activate Window	DDR1	Should be set to 00001	9.4.1.6/9-19
		DDR2	Should be set according to the specifications for the memory used (t_{FAW}). Only applies to eight logical banks.	
RD_EN	Registered DIMM Enable	DDR1	If registered DIMMs are used, then this field should be set to 1	9.4.1.7/9-21
		DDR2	If registered DIMMs are used, then this field should be set to 1	
8_BE	8-beat burst enable	DDR1	If a 32-bit bus is used, and 8-beat bursts are desired, then this field should be set to 1	9.4.1.7/9-21
		DDR2	Should be set to 0	
2T_EN	2T Timing Enable	DDR1	In heavily loaded systems, this can be set to 1 to gain extra timing margin on the interface at the cost of address/command bandwidth.	9.4.1.7/9-21
		DDR2	In heavily loaded systems, this can be set to 1 to gain extra timing margin on the interface at the cost of address/command bandwidth.	
DLL_RST_DIS	DLL Reset Disable	DDR1	Should typically be set to 0, unless it is desired to bypass the DLL reset when exiting self refresh.	9.4.1.8/9-23
		DDR2	Should typically be set to 0, unless it is desired to bypass the DLL reset when exiting self refresh.	

Table 9-53. Programming Differences between Memory Types (continued)

Parameter	Description	Differences		Section/page
DQS_CFG	DQS Configuration	DDR1	Should be set to 00	9.4.1.8/9-23
		DDR2	Should be set to 00	
ODT_CFG	ODT Configuration	DDR1	Should be set to 00	9.4.1.8/9-23
		DDR2	Can be set for termination at the IOs according to system topology. Typically, if ODT is enabled, then the internal IOs should be set up for termination only during reads to DRAM.	
BSTOPR	Burst To Precharge Interval	DDR1	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	9.4.1.12/9-29
		DDR2	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	

9.6.2 DDR SDRAM Initialization Sequence

After configuration of all parameters is complete, system software must set `DDR_SDRAM_CFG[MEM_EN]` to enable the memory interface. Note that 200 μ s must elapse after DRAM clocks are stable (`DDR_SDRAM_CLK_CNTL[CLK_ADJUST]` is set and any chip select is enabled) before `MEM_EN` can be set, so a delay loop in the initialization code may be necessary if software is enabling the memory controller. If `DDR_SDRAM_CFG[BI]` is not set, the DDR memory controller conducts an automatic initialization sequence to the memory, which follows the memory specifications. If the bypass initialization mode is used, then software can initialize the memory through the `DDR_SDRAM_MD_CNTL` register.



Chapter 10

Local Bus Controller

This chapter describes the local bus controller (LBC) block. It describes the external signals and the memory-mapped registers and contains a functional description of the general-purpose chip-select machine (GPCM), synchronous DRAM (SDRAM) machine, and user-programmable machines (UPMs) of the LBC. Finally, it includes initialization and applications information sections with many specific examples of its use.

10.1 LBC Introduction

Figure 10-1 is a functional block diagram of the LBC, which supports three interfaces: GPCM, UPM, and an SDRAM controller.

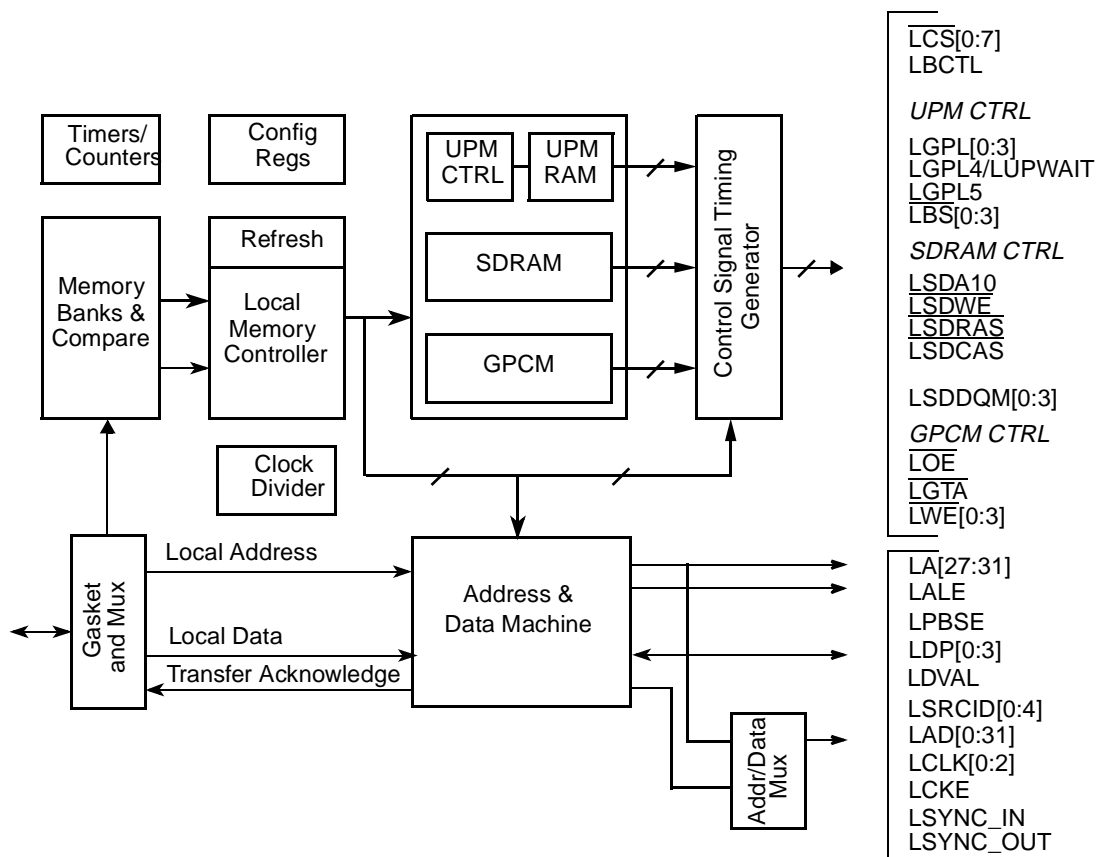


Figure 10-1. Local Bus Controller Block Diagram

The main component of the LBC is its memory controller, which provides a seamless interface to many types of memory devices and peripherals. The memory controller is responsible for controlling eight

memory banks shared by a high performance SDRAM machine, a GPCM, and up to three UPMs. As such, it supports a minimal glue logic interface to SDRAM, SRAM, EPROM, flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals. The external address latch signal (LALE) allows multiplexing of addresses with data signals to reduce the device signal count.

The LBC also includes a number of data checking and protection features such as data parity generation and checking, write protection and a bus monitor to ensure that each bus cycle is terminated within a user-specified period.

10.1.1 LBC Features

The LBC main features are as follows:

- Memory controller with eight memory banks
 - 32-bit address decoding with mask
 - Variable memory block sizes (32 Kbytes to 2 Gbytes)
 - Selection of control signal generation on a per-bank basis
 - Data buffer controls activated on a per-bank basis
 - Automatic segmentation of large transactions
 - Odd/even parity checking including read-modify-write (RMW) parity for single accesses
 - Write-protection capability
 - Atomic operation
 - Parity byte-select
- SDRAM machine
 - Provides the control functions and signals for glueless connection to JEDEC-compliant SDRAM devices
 - Supports up to 4 concurrent open pages per device
 - Supports SDRAM port size of 32-bit, 16-bit and 8-bit
 - Supports external address and/or command lines buffering
- General-purpose chip-select machine (GPCM)
 - Compatible with SRAM, EPROM, FEPRM, and peripherals
 - Global (boot) chip-select available at system reset
 - Boot chip-select support for 8-, 16-, or 32-bit devices
 - Minimum three-clock access to external devices
 - Four byte-write-enable signals ($\overline{\text{LWE}}[0:3]$)
 - Output enable signal ($\overline{\text{LOE}}$)
 - External access termination signal ($\overline{\text{LGTA}}$)

- Three user-programmable machines (UPMs)
 - Programmable-array-based machine controls external signal timing with a granularity of up to one quarter of an external bus clock period
 - User-specified control-signal patterns run when an internal master requests a single-beat or burst read or write access.
 - UPM refresh timer runs a user-specified control signal pattern to support refresh
 - User-specified control-signal patterns can be initiated by software
 - Each UPM can be defined to support DRAM devices with depths of 64, 128, 256, and 512 Kbytes, and 1, 2, 4, 8, 16, 32, 64, 128, and 256 Mbytes
 - Support for 8-, 16-, 32-bit devices
 - Page mode support for successive transfers within a burst
 - Internal address multiplexing supporting 64-, 128-, 256-, and 512-Kbyte, and 1-, 2-, 4-, 8-, 16-, 32-, 64-, 128-, and 256-Mbyte page banks
- Optional monitoring of transfers between local bus internal masters and local bus slaves (local bus error reporting)
- Support for delay-locked loop (DLL) with software-configurable bypass for low frequency bus clocks

10.1.2 Modes of Operation

The LBC provides one GPCM, one SDRAM machine, and three UPMs for the local bus, with no restriction on how many of the eight banks (chip selects) can be programmed to operate with any given machine. When a memory transaction is dispatched to the LBC, the memory address is compared with the address information of each bank (chip select). The corresponding machine assigned to that bank (GPCM, SDRAM, or UPM) then takes ownership of the external signals that control the access and maintains control until the transaction ends. Thus, with the LBC in GPCM, SDRAM, or UPM mode, only one of the eight chip selects is active at any time for the duration of the transaction. See [Section 10.4, “Functional Description.”](#)

10.1.2.1 LBC Bus Clock and Clock Ratios

The LBC supports ratios of 2, 4, and 8 between the faster internal (system) clock and slower external bus clock (LCLK[0:2]). This ratio is software programmable through the clock ratio register (LCRR[CLKDIV]). This ratio has no functional effect on operation in SDRAM mode but it affects the resolution of signal timing shifts in GPCM mode and the interpretation of UPM array words in UPM mode. The bus clock is driven identically onto signals, LCLK[0:2] to allow the clock load to be shared equally across a pair of signal nets, thereby enhancing the edge rates of the bus clock.

10.1.2.2 Source ID Debug Mode

The LBC provides the ID of a transaction source on external device signals. When those signals are selected, the 5-bit internal ID of the current transaction source appears on LSRCID[0:4] whenever valid address or data is available on the LBC external signals. The reserved value of 0x1F, which indicates invalid address or data, appears on the source ID signals at all other times. The combination of a valid

source ID (any value except 0x1F) and the value of external address latch enable (LALE) and data valid (LDVAL) facilitate capturing useful debug data as follows:

- If a valid source ID is detected on LSRCID[0:4] and LALE is asserted, a valid full 32-bit address may be latched from LAD[0:31]. Note that in SDRAM mode the address vector contains the full address as {row, bank, column, lsb's} where row corresponds to the same row address for the given column address and lsb's are the unconnected lsb's of the address for a given port size.
- If a valid source ID is detected on LSRCID[0:4] and LDVAL is asserted, valid data may be latched from LAD[0:31].

The LSRCID[0:4] and LDVAL signals are multiplexed with other functions sharing the same external signals. Refer to chapter 3, external signals description and to chapter 5, system configuration to learn how to enable the LSRCID/LDVAL signals.

10.2 LBC External Signal Descriptions

Table 10-1 contains a list of external signals related to the LBC and summarizes their function. The table also shows the reset state of all external signals during assertion of $\overline{\text{HRESET}}$. For more information on the use of some of these signals as reset configuration signals on the MPC8360E, see Section 4.2.2, “Power-On Reset Flow.” Note that during assertion of $\overline{\text{HRESET}}$, the DLL is initially unlocked, so the LCLK and LSYNC_OUT values are likely to be unstable/jittery for several microseconds; after the DLL locks, stable clock signals are driven on these signals.

Table 10-1. Signal Properties—Summary

Name	Alternate Functions	Mode	Descriptions	No. of Signals	I/O	Reset State (Outputs)
LALE	—	—	External address latch enable	1	O	Reset_cfg
$\overline{\text{LCS0}}$	—	—	Chip select 0	1	O	Reset_cfg
$\overline{\text{LCS}}[1:7]$	—	—	Chip selects [1–7]	7	O	All high
$\overline{\text{LWE}}[0:3]/$ LSDDQM/ $\overline{\text{LBS}}[0:3]$	$\overline{\text{LWE}}[0:3]$	GPCM	Write enable	4	O	Reset_cfg
	SDDQM	SDRAM	Byte lane data mask			
	$\overline{\text{LBS}}[0:3]$	UPM	Byte (lane) select			
LGPL0 LSDA10	LGPL0	UPM	General purpose line 0	1	O	Reset_cfg
	LSDA10	SDRAM	Row address bit/command bit			
LGPL1 $\overline{\text{LSDWE}}$	LGPL1	UPM	General purpose line 1	1	O	Reset_cfg
	$\overline{\text{LSDWE}}$	SDRAM	Write enable			
$\overline{\text{LOE}}/$ $\overline{\text{LSDRAS}}/$ LGPL2	$\overline{\text{LOE}}$	GPCM	Output enable	1	O	
	$\overline{\text{LSDRAS}}$	SDRAM	Row address strobe			
	LGPL2	UPM	General purpose line 2			
LGPL3 $\overline{\text{LSDCAS}}$	LGPL3	UPM	General purpose line 3	1	O	Reset_cfg
	$\overline{\text{LSDCAS}}$	SDRAM	Column address strobe			

Table 10-1. Signal Properties—Summary (continued)

Name	Alternate Functions	Mode	Descriptions	No. of Signals	I/O	Reset State (Outputs)
LGTA/ LGPL4/ LUPWAIT/ LPBSE	LGTA	GPCM	Transaction termination	1	I	High-Z
	LGPL4	UPM	General purpose line 4		O	
	LUPWAIT	UPM	External device wait		I	
	LPBSE	—	Local bus parity byte select		O	
LGPL5	—	UPM	General purpose line 5	1	O	Reset_cfg
LBCTL	—	—	Data buffer control	1	O	
LA[27:31]	—	—	Local bus non-multiplexed address lsb's	5	O	
LAD[0:31]	—	—	Multiplexed address/data bus	32	I/O	
LDP	—	—	Local bus data parity	4	I/O	High-Z
LCKE	—	—	Local bus clock enable	1	O	High
LCLK[0:2]	—	—	Local bus clocks	3	O	Driven
LSYNC_IN	—	—	DLL synchronize input	1	I	—
LSYNC_OUT	—	—	DLL synchronize output	1	O	Driven
LDVAL	—	LBC debug	Local bus data valid	1	O	Not connected to external signals
LSRCID[0:4]	—	LBC debug	Local bus source ID	5	O	Not connected to external signals

Table 10-2 shows the detailed external signal descriptions for the LBC.

Table 10-2. Local Bus Controller Detailed Signal Descriptions

Signal	I/O	Description
LALE	O	External address latch enable. The local bus memory controller provides control for an external address latch, which allows address and data to be multiplexed on the device signals. See Section 10.4.1.2 , “External Address Latch Enable Signal (LALE).”
		<table border="1"> <tr> <td>State Meaning</td> <td>Asserted/Negated—LALE is asserted with the address at the beginning of each memory controller transaction. The number of cycles for which it is asserted is governed by the ORn[EAD] and LCRR[EADC] fields. The exact timing of the negation of LALE is controlled by the RCWH[LALE] field. Note that no other control signals are asserted during the assertion of LALE.</td> </tr> </table>
State Meaning	Asserted/Negated—LALE is asserted with the address at the beginning of each memory controller transaction. The number of cycles for which it is asserted is governed by the ORn[EAD] and LCRR[EADC] fields. The exact timing of the negation of LALE is controlled by the RCWH[LALE] field. Note that no other control signals are asserted during the assertion of LALE.	
LCS[0:7]	O	Chip selects. Eight chip selects are provided which are mutually exclusive.
		<table border="1"> <tr> <td>State Meaning</td> <td>Asserted/Negated—Used to enable specific memory devices or peripherals connected to the LBC. LCS[0:7] are provided on a per-bank basis with LCS0 corresponding to the chip select for memory bank 0, which has the memory type and attributes defined by BR0 and OR0.</td> </tr> </table>
State Meaning	Asserted/Negated—Used to enable specific memory devices or peripherals connected to the LBC. LCS[0:7] are provided on a per-bank basis with LCS0 corresponding to the chip select for memory bank 0, which has the memory type and attributes defined by BR0 and OR0.	

Table 10-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{LWE}}[0:3]/$ $\text{LSDDQM}[0:3]/$ $\overline{\text{LBS}}[0:3]$	O	GPCM write enable/SDRAM data mask/UPM byte select. These signals select or validate each byte lane of the data bus. For banks with port sizes of 32 bits (as set by $\text{BRn}[\text{PS}]$), all four signals are defined. For a 16-bit port size, only bits 0–1 are defined; and for an 8-bit port size, bit 0 is the only defined signal. The least-significant address bits of each access also determine which byte lanes are considered valid for a given data transfer.	
		State Meaning	Asserted/Negated—For GPCM operation, $\overline{\text{LWE}}[0:3]$ assert for each byte lane enabled for writing. For SDRAM operation, $\text{LSDDQM}[0:3]$ function as the DQM or data mask signals provided by JEDEC-compliant SDRAM devices, with one DQM provided per byte lane. $\text{LSDDQM}[0:3]$ are driven high when the LBC wishes to mask a write or disable read data output from the SDRAM. $\overline{\text{LBS}}[0:3]$ are programmable byte-select signals in UPM mode. See Section 10.4.4.4, “UPM RAM Array,” for programming details about $\overline{\text{LBS}}[0:3]$.
		Timing	Assertion/Negation—See Section 10.4.2, “General-Purpose Chip-Select Machine (GPCM),” for details regarding the timing of $\overline{\text{LWE}}[0:3]$.
$\text{LSDA10}/$ LGPL0	O	SDRAM A10/General purpose line 0	
		State Meaning	Asserted/Negated—For SDRAM accesses, represents address bit 10. When the row address is driven, it drives the value of address bit 10. When the column address is driven, it forms part of the SDRAM command. One of six general purpose signals when in UPM mode; it drives a value programmed in the UPM array.
$\overline{\text{LSDWE}}/$ LGPL1	O	SDRAM write enable/General-purpose line 1	
		State Meaning	Asserted/Negated—Should be connected to the SDRAM device WE input. Acts as the SDRAM write enable when accessing SDRAM. One of six general purpose signals when in UPM mode, and drives a value programmed in the UPM array.
$\overline{\text{LOE}}/$ $\text{LSDRAS}/$ LGPL2	O	GPCM output enable/SDRAM RAS/General-purpose line 2	
		State Meaning	Asserted/Negated—Controls the output buffer of memory when accessing memory/devices in GPCM mode. For SDRAM accesses, it is the row address strobe (RAS). One of six general purpose lines when in UPM mode; it drives a value programmed in the UPM array.
$\overline{\text{LSDCAS}}/$ LGPL3	O	SDRAM CAS/General-purpose line 3	
		State Meaning	Asserted/Negated—In SDRAM mode, drives the column address strobe (CAS). This signal is one of six general purpose signals when in UPM mode, and drives a value programmed in the UPM array.

Table 10-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description
$\overline{\text{LGTA}}$ / LGPL4/ LUPWAIT/ LPBSE	I/O	GPCM transfer acknowledge/General-purpose line 4/UPM wait/parity byte select
		<p>State Meaning Asserted/Negated—Input in GPCM mode used for transaction termination. It may also be configured as one of six general-purpose output signals when in UPM mode or as an input to force the UPM controller to wait for the memory/device.</p> <p>When configured as LPBSE, it disables any use in GPCM or UPM modes. Because systems that use read-modify-write parity require an additional memory device, they must generate a byte-select like a normal data device. ANDing $\overline{\text{LBS}}[0:3]$ through external logic to achieve the logical function of this byte-select adds a delay to the byte-select path that can affect memory access timing. The LBC provides this optional byte-select signal that is an internal AND of the four (active low) byte selects, allowing glueless, faster connection to n -parity devices.</p> <p>Note: The $\overline{\text{LGTA}}$/LUPWAIT signal can be sampled as asserted (active-low) during LGPL4/LPBSE is brought low by the UPM. For a subsequent GPCM transaction, it functions as $\overline{\text{LGTA}}$/LUPWAIT. As a result, GPCM transactions may be terminated prematurely before $\overline{\text{LGTA}}$/LUPWAIT has drifted to a logical one.</p> <p>Work Around: One way to resolve this issue is to ensure that $\overline{\text{LGTA}}$/LGPL4 is pulled-up to 3.3 V by an external 1-KΩ register. This will ensure that $\overline{\text{LGTA}}$ is sampled high (not asserted) by the time any subsequent GPCM transactions are initiated by the local bus memory controller. If this signal is used purely as an input ($\overline{\text{LGTA}}$/LUPWAIT), a weaker (10-KΩ) pull-up register may be substituted; alternatively, if this signal is used as LPBSE, no pull-up is required, as $\overline{\text{LGTA}}$/LUPWAIT are disabled.</p> <p>A software work around to this errata is to program UPM. Therefore, it can pre-drive LGPL4 high prior to switching to input mode. A weak (10-KΩ or greater) pull-up is still required to maintain a stable level on $\overline{\text{LGTA}}$ for GPCM purposes.</p>
LGPL5	O	General-purpose line 5
		<p>State Meaning Asserted/Negated—One of six general purpose signals when in UPM mode, and drives a value programmed in the UPM array.</p>
LBCTL	O	Data buffer control. The memory controller activates LBCTL for the local bus when a GPCM- or UPM-controlled bank is accessed. Access to an SDRAM machine-controlled bank does not activate the buffer control. Buffer control is disabled by setting $\text{ORn}[\text{BCTLD}]$.
		<p>State Meaning Asserted/Negated—Normally functions as a write/$\overline{\text{read}}$ control for a bus transceiver connected to the LAD lines. Note that an external data buffer must not drive the LAD lines in conflict with the LBC when LBCTL is high, because LBCTL remains high after reset and during address phases.</p>
LA[27:31]	O	Local bus nonmultiplexed address lsb's. All bits driven on LA[27:31] are defined for 8-bit port sizes. For 32-bit port sizes, LA[30:31] are don't cares; for 16-bit port sizes LA31 is a don't care.
		<p>State Meaning Asserted/Negated—Although the LBC shares an address and data bus, up to five lsb's of the RAM address always appear on the dedicated address signals, LA[27:31]. These may be used, unlatched, in place of LAD[27:31] to connect the five lsb's of the address for address phases. For some RAM devices, such as fast-page DRAM, LA[27:31] serve as the column address offset during a burst access.</p>

Table 10-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LAD[0:31]	I/O	Multiplexed address/data bus. For configuration of a port size in BR _n [PS] as 32 bits, all of LAD[0:31] must be connected to the external RAM data bus, with LAD[0:7] occupying the most significant byte lane (at address offset 0). For a port size of 16 bits, LAD[0:7] connect to the most significant byte lane (at address offset 0), while LAD[8:15] connect to the least significant byte lane (at address offset 1); LAD[16:31] are unused for 16-bit port sizes. For a port size of 8 bits, only LAD[0:7] are connected to the external RAM.	
		State Meaning	Asserted/Negated—LAD[0:31] is the shared 32-bit address/data bus through which external RAM devices transfer data and receive addresses
		Timing	Assertion/Negation—During assertion of LALE, LAD[0:31] are driven with the RAM address for the access to follow. External logic should propagate the address on LAD[0:31] while LALE is asserted, and latch the address upon negation of LALE. After LALE is negated, LAD[0:31] are either driven by write data or are made high-impedance by the LBC in order to sample read data driven by an external device. Following the last data transfer of a write access, LAD[0:31] are again taken into a high-impedance state.
LDP[0:3]	I/O	Local bus data parity. Drives and receives the data parity corresponding with the data phases on LAD[0:31].	
		State Meaning	Asserted/Negated—During write accesses, a parity bit is generated for each 8 bits of LAD[0:31], such that LDP0 is even/odd parity for LAD[0:7], while LDP[3] is even/odd parity for LAD[24:31]. Unused byte lanes for port sizes less than 32 bits have undefined parity.
		Timing	Assertion/Negation—Drive and receive the data parity corresponding with the data phases on LAD[0:31]. For read accesses, the parity bits for each byte lane are sampled on LDP[0:3] with the same timing that read data is sampled on LAD[0:31]. LDP[0:3] change impedance in concert with LAD[0:31].
LCKE	O	Local bus clock enable	
		State Meaning	Asserted/Negated—Bus clock enable signal (CKE) for JEDEC-standard SDRAM devices. Asserted during normal SDRAM operation.
LCLK[0:2]	O	Local bus clocks. Enabled by the DLLCK register. See Section 18.4.3, “DLL Clock Register (DLLCK),” for details.	
		State Meaning	Asserted/Negated—LCLK[0:2] drive an identical bus clock signal for distributed loads. If the LBC DLL is enabled (see LCRR[DBYP], Figure 10-19 on page 10-31), the bus clock phase is shifted earlier than transitions on other LBC signals (such as LAD[0:310:15] and $\overline{LCS_n}$) by a time delay matching the delay of the DLL timing loop set up between LSYNC_OUT and LSYNC_IN. DLLLCK
LSYNC_OUT	O	DLL synchronization out.	
		State Meaning	Asserted/Negated—A replica of the bus clock, appearing on LSYNC_OUT, should be propagated through a passive timing loop and returned to LSYNC_IN for achieving correct DLL lock.
		Timing	Assertion/Negation—The time delay of the timing loop should be such that it compensates for the round-trip flight time of LCLK[0:2] and clocked drivers in the system. No load other than a timing loop should be placed on LSYNC_OUT.
LSYNC_IN	I	DLL synchronization in.	
		State Meaning	Asserted/Negated—See description of LSYNC_OUT.

Table 10-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LDVAL	O	Local bus data valid (LBC debug mode only)	
		State Meaning	Asserted/Negated—For a read, LDVAL asserts for one bus cycle in the cycle immediately preceding the sampling of read data on LAD[0:31]. For a write, LDVAL asserts for one bus cycle during the final cycle for which the current write data on LAD[0:31] is valid. During burst transfers, LDVAL asserts for each data beat.
		Timing	Assertion/Negation—Valid only while the LBC is in system debug mode. In debug mode, LDVAL asserts when the LBC generates a data transfer acknowledge.
LSRCID[0:4]	O	Local bus source ID (LBC debug mode only). In debug mode, all LSRCID[0:4] signals are driven high unless LSRCID[0:4] is driving a debug source ID for identifying the internal system device controlling the LBC.	
		State Meaning	Asserted/Negated—Remain high until the last bus cycle of the assertion of LALE, in which case the source ID of the address is indicated, or until LDVAL is asserted, in which case the source ID relating to the data transfer is indicated. In case of address debug, LSRCID[0:4] is valid only when the address on LAD[0:31] consists of all physical address bits—with optional padding—for reconstructing the system address presented to the LBC. For example, LSRCID[0:4] is valid only during CAS phases of SDRAM accesses, because the column, bank select, and (normally unused) row address bits are all present on LAD[0:31] during a CAS cycle.

10.3 LBC Memory Map/Register Definition

Table 10-3 shows the memory-mapped registers of the LBC. Undefined 4-byte address spaces within offset 0x000–0xFFFF are reserved.

Table 10-3. Local Bus Controller Memory Map

Address Offset	Use	Access	Reset	Section/Page
0x0_5000	BR0—Base register 0	R/W	0x0000_RR01 ¹	10.3.1.1/10-11
0x0_5008	BR1—Base register 1		All zeros	
0x0_5010	BR2—Base register 2			
0x0_5018	BR3—Base register 3			
0x0_5020	BR4—Base register 4			
0x0_5028	BR5—Base register 5			
0x0_5030	BR6—Base register 6			
0x0_5038	BR7—Base register 7			

Table 10-3. Local Bus Controller Memory Map (continued)

Address Offset	Use	Access	Reset	Section/Page
0x0_5004	OR0—Option register 0	R/W	0x0000_0FF7	10.3.1.2/10-12
0x0_500C	OR1—Option register 1		All zeros	
0x0_5014	OR2—Option register 2			
0x0_501C	OR3—Option register 3			
0x0_5024	OR4—Option register 4			
0x0_502C	OR5—Option register 5			
0x0_5034	OR6—Option register 6			
0x0_503C	OR7—Option register 7			
0x0_5068	MAR—UPM address register	R/W	All zeros	10.3.1.3/10-18
0x0_5070	MAMR—UPMA mode register	R/W	All zeros	10.3.1.4/10-19
0x0_5074	MBMR—UPMB mode register	R/W	All zeros	10.3.1.4/10-19
0x0_5078	MCMR—UPMC mode register	R/W	All zeros	10.3.1.4/10-19
0x0_5084	MRTPR—Memory refresh timer prescaler register	R/W	All zeros	10.3.1.5/10-21
0x0_5088	MDR—UPM data register	R/W	All zeros	10.3.1.6/10-22
0x0_5094	LSDMR—SDRAM mode register	R/W	All zeros	10.3.1.7/10-22
0x0_50A0	LURT—UPM refresh timer	R/W	All zeros	10.3.1.8/10-24
0x0_50A4	LSRT—SDRAM refresh timer	R/W	All zeros	10.3.1.9/10-25
0x0_50B0	LTESR—Transfer error status register	Read/bit-reset	All zeros	10.3.1.10/10-26
0x0_50B4	LTEDR—Transfer error disable register	R/W	All zeros	10.3.1.11/10-27
0x0_50B8	LTEIR—Transfer error interrupt register	R/W	All zeros	10.3.1.12/10-28
0x0_50BC	LTEATR—Transfer error attributes register	R/W	All zeros	10.3.1.13/10-29
0x0_50C0	LTEAR—Transfer error address register	R/W	All zeros	10.3.1.14/10-30
0x0_50D0	LBCR—Configuration register	R/W	All zeros	10.3.1.15/10-30
0x0_50D4	LCRR—Clock ratio register	R/W	0x8000_0008	10.3.1.16/10-31

¹ Port size for BR0 is configured from the value of RCWH[ROMLOC], which is loaded during reset, hence 'RR' is either 0x08, 0x10, or 0x18.

10.3.1 LBC Register Descriptions

This section provides a detailed description of the LBC configuration, status, and control registers with detailed bit and field descriptions.

Address offsets in the LBC address range that are not defined in [Table 10-3](#) should not be accessed for reading or writing. Similarly, only zero should be written to reserved bits of defined registers, as writing ones can have unpredictable results in some cases.

Table 10-4. BR_n Field Descriptions (continued)

Bits	Name	Description
23	WP	Write protect. 0 Read and write accesses are allowed. 1 Only read accesses are allowed. The memory controller does not assert \overline{LCS}_n on write cycles to this memory bank. LTESR[WP] is set (if WP is set) if a write to this memory bank is attempted, and a local bus error interrupt is generated (if enabled), terminating the cycle.
24–26	MSEL	Machine select. Specifies the machine to use for handling memory operations. 000 GPCM (reset value) 001 Reserved 010 Reserved 011 SDRAM 100 UPMA 101 UPMB 110 UPMC 111 Reserved
27	—	Reserved
28–29	ATOM	Atomic operation. Writes (reads) to the address space handled by the memory controller bank reserve the selected memory bank for the exclusive use of the accessing device. The reservation is released when the device performs a read (write) operation to this memory controller bank. If a subsequent read (write) request to this memory controller bank is not detected within 256 bus clock cycles of the last write (read), the reservation is released and an atomic error is reported (if enabled). 00 The address space controlled by this bank is not used for atomic operations. 01 Read-after-write-atomic (RAWA). 10 Write-after-read-atomic (WARA). 11 Reserved
30	—	Reserved
31	V	Valid bit. Indicates that the contents of the BR _n and OR _n pair are valid. \overline{LCS}_n does not assert unless V is set (an access to a region that has no valid bit set may cause a bus time-out). After a system reset, only BR0[V] is set. 0 This bank is invalid. 1 This bank is valid.

10.3.1.2 Option Registers (OR0–OR7)

The OR_n registers define the sizes of memory banks and access attributes. The OR_n attribute bits support the following three modes of operation as defined by BR_n[MSEL].

- GPCM mode (see [Section 10.3.1.2.2, “Option Registers \(OR_n\)—GPCM Mode.”](#))
- UPM mode (see [Section 10.3.1.2.3, “Option Registers \(OR_n\)—UPM Mode.”](#))
- SDRAM mode (see [Section 10.3.1.2.4, “Option Registers \(OR_n\)—SDRAM Mode.”](#))

The OR_n registers are interpreted differently depending on which of the machine types is selected for that bank.

10.3.1.2.1 Address Mask

The address mask fields of the option registers (OR_n[AM]) mask up to 17 bits of the corresponding BR_n[BA] fields. The 15 lsbs of the 32-bit internal address do not participate in bank address matching in

selecting a bank for access. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. Table 10-5 shows memory bank sizes from 256 Kbytes to 4 Gbytes.

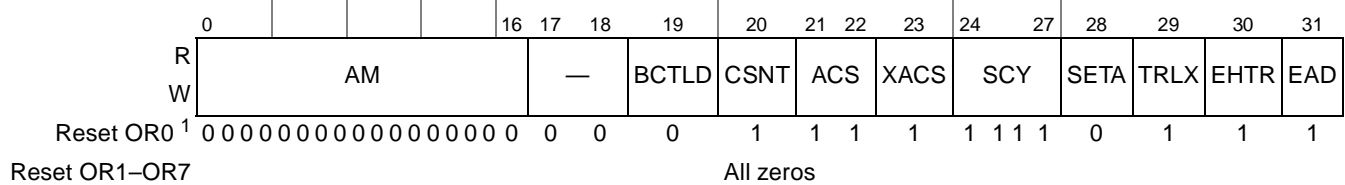
Table 10-5. Memory Bank Sizes in Relation to Address Mask

AM	Memory Bank Size
0000_0000_0000_0000_0	4 Gbytes
1000_0000_0000_0000_0	2 Gbytes
1100_0000_0000_0000_0	1 Gbyte
1110_0000_0000_0000_0	512 Mbytes
1111_0000_0000_0000_0	256 Mbytes
1111_1000_0000_0000_0	128 Mbytes
1111_1100_0000_0000_0	64 Mbytes
1111_1110_0000_0000_0	32 Mbytes
1111_1111_0000_0000_0	16 Mbytes
1111_1111_1000_0000_0	8 Mbytes
1111_1111_1100_0000_0	4 Mbytes
1111_1111_1110_0000_0	2 Mbytes
1111_1111_1111_0000_0	1 Mbyte
1111_1111_1111_1000_0	512 Kbytes
1111_1111_1111_1100_0	256 Kbytes
1111_1111_1111_1110_0	128 Kbytes
1111_1111_1111_1111_0	64 Kbytes
1111_1111_1111_1111_1	32 Kbytes

10.3.1.2.2 Option Registers (OR_n)—GPCM Mode

Figure 10-3 shows the bit fields for OR_n when the corresponding BR_n[MSEL] selects the GPCM machine.

Offset 0x0_5004 (OR0) 0x0_5024 (OR4) Access: read/write
 0x0_500C (OR1) 0x0_502C (OR5)
 0x0_5014 (OR2) 0x0_5034 (OR6)
 0x0_501C (OR3) 0x0_503C (OR7)



¹ OR0 has this value set during reset (GPCM is the default control machine for all banks coming out of reset). All other option registers have all bits cleared.

Figure 10-3. Option Registers (OR_n) in GPCM Mode

Table 10-6 describes OR_n fields for GPCM mode.

Table 10-6. OR_n—GPCM Field Descriptions

Bits	Name	Description																		
0–16	AM	<p>GPCM address mask. Masks corresponding BR_n bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map.</p> <p>0 Corresponding address bits are masked. 1 Corresponding address bits are used in the comparison between base and transaction addresses. See Section 10.3.1.2.1, “Address Mask.”</p>																		
17–18	—	Reserved																		
19	BCTLD	<p>Buffer control disable. Disables assertion of LBCTL during access to the current memory bank.</p> <p>0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.</p>																		
20	CSNT	<p>Chip select negation time. Determines when \overline{LCSn} and \overline{LWE} are negated during an external memory write access handled by the GPCM, provided that ACS ≠ 00 (when ACS = 00, only \overline{LWE} is affected by the setting of CSNT). This helps meet address/data hold times for slow memories and peripherals.</p> <p>0 \overline{LCSn} and \overline{LWE} are negated normally. 1 \overline{LCSn} and \overline{LWE} are negated earlier depending on the value of LCRR[CLKDIV].</p> <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>LCRR[CLKDIV]</th> <th>CSNT</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>x</td> <td>0</td> <td>\overline{LCSn} and \overline{LWE} are negated normally.</td> </tr> <tr> <td>2</td> <td>1</td> <td>\overline{LCSn} and \overline{LWE} are negated normally.</td> </tr> <tr> <td>4 or 8</td> <td>1</td> <td>\overline{LCSn} and \overline{LWE} are negated quarter of a bus clock cycle earlier.</td> </tr> </tbody> </table>	LCRR[CLKDIV]	CSNT	Meaning	x	0	\overline{LCSn} and \overline{LWE} are negated normally.	2	1	\overline{LCSn} and \overline{LWE} are negated normally.	4 or 8	1	\overline{LCSn} and \overline{LWE} are negated quarter of a bus clock cycle earlier.						
LCRR[CLKDIV]	CSNT	Meaning																		
x	0	\overline{LCSn} and \overline{LWE} are negated normally.																		
2	1	\overline{LCSn} and \overline{LWE} are negated normally.																		
4 or 8	1	\overline{LCSn} and \overline{LWE} are negated quarter of a bus clock cycle earlier.																		
21–22	ACS	<p>Address to chip-select setup. Determines the delay of the \overline{LCSn} assertion relative to the address change when the external memory access is handled by the GPCM. At system reset, OR0[ACS] = 11</p> <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>LCRR[CLKDIV]</th> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td rowspan="2">x</td> <td>00</td> <td>\overline{LCSn} is output at the same time as the address lines. Note that this overrides the value of CSNT such that CSNT=0.</td> </tr> <tr> <td>01</td> <td>Reserved.</td> </tr> <tr> <td rowspan="2">2</td> <td>10</td> <td>\overline{LCSn} is output a half bus clock cycle after the address lines.</td> </tr> <tr> <td>11</td> <td>\overline{LCSn} is output a half bus clock cycle after the address lines.</td> </tr> <tr> <td rowspan="2">4 or 8</td> <td>10</td> <td>\overline{LCSn} is output a quarter bus clock cycle after the address lines.</td> </tr> <tr> <td>11</td> <td>\overline{LCSn} is output a half bus clock cycle after the address lines.</td> </tr> </tbody> </table>	LCRR[CLKDIV]	Value	Meaning	x	00	\overline{LCSn} is output at the same time as the address lines. Note that this overrides the value of CSNT such that CSNT=0.	01	Reserved.	2	10	\overline{LCSn} is output a half bus clock cycle after the address lines.	11	\overline{LCSn} is output a half bus clock cycle after the address lines.	4 or 8	10	\overline{LCSn} is output a quarter bus clock cycle after the address lines.	11	\overline{LCSn} is output a half bus clock cycle after the address lines.
LCRR[CLKDIV]	Value	Meaning																		
x	00	\overline{LCSn} is output at the same time as the address lines. Note that this overrides the value of CSNT such that CSNT=0.																		
	01	Reserved.																		
2	10	\overline{LCSn} is output a half bus clock cycle after the address lines.																		
	11	\overline{LCSn} is output a half bus clock cycle after the address lines.																		
4 or 8	10	\overline{LCSn} is output a quarter bus clock cycle after the address lines.																		
	11	\overline{LCSn} is output a half bus clock cycle after the address lines.																		
23	XACS	<p>Extra address to chip-select setup. Setting this bit increases the delay of the \overline{LCSn} assertion relative to the address change when the external memory access is handled by the GPCM. After a system reset, OR0[XACS] = 1.</p> <p>0 Address to chip-select setup is determined by OR_x[ACS] and LCRR[CLKDIV]. 1 Address to chip-select setup is extended (see Table 10-23 and Table 10-24 for LCRR[CLKDIV] = 4 or 8, Table 10-25 and Table 10-26 for LCRR[CLKDIV] = 2).</p>																		

Table 10-6. OR_n—GPCM Field Descriptions (continued)

Bits	Name	Description															
24–27	SCY	Cycle length in bus clocks. Determines the number of wait states inserted in the bus cycle, when the GPCM handles the external memory access. Thus it is the main parameter for determining cycle length. The total cycle length depends on other timing attribute settings. After a system reset, OR0[SCY] = 1111. 0000 No wait states 0001 1 bus clock cycle wait state ... 1111 15 bus clock cycle wait states															
28	SETA	External address termination. 0 Access is terminated internally by the memory controller unless the external device asserts $\overline{\text{LGTA}}$ earlier to terminate the access. 1 Access is terminated externally by asserting the $\overline{\text{LGTA}}$ external signal. (Only $\overline{\text{LGTA}}$ can terminate the access).															
29	TRLX	Timing relaxed. Modifies the settings of timing parameters for slow memories or peripherals. 0 Normal timing is generated by the GPCM. 1 Relaxed timing on the following parameters: <ul style="list-style-type: none"> • Adds an additional cycle between the address and control signals (only if ACS is not equal to 00). • Doubles the number of wait states specified by SCY, providing up to 30 wait states. • Works in conjunction with EHTR to extend hold time on read accesses. • $\overline{\text{LCSn}}$ (only if ACS is not equal to 00) and $\overline{\text{LWE}}$ signals are negated one cycle earlier during writes. 															
30	EHTR	Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access. <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>TRLX</th> <th>EHTR</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>The memory controller generates normal timing. No additional cycles are inserted.</td> </tr> <tr> <td>0</td> <td>1</td> <td>1 idle clock cycle is inserted.</td> </tr> <tr> <td>1</td> <td>0</td> <td>4 idle clock cycles are inserted.</td> </tr> <tr> <td>1</td> <td>1</td> <td>8 idle clock cycles are inserted.</td> </tr> </tbody> </table>	TRLX	EHTR	Meaning	0	0	The memory controller generates normal timing. No additional cycles are inserted.	0	1	1 idle clock cycle is inserted.	1	0	4 idle clock cycles are inserted.	1	1	8 idle clock cycles are inserted.
TRLX	EHTR	Meaning															
0	0	The memory controller generates normal timing. No additional cycles are inserted.															
0	1	1 idle clock cycle is inserted.															
1	0	4 idle clock cycles are inserted.															
1	1	8 idle clock cycles are inserted.															
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).															

10.3.1.2.3 Option Registers (OR_n)—UPM Mode

Figure 10-4 shows the bit fields for OR_n when the corresponding BR_n[MSEL] selects a UPM machine.

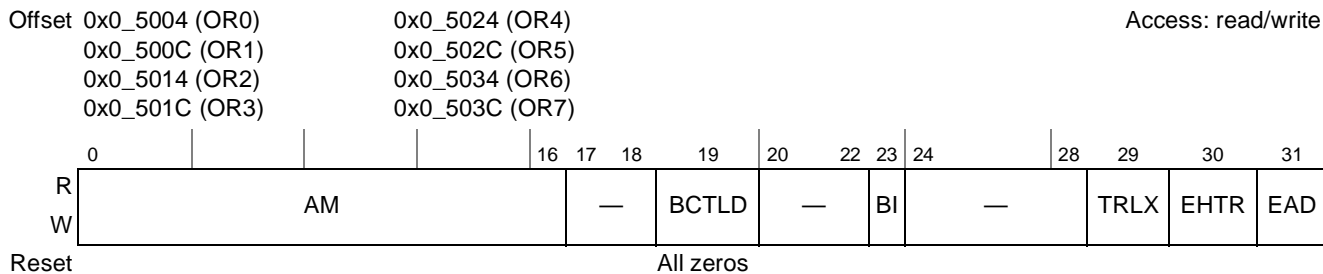


Figure 10-4. Option Registers (OR_n) in UPM Mode

Table 10-7 describes OR_n fields for UPM mode.

Table 10-7. OR_n—UPM Field Descriptions

Bits	Name	Description
0–16	AM	UPM address mask. Masks corresponding BR _n bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. 0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address signals.
17–18	—	Reserved
19	BCTLD	Buffer control disable. Disables assertion of LBCTL during access to the current memory bank. 0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.
20–22	—	Reserved
23	BI	Burst inhibit. Indicates if this memory bank supports burst accesses. 0 The bank supports burst accesses. 1 The bank does not support burst accesses. The selected UPM executes burst accesses as a series of single accesses.
24–28	—	Reserved
29	TRLX	Timing relaxed. Works in conjunction with EHTR to extend hold time on read accesses.

Table 10-7. OR_n—UPM Field Descriptions (continued)

Bits	Name	Description															
30	EHTR	Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access.															
		<table border="1"> <thead> <tr> <th>TRLX</th> <th>EHTR</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>The memory controller generates normal timing. No additional cycles are inserted.</td> </tr> <tr> <td>0</td> <td>1</td> <td>1 idle clock cycle is inserted.</td> </tr> <tr> <td>1</td> <td>0</td> <td>4 idle clock cycles are inserted.</td> </tr> <tr> <td>1</td> <td>1</td> <td>8 idle clock cycles are inserted.</td> </tr> </tbody> </table>	TRLX	EHTR	Meaning	0	0	The memory controller generates normal timing. No additional cycles are inserted.	0	1	1 idle clock cycle is inserted.	1	0	4 idle clock cycles are inserted.	1	1	8 idle clock cycles are inserted.
		TRLX	EHTR	Meaning													
		0	0	The memory controller generates normal timing. No additional cycles are inserted.													
		0	1	1 idle clock cycle is inserted.													
1	0	4 idle clock cycles are inserted.															
1	1	8 idle clock cycles are inserted.															
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).															

10.3.1.2.4 Option Registers (OR_n)—SDRAM Mode

Figure 10-5 shows the bit fields for OR_n when the corresponding BR_n[MSEL] selects the SDRAM machine.

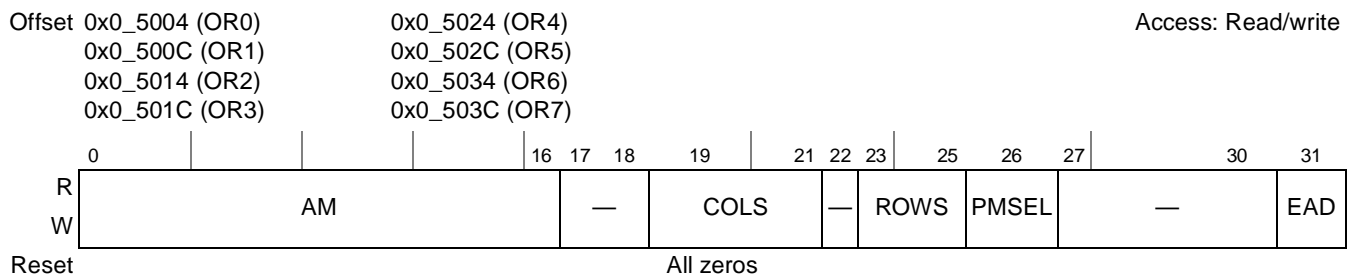


Figure 10-5. Option Registers (OR_n) in SDRAM Mode

Table 10-8 describes OR_n fields for SDRAM mode.

Table 10-8. OR_n—SDRAM Field Descriptions

Bits	Name	Description
0–16	AM	SDRAM address mask. Masks corresponding BR _n bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. AM can be read or written at any time. 0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address signals.
17–18	—	Reserved
19–21	COLS	Number of column address lines. Sets the number of column address lines in the SDRAM device. 000 7 100 11 001 8 101 12 010 9 110 13 011 10 111 14

Table 10-8. OR_n—SDRAM Field Descriptions (continued)

Bits	Name	Description
22	—	Reserved
23–25	ROWS	Number of row address lines. Sets the number of row address lines in the SDRAM device. 000 9 100 13 001 10 101 14 010 11 110 15 011 12 111 Reserved
26	PMSEL	Page mode select. Selects page mode for the SDRAM connected to the memory controller bank. 0 Back-to-back page mode (normal operation). Page is closed when the bus becomes idle. 1 Page is kept open until a page miss or refresh occurs.
27–30	—	Reserved
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).

10.3.1.3 UPM Memory Address Register (MAR)

Figure 10-6 shows the fields of the UPM memory address register (MAR).

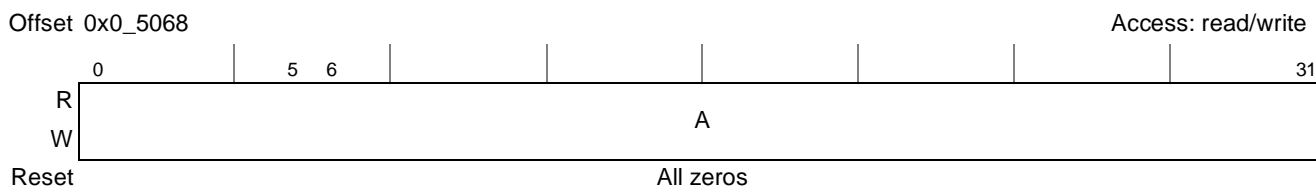


Figure 10-6. UPM Memory Address Register (MAR)

Table 10-9 describes the MAR fields.

Table 10-9. MAR Field Descriptions

Bits	Name	Description
0–31	A	Address that can be output to the address signals under control of the AMX bits in the UPM RAM word.

10.3.1.4 UPM Mode Registers (MnMR)

The UPM machine mode registers (MAMR, MBMR, and MCMR), shown in Figure 10-7, contain the configuration for the three UPMs.

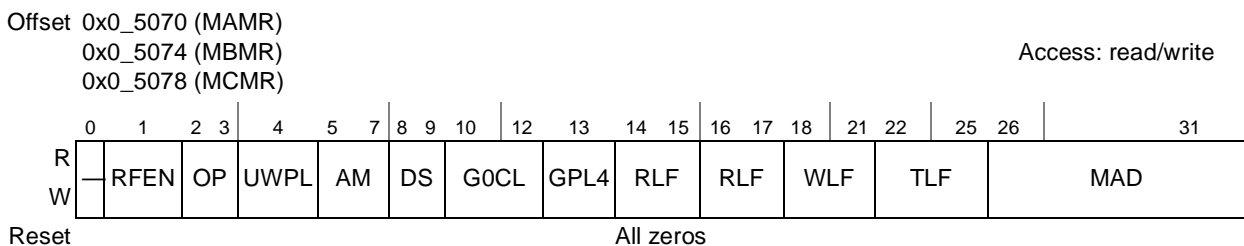


Figure 10-7. UPM Mode Registers (MnMR)

Table 10-10 describes UPM mode fields.

Table 10-10. MnMR Field Descriptions

Bits	Name	Description
0	—	Reserved
1	RFEN	Refresh enable. Indicates that the UPM needs refresh services. This bit must be set for UPMA (refresh executor) if refresh services are required on any UPM assigned chip selects. If MAMR[RFEN] = 0, no refresh services can be provided, even if UPMB and/or UPMC have their RFEN bit set. 0 Refresh services are not required 1 Refresh services are required
2–3	OP	Command opcode. Determines the command executed by the UPM _n when a memory access hits a UPM assigned bank. 00 Normal operation 01 Write to UPM array. On the next memory access that hits a UPM assigned bank, write the contents of the MDR into the RAM location pointed to by MAD. After the access, MAD is automatically incremented. 10 Read from UPM array. On the next memory access that hits a UPM assigned bank, read the contents of the RAM location pointed to by MAD into the MDR. After the access, MAD is automatically incremented. 11 Run pattern. On the next memory access that hits a UPM assigned bank, run the pattern written in the RAM array. The pattern run starts at the location pointed to by MAD and continues until the LAST bit is set in the RAM word.
4	UWPL	LUPWAIT polarity active low. Sets the polarity of the LUPWAIT signal when in UPM mode. 0 LUPWAIT is active high. 1 LUPWAIT is active low.

Table 10-10. MnMR Field Descriptions (continued)

Bits	Name	Description																																																																								
5–7	AM	Address multiplex size. Determines how the address of the current memory cycle can be output on the address signals. This field is needed when interfacing with devices requiring row and column addresses multiplexed on the same signals.																																																																								
		<table border="1"> <thead> <tr> <th>Value</th> <th>LA0–LA15</th> <th>LA16</th> <th>LA17</th> <th>LA18</th> <th>LA19–LA28</th> <th>LA29</th> <th>LA30</th> <th>LA31</th> </tr> </thead> <tbody> <tr> <td>000</td> <td>0</td> <td>A8</td> <td>A9</td> <td>A10</td> <td>A11–A20</td> <td>A21</td> <td>A22</td> <td>A23</td> </tr> <tr> <td>001</td> <td>0</td> <td>A7</td> <td>A8</td> <td>A9</td> <td>A10–A19</td> <td>A20</td> <td>A21</td> <td>A22</td> </tr> <tr> <td>010</td> <td>0</td> <td>A6</td> <td>A7</td> <td>A8</td> <td>A9–A18</td> <td>A19</td> <td>A20</td> <td>A21</td> </tr> <tr> <td>011</td> <td>0</td> <td>A5</td> <td>A6</td> <td>A7</td> <td>A8–A17</td> <td>A18</td> <td>A19</td> <td>A20</td> </tr> <tr> <td>100</td> <td>0</td> <td>A4</td> <td>A5</td> <td>A6</td> <td>A7–A16</td> <td>A17</td> <td>A18</td> <td>A19</td> </tr> <tr> <td>101</td> <td>0</td> <td>A3</td> <td>A4</td> <td>A5</td> <td>A6–A15</td> <td>A16</td> <td>A17</td> <td>A18</td> </tr> <tr> <td>110–111</td> <td colspan="8">Reserved</td> </tr> </tbody> </table>	Value	LA0–LA15	LA16	LA17	LA18	LA19–LA28	LA29	LA30	LA31	000	0	A8	A9	A10	A11–A20	A21	A22	A23	001	0	A7	A8	A9	A10–A19	A20	A21	A22	010	0	A6	A7	A8	A9–A18	A19	A20	A21	011	0	A5	A6	A7	A8–A17	A18	A19	A20	100	0	A4	A5	A6	A7–A16	A17	A18	A19	101	0	A3	A4	A5	A6–A15	A16	A17	A18	110–111	Reserved							
		Value	LA0–LA15	LA16	LA17	LA18	LA19–LA28	LA29	LA30	LA31																																																																
		000	0	A8	A9	A10	A11–A20	A21	A22	A23																																																																
		001	0	A7	A8	A9	A10–A19	A20	A21	A22																																																																
		010	0	A6	A7	A8	A9–A18	A19	A20	A21																																																																
		011	0	A5	A6	A7	A8–A17	A18	A19	A20																																																																
		100	0	A4	A5	A6	A7–A16	A17	A18	A19																																																																
101	0	A3	A4	A5	A6–A15	A16	A17	A18																																																																		
110–111	Reserved																																																																									
8–9	DS	Disable timer period. Guarantees a minimum time between accesses to the same memory bank controlled by UPMn. The disable timer is turned on by the TODT bit in the RAM array word, and when expired, the UPMn allows the machine access to handle a memory pattern to the same bank. Accesses to a different bank by the same UPMn is also allowed. To avoid conflicts between successive accesses to different banks, the minimum pattern in the RAM array for a request serviced should not be shorter than the period established by DS.																																																																								
		00 1-bus clock cycle disable period																																																																								
		01 2-bus clock cycle disable period																																																																								
		10 3-bus clock cycle disable period																																																																								
11 4-bus clock cycle disable period																																																																										
10–12	G0CL	General line 0 control. Determines which logical address line can be output to the LGPL0 signal when the UPMn is selected to control the memory access.																																																																								
		000 A12																																																																								
		001 A11																																																																								
		010 A10																																																																								
		011 A9																																																																								
		100 A8																																																																								
		101 A7																																																																								
		110 A6																																																																								
111 A5																																																																										
13	GPL4	LGPL4 output line disable. Determines how the LGPL4/LUPWAIT signal is controlled by the corresponding bits in the UPMn array. See Table 10-30 on page 10-67 .																																																																								
		<table border="1"> <thead> <tr> <th rowspan="2">Value</th> <th rowspan="2">LGPL4/LUPWAIT Signal Function</th> <th colspan="2">Interpretation of UPM Word Bits</th> </tr> <tr> <th>G4T1/DLT3</th> <th>G4T3/WAEN</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>LGPL4 (output)</td> <td>G4T1</td> <td>G4T3</td> </tr> <tr> <td>1</td> <td>LUPWAIT (input)</td> <td>DLT3</td> <td>WAEN</td> </tr> </tbody> </table>	Value	LGPL4/LUPWAIT Signal Function	Interpretation of UPM Word Bits		G4T1/DLT3	G4T3/WAEN	0	LGPL4 (output)	G4T1	G4T3	1	LUPWAIT (input)	DLT3	WAEN																																																										
		Value			LGPL4/LUPWAIT Signal Function	Interpretation of UPM Word Bits																																																																				
G4T1/DLT3	G4T3/WAEN																																																																									
0	LGPL4 (output)	G4T1	G4T3																																																																							
1	LUPWAIT (input)	DLT3	WAEN																																																																							

Table 10-10. MnMR Field Descriptions (continued)

Bits	Name	Description
14–17	RLF	Read loop field. Determines the number of times a loop defined in the UPMn is executed for a burst- or single-beat read pattern or when MnMR[OP] = 11 (run command) 0000 16 0001 1 0010 2 0011 3 ... 1110 14 1111 15
18–21	WLF	Write loop field. Determines the number of times a loop defined in the UPMn is executed for a burst- or single-beat write pattern. 0000 16 0001 1 0010 2 0011 3 ... 1110 14 1111 15
22–25	TLF	Refresh loop field. Determines the number of times a loop defined in the UPMn is executed for a refresh service pattern. 0000 16 0001 1 0010 2 0011 3 ... 1110 14 1111 15
26–31	MAD	Machine address. RAM address pointer for the command executed. This field is incremented by 1, each time the UPM is accessed and the OP field is set to WRITE or READ. Address range is 64 words per UPMn.

10.3.1.5 Memory Refresh Timer Prescaler Register (MRTPR)

MRTPR shown in [Figure 10-8](#), is used to divide the system clock to provide the SDRAM and UPM refresh timers clock.

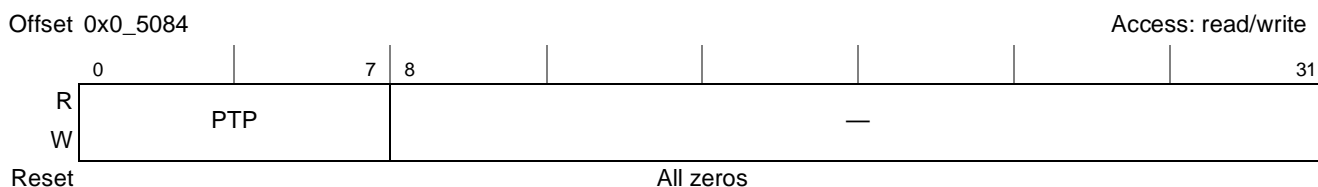


Figure 10-8. Memory Refresh Timer Prescaler Register (MRTPR)

Table 10-11 describes MRTPR fields.

Table 10-11. MRTPR Field Descriptions

Bits	Name	Description
0–7	PTP	Refresh timers prescaler. Determines the period of the refresh timers input clock. The system clock is divided by PTP except when the value is 0x0000_0000, which represents the maximum divider of 256.
8–31	—	Reserved

10.3.1.6 UPM Data Register (MDR)

MDR shown in Figure 10-9, contains data written to or read from the RAM array for UPM read or write commands. MDR must be set up before issuing a write command to the UPM.

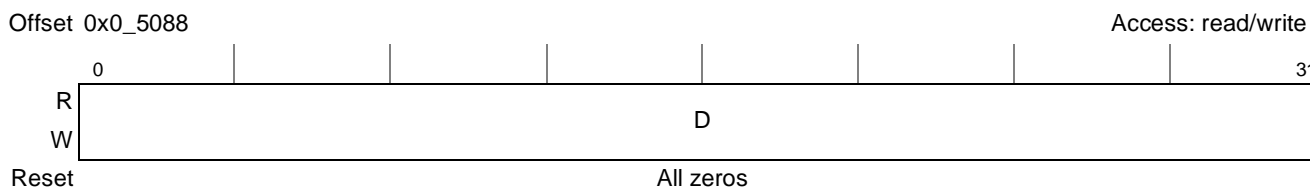


Figure 10-9. UPM Data Register (MDR)

Table 10-12 describes MDR[D].

Table 10-12. MDR Field Description

Bits	Name	Description
0–31	D	The data to be read or written into the RAM array when a write or read command is supplied to the UPM ($MnMR[OP] = 01$ or $MnMR[OP] = 10$).

10.3.1.7 Local Bus SDRAM Machine Mode Register (LSDMR)

LSDMR shown in Figure 10-10, is used to configure operations pertaining to SDRAM.

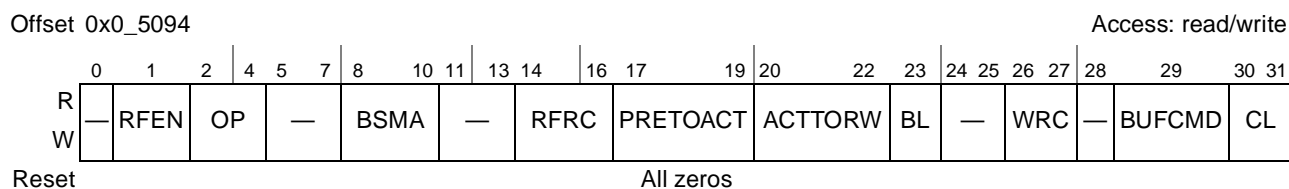


Figure 10-10. Local Bus SDRAM Machine Mode Register (LSDMR)

Table 10-12 describes LSDMR fields.

Table 10-13. LSDMR Field Descriptions

Bits	Name	Description
0	—	Reserved
1	RFEN	Refresh enable. Indicates that the SDRAM requires refresh services. 0 Refresh services are not required. 1 Refresh services are required.

Table 10-13. LSDMR Field Descriptions (continued)

Bits	Name	Description																											
2–4	OP	<p>SDRAM operation. Selects the operation that occurs when the SDRAM device is accessed. See Section 10.4.3.3, “Intel PC133 and JEDEC-Standard SDRAM Interface Commands.”</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>000</td> <td>Normal operation</td> <td>Normal operation</td> </tr> <tr> <td>001</td> <td>Auto refresh</td> <td>Initialization</td> </tr> <tr> <td>010</td> <td>Self refresh</td> <td>Debug</td> </tr> <tr> <td>011</td> <td>Mode register write</td> <td>Initialization</td> </tr> <tr> <td>100</td> <td>Precharge bank</td> <td>Debug</td> </tr> <tr> <td>101</td> <td>Precharge all banks</td> <td>Initialization</td> </tr> <tr> <td>110</td> <td>Activate bank</td> <td>Debug</td> </tr> <tr> <td>111</td> <td>Read/write without valid data transfer</td> <td>Debug</td> </tr> </tbody> </table>	Value	Meaning	Use	000	Normal operation	Normal operation	001	Auto refresh	Initialization	010	Self refresh	Debug	011	Mode register write	Initialization	100	Precharge bank	Debug	101	Precharge all banks	Initialization	110	Activate bank	Debug	111	Read/write without valid data transfer	Debug
Value	Meaning	Use																											
000	Normal operation	Normal operation																											
001	Auto refresh	Initialization																											
010	Self refresh	Debug																											
011	Mode register write	Initialization																											
100	Precharge bank	Debug																											
101	Precharge all banks	Initialization																											
110	Activate bank	Debug																											
111	Read/write without valid data transfer	Debug																											
5–7	—	Reserved																											
8–10	BSMA	<p>Bank select multiplexed address line. Selects which address signals serve as the 2-bit bank-select address for SDRAM. Note that only 4-bank SDRAMs are supported.</p> <table> <tbody> <tr> <td>000</td> <td>LA[12:13]</td> <td>100</td> <td>LA[16:17]</td> </tr> <tr> <td>001</td> <td>LA[13:14]</td> <td>101</td> <td>LA[17:18]</td> </tr> <tr> <td>010</td> <td>LA[14:15]</td> <td>110</td> <td>LA[18:19]</td> </tr> <tr> <td>011</td> <td>LA[15:16]</td> <td>111</td> <td>LA[19:20]</td> </tr> </tbody> </table>	000	LA[12:13]	100	LA[16:17]	001	LA[13:14]	101	LA[17:18]	010	LA[14:15]	110	LA[18:19]	011	LA[15:16]	111	LA[19:20]											
000	LA[12:13]	100	LA[16:17]																										
001	LA[13:14]	101	LA[17:18]																										
010	LA[14:15]	110	LA[18:19]																										
011	LA[15:16]	111	LA[19:20]																										
11–13	—	Reserved																											
14–16	RFRC	<p>Refresh recovery. Sets the refresh recovery interval in bus clock cycles. Defines the earliest timing for an ACTIVATE or REFRESH command after a REFRESH command. See Section 10.4.3.7.5, “Refresh Recovery Interval (RFRC).”</p> <table> <tbody> <tr> <td>000</td> <td>Reserved</td> <td>100</td> <td>6 clocks</td> </tr> <tr> <td>001</td> <td>3 clocks</td> <td>101</td> <td>7 clocks</td> </tr> <tr> <td>010</td> <td>4 clocks</td> <td>110</td> <td>8 clocks</td> </tr> <tr> <td>011</td> <td>5 clocks</td> <td>111</td> <td>16 clocks</td> </tr> </tbody> </table>	000	Reserved	100	6 clocks	001	3 clocks	101	7 clocks	010	4 clocks	110	8 clocks	011	5 clocks	111	16 clocks											
000	Reserved	100	6 clocks																										
001	3 clocks	101	7 clocks																										
010	4 clocks	110	8 clocks																										
011	5 clocks	111	16 clocks																										
17–19	PRETOACT	<p>Defines the earliest timing for ACTIVATE or REFRESH command after a PRECHARGE command (number of bus clock cycle wait states). See Section 10.4.3.7.1, “Precharge-to-Activate Interval.”</p> <table> <tbody> <tr> <td>000</td> <td>8</td> <td>100</td> <td>4</td> </tr> <tr> <td>001</td> <td>1</td> <td>101</td> <td>5</td> </tr> <tr> <td>010</td> <td>2</td> <td>110</td> <td>6</td> </tr> <tr> <td>011</td> <td>3</td> <td>111</td> <td>7</td> </tr> </tbody> </table>	000	8	100	4	001	1	101	5	010	2	110	6	011	3	111	7											
000	8	100	4																										
001	1	101	5																										
010	2	110	6																										
011	3	111	7																										
20–22	ACTTORW	<p>Defines the earliest timing for READ/WRITE command after an ACTIVATE command (number of bus clock cycle wait states). See Section 10.4.3.7.2, “Activate-to-Read/Write Interval.”</p> <table> <tbody> <tr> <td>000</td> <td>8</td> <td>100</td> <td>4</td> </tr> <tr> <td>001</td> <td>Reserved</td> <td>101</td> <td>5</td> </tr> <tr> <td>010</td> <td>2</td> <td>110</td> <td>6</td> </tr> <tr> <td>011</td> <td>3</td> <td>111</td> <td>7</td> </tr> </tbody> </table>	000	8	100	4	001	Reserved	101	5	010	2	110	6	011	3	111	7											
000	8	100	4																										
001	Reserved	101	5																										
010	2	110	6																										
011	3	111	7																										
23	BL	<p>Sets the burst length for SDRAM accesses.</p> <p>0 SDRAM burst length is 4. Use this value if the device port size is 16.</p> <p>1 SDRAM burst length is 8. Use this value if the device port size is 32 or 8.</p>																											

Table 10-13. LSDMR Field Descriptions (continued)

Bits	Name	Description
24–25	—	Reserved
26–27	WRC	Write recovery time. Defines the earliest timing for PRECHARGE command after the last data is written to the SDRAM. See Section 10.4.3.7.3, “Column Address to First Data Out—CAS Latency.” 00 4 01 Reserved 10 2 11 3
28	—	Reserved
29	BUFCMD	Control line assertion timing. If external buffers are placed on the control lines going to both the SDRAM and address lines, setting BUFCMD causes all SDRAM control lines except \overline{LCSn} , LCKE, LALE and $\overline{LSDQM}[0:3]$ to be asserted for LCRR[BUFCMDC] cycles, instead of one. See Section 10.4.3.7.6, “External Address and Command Buffers (BUFCMD).” 0 Normal timing for the control lines 1 All control lines except \overline{LCSn} are asserted for the number of cycles specified by LCRR[BUFCMDC].
30–31	CL	CAS latency. Defines the timing for first read data after SDRAM samples a column address. 00 Extended CAS latency. According to LCRR[ECL]. See Table 10-22. 01 1 10 2 11 3

10.3.1.8 UPM Refresh Timer (LURT)

LURT, shown in [Figure 10-11](#), generates a refresh request for all valid banks that selected a UPM machine and are refresh-enabled ($MnMR[RFEN] = 1$). Each time the timer expires, a qualified bank generates a refresh request using the selected UPM. The qualified banks rotate their requests.

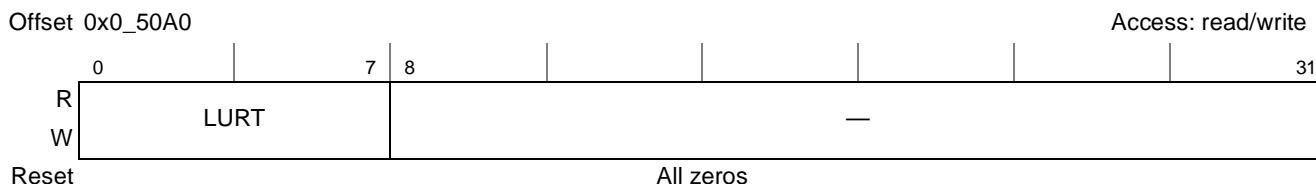


Figure 10-11. UPM Refresh Timer (LURT)

Table 10-16. LTESR Field Descriptions (continued)

Bits	Name	Description
10–11	—	Reserved
12	CS	Chip select error 0 No chip select error occurred. 1 A transaction was sent to the LBC that did not hit any memory bank.
13–31	—	Reserved

10.3.1.11 Transfer Error Check Disable Register (LTEDR)

LTEDR shown in [Figure 10-14](#), is used to disable error checking. Note that control of error checking is independent of control of reporting of errors (LTEIR) through the interrupt mechanism.

Offset 0x0_50B4

Access: read/write

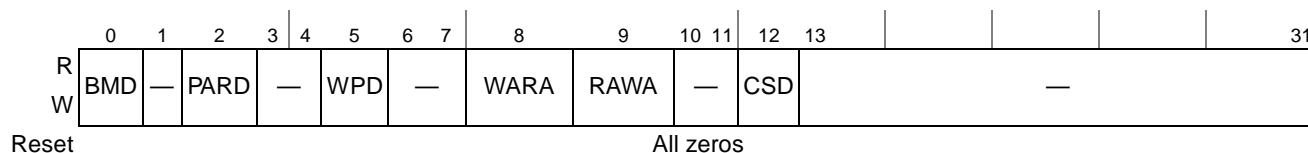


Figure 10-14. Transfer Error Check Disable Register (LTEDR)

[Table 10-17](#) describes LTEDR fields.

Table 10-17. LTEDR Field Descriptions

Bits	Name	Description
0	BMD	Bus monitor disable. 0 Bus monitor is enabled. 1 Bus monitor is disabled.
1	—	Reserved
2	PARD	Parity error checking disabled. 0 Parity error checking is enabled. 1 Parity error checking is disabled.
3–4	—	Reserved
5	WPD	Write protect error checking disable. 0 Write protect error checking is enabled. 1 Write protect error checking is disabled.
6–7	—	Reserved
8	WARA	Write after read atomic (WARA) error checking disable. 0 WARA error checking is enabled. 1 WARA error checking is disabled.
9	RAWA	Read after write atomic (RAWA) error checking disable. 0 RAWA error checking is enabled. 1 RAWA error checking is disabled.
10–11	—	Reserved

Table 10-18. LTEIR Field Descriptions (continued)

Bits	Name	Description
12	CSI	Chip select error interrupt enable. 0 Chip select error reporting is disabled. 1 Chip select error reporting is enabled.
13–31	—	Reserved

10.3.1.13 Transfer Error Attributes Register (LTEATR)

LTEATR is shown in [Figure 10-16](#). After LTEATR[V] has been set, software must clear this bit to allow LTEATR and LTEAR to update following any subsequent errors.

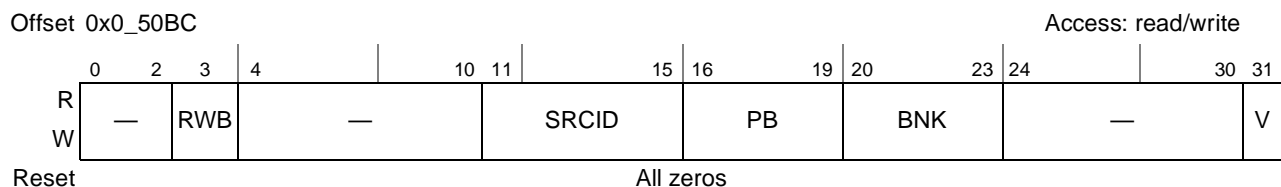


Figure 10-16. Transfer Error Attributes Register (LTEATR)

[Table 10-19](#) describes LTEATR fields.

Table 10-19. LTEATR Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3	RWB	Transaction type for the error: 0 The transaction for the error was a write transaction. 1 The transaction for the error was a read transaction.
4–10	—	Reserved
11–15	SRCID	Captures the source of the transaction when this information is provided on the internal interface to the LBC. The coding of the source ID debug information is the same as the coding of AEATR[MSTR_ID] (see Section 6.2.6, “Arbiter Event Attributes Register (AEATR).”)
16–19	PB	Parity error on byte. There are four parity error status bits, one per byte lane. A bit is set for the byte that had a parity error (bit 16 represents byte 0, the most significant byte lane).
20–23	BNK	Memory controller bank. There is one error status bit per memory controller bank (bit 20 represents bank 0). A bit is set for the local bus memory controller bank that had an error. Note that BNK is invalid if the error was not caused by parity checks.
24–30	—	Reserved
31	V	Error attribute capture is valid. Indicates that the captured error information is valid. 0 Captured error attributes and address are not valid. 1 Captured error attributes and address are valid.

Table 10-21. LBCR Field Descriptions (continued)

Bits	Name	Description
15	EPAR	Determines odd or even parity. Writing the memory with EPAR = 1 and reading the memory with EPAR = 0 generates parity errors for testing. 0 Odd parity 1 Even parity
16–23	BMT	Bus monitor timing. Defines the bus monitor time-out period. Clearing BMT (reset value) selects the maximum count of 2048 bus clock cycles. For non-zero values of BMT, the number of LCLK clock cycles to count down before a time-out error is generated is given by bus cycles = BMT x 8. Apart from BMT = 0x00, the minimum value of BMT is 5, corresponding with 40 bus cycles. Shorter time-outs may result in spurious errors during SDRAM operation. See Section 10.4.1.6, “Parity Generation and Checking (LDP),”
24–31	—	Reserved

10.3.1.16 Clock Ratio Register (LCRR)

The clock ratio register, shown in [Figure 10-19](#), sets the system clock to LBC bus frequency ratio. It also provides configuration bits for extra delay cycles for address and control signals.

NOTE

For proper operation of the system, this register setting must not be altered while local bus memories or devices are being accessed. Special care needs to be taken when running instructions from an local bus controller memory.

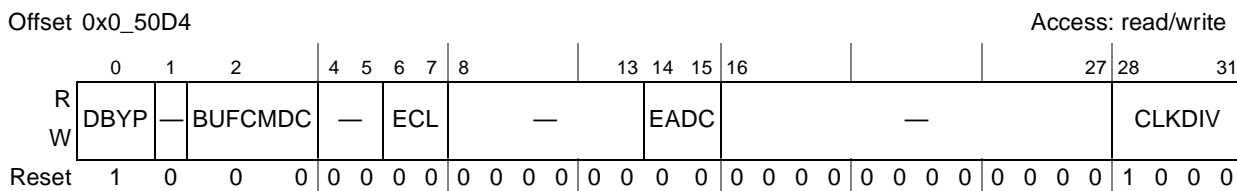


Figure 10-19. Clock Ratio Register (LCRR)

[Table 10-22](#) describes LCRR fields.

Table 10-22. LCRR Field Descriptions

Bits	Name	Description
0	DBYP	DLL bypass. Should be set when using low bus clock frequencies if the DLL is unable to lock. When in DLL bypass mode, incoming data is captured in the middle of the bus clock cycle. 0 The DLL is enabled. 1 The DLL is bypassed (default).
1	—	Reserved
2–3	BUFCMDC	Additional delay cycles for SDRAM control signals. Defines the number of cycles to be added for each SDRAM command when LSDMR[BUFCMD] = 1. 00 4 01 1 10 2 11 3

Table 10-22. LCRR Field Descriptions (continued)

Bits	Name	Description
4–5	—	Reserved
6–7	ECL	Extended CAS latency. Determines the extended CAS latency for SDRAM accesses when LSDMR[CL] = 00. 00 4 01 5 10 6 11 7
8–13	—	Reserved
14–15	EADC	External address delay cycles. Defines the number of cycles for the assertion of LALE. 00 4 01 1 10 2 11 3
16–27	—	Reserved
28–31	CLKDIV	System (input) clock divider. Sets the frequency ratio between the (input) system clock and the memory bus clock. Only the values shown in the table below are allowed. 0000–0001 reserved 0010 2 0011 reserved 0100 4 0101–0111 reserved 1000 8 1001–1111 reserved

10.4 Functional Description

The LBC allows the implementation of memory systems with very specific timing requirements.

- The SDRAM machine provides an interface to SDRAMs using bank interleaving and back-to-back page mode to achieve high performance through a multiplexed address/data bus. An internal DLL for bus clock generation ensures improved data set-up margins for board designs.
- The GPCM provides interfacing for simpler, lower-performance memories and memory-mapped devices. It has inherently lower performance because it does not support bursting. For this reason, GPCM-controlled banks are used primarily for boot-loading and access to low-performance memory-mapped peripherals.
- The UPM supports refresh timers, address multiplexing of the external bus and generation of programmable control signals for row address and column address strobes, to allow for a minimal glue logic interface to DRAMs, burstable SRAMs, and almost any other kind of peripheral. The UPM can be used to generate flexible, user-defined timing patterns for control signals that govern a memory device. These patterns define how the external control signals behave during a read, write, burst-read, or burst-write access. Refresh timers are also available to periodically initiate user-defined refresh patterns.

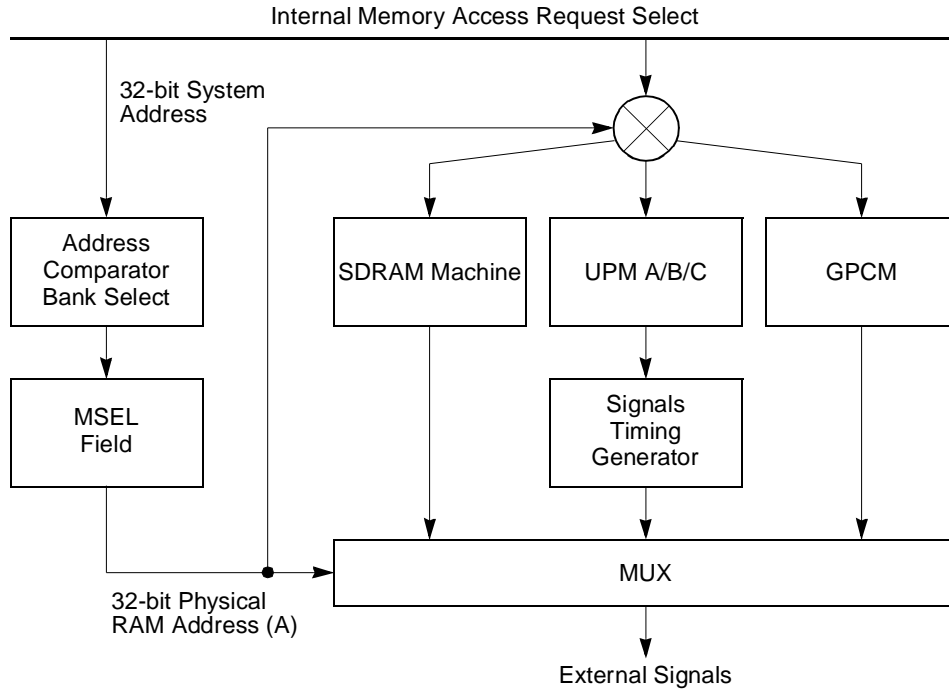


Figure 10-20. Basic Operation of Memory Controllers in the LBC

Each memory bank (chip select) can be assigned to any one of these three type of machines through the machine select bits of the base register for that bank ($BR_n[MSEL]$), as illustrated in Figure 10-20. If a bank match occurs, the corresponding machine (GPCM, SDRAM or UPM) then takes ownership of the external signals that control the access and maintains control until the transaction ends.

10.4.1 Basic Architecture

These subsections describe the basic architecture of the LBC.

10.4.1.1 Address and Address Space Checking

The defined base addresses are written to the BR_n registers; the corresponding address masks are written to the OR_n registers. Each time a local bus access is requested, the internal transaction address is compared with each bank. Addresses are decoded by comparing the 17 msbs of the address, masked by $OR_n[AM]$, with the base address for each bank ($BR_n[BA]$). If a match is found on a memory controller bank, the attributes defined in the BR_n and OR_n for that bank are used to control the memory access. If a match is found in more than one bank, the lowest-numbered bank handles the memory access (that is, bank 0 has priority over bank 1).

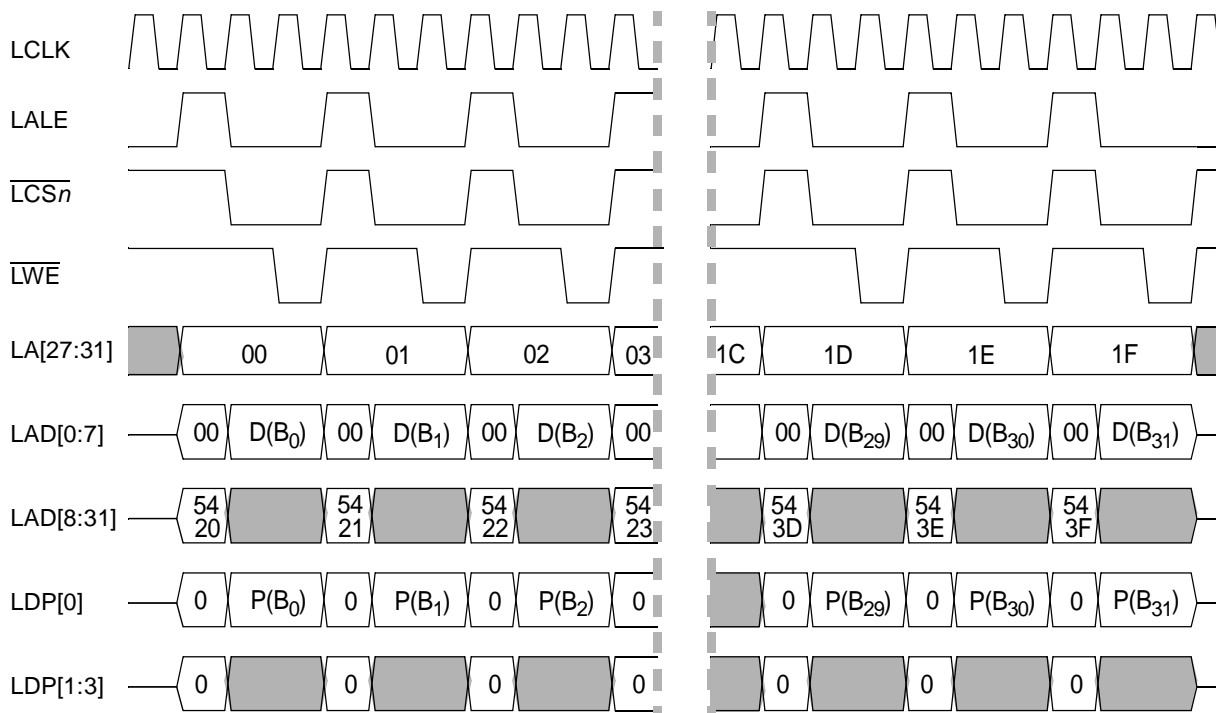
10.4.1.2 External Address Latch Enable Signal (LALE)

The local bus uses a multiplexed address/data bus. When used in a multiplexed mode, the LBC must distinguish between address and data phases, which take place on the same bus (LAD[0:31] signals). The LALE signal, when asserted, signifies an address phase during which the LBC drives the memory address

on the LAD[0:31] signals. An external address latch uses this signal to capture the address and provide it to the address signals of the memory or peripheral device. When LALE is negated, LAD[0:31] then serves as the (bi-directional) data bus for the access. Any address phase initiates the assertion of LALE, which has a programmable duration of between 1 and 4 bus clock cycles.

The frequency of LALE assertion varies across the three memory controllers. For GPCM, every assertion of \overline{LCSn} is considered an independent access, and accordingly, LALE asserts prior to each such access. For example, GPCM driving an 8-bit port would assert LALE and \overline{LCSn} 32 times in order to satisfy a 32-byte cache line transfer. The SDRAM controller asserts LALE only to initiate a burst transfer with a starting address, therefore no more than one assertion of LALE may be required for SDRAM to transfer a 32-byte cache line via a 32-bit port. In the case of UPM, the frequency of LALE assertion depends on how the UPM RAM is programmed. UPM single accesses typically assert LALE once, upon commencement, but it is possible to program UPM to assert LALE several times, and to change the values of LA[27:31] with and without LALE being involved. In general, when using the GPCM and SDRAM controllers it is not necessary to use LA[27:31] if a sufficiently wide latch is used to capture the entire address during LALE phases. UPM may require LA[27:31] if the LBC is generating its own burst address sequence.

To illustrate how a large transaction is handled by the LBC, Figure 10-21 shows LBC signals for GPCM performing a 32-byte write starting at address 0x5420. Note that during each of the 32 assertions of LALE, LA[27:31] exactly mirror LAD[27:31], but during data phases, only LAD[0:7] and LDP0 are driven with valid data and parity, respectively.



Note: All address and signal values are shown in hexadecimal.
 $D(B_k)$ = k^{th} of 32 data bytes, $P(B_k)$ = parity bit of k^{th} data byte.

Figure 10-21. Example of 8-Bit GPCM Writing 32 Bytes to Address 0x5420

10.4.1.3 Data Transfer Acknowledge (TA)

The three memory controllers in the LBC generate an internal transfer acknowledge signal, TA, to allow data on LAD[0:31] to be either sampled (for reads) or changed (on writes). The data sampling/data change always occurs at the end of the bus cycle in which the LBC asserts TA internally. In LBC debug mode, TA is also visible externally on the LDVAL signal. GPCM and SDRAM controllers automatically generate TA according to the timing parameters programmed for them in option and mode registers; a UPM generates TA only when a UPM pattern has the UTA RAM word bit set. Figure 10-22 shows LALE, TA (internal), and $\overline{\text{LCSn}}$. Note that TA and LALE are never asserted together, and that for the duration of LALE, $\overline{\text{LCSn}}$ (or any other control signal) remains negated or frozen.

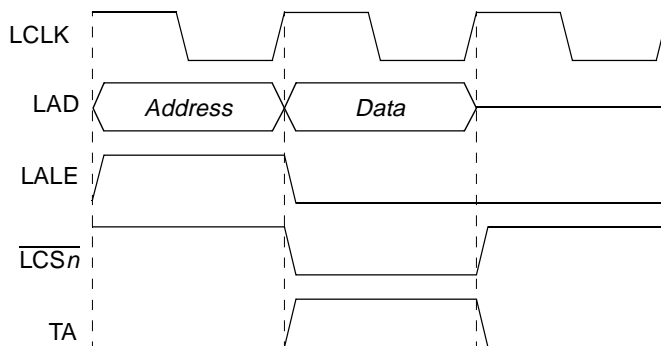


Figure 10-22. Basic LBC Bus Cycle with LALE, TA, and $\overline{\text{LCSn}}$

10.4.1.4 Data Buffer Control (LBCTL)

The memory controller provides a data buffer control signal for the local bus (LBCTL). This signal is activated when a GPCM or UPM controlled bank is accessed. LBCTL can be disabled by setting $\text{ORn}[\text{BCTLd}]$. Access to an SDRAM machine controlled bank does not activate the LBCTL control. LBCTL can be further configured by $\text{LBCR}[\text{BCTLc}]$ to act as an extra $\overline{\text{LWE}}$ or an extra $\overline{\text{LOE}}$ signal when in GPCM mode.

If LBCTL is configured as a data buffer control ($\text{LBCR}[\text{BCTLc}] = 00$), the signal is asserted (high) on the rising edge of the bus clock on the first cycle of the memory controller operation, coincident with LALE. If the access is a write, LBCTL remains high for the whole duration. However, if the access is a read, LBCTL is negated (low) with the negation of LALE so that the memory device is able to drive the bus. If back-to-back read accesses are pending, LBCTL is asserted (high) one bus clock cycle before the next transaction starts (that is, one bus clock cycle before LALE) to allow a whole bus cycle for the bus to turn around before the next address is driven.

If an external bus transceiver is used, LBCTL should be used to signify the write direction when high. Note that the default (reset and bus idle) value of LBCTL is also high.

10.4.1.5 Atomic Operation

The LBC supports the following kinds of atomic bus operations (set by $\text{BRn}[\text{ATOM}]$):

- Read-after-write atomic (RAWA). When a write access hits a memory bank in which $\text{ATOM} = 01$, the LBC reserves the selected memory bank for the exclusive use of the accessing master.

While the bank is reserved, no other device can be granted access to this bank. The reservation is released when the master that created it accesses the same bank with a read transaction. Additional write transactions prior to the releasing read do not change reservation status, but are otherwise processed normally. If the master fails to release the reservation within 256 bus clock cycles, the reservation is released and an atomic error is reported (if enabled). This feature is intended for CAM operations.

- Write-after-read atomic (WARA). When a read access hit a memory bank in which $ATOM = 10$, the LBC reserves the bus for the exclusive use of the accessing master.

During the reservation period, no other device can be granted access to the atomic bank. The reservation is released when the device that created it accesses the same bank with a write transaction. Additional read transactions prior to the releasing write are otherwise processed normally and do not change the reservation status. If the device fails to release the reservation within 256 bus clock cycles, the reservation is released and an atomic error is reported (if enabled).

10.4.1.6 Parity Generation and Checking (LDP)

Parity can be configured for any bank by programming $BRn[DECC]$. Parity is generated and checked on a per-byte basis using $LDP[0:3]$ for the bank if $BRn[DECC] = 01$ (normal parity) or $BRn[DECC] = 10$ for read-modify-write (RMW) parity. Byte lane parity on $LDP[0:3]$ is generated regardless of the $BRn[DECC]$ setting. Note that RMW parity can be used only for 32-bit port size banks. $LBCR[EPAR]$ determines the global type of parity (odd or even).

10.4.1.7 Bus Monitor

A bus monitor is provided to ensure that each bus cycle is terminated within a reasonable (user defined) period. When a transaction starts, the bus monitor starts counting down from the time-out value ($LBCR[BMT]$) until a data beat is acknowledged on the bus. It then reloads the time-out value and resumes the countdown until the data tenure completes and then idles if there is no pending transaction. Setting $LTEDR[BMD]$ disables bus monitor error checking (i.e. the $LTESR[BM]$ bit is not set by a bus monitor time-out); however, the bus monitor is still active and can generate a UPM exception (as noted in [Section 10.4.4.1.4, “Exception Requests,”](#)) or terminate a GPCM access.

It is very important to ensure that the value of $LBCR[BMT]$ is not set too low; otherwise spurious bus time-outs may occur during normal operation—particularly for SDRAMs—resulting in incomplete data transfers. Accordingly, apart from the reset value of 0x00 (corresponding with the maximum time-out of 2048 bus cycles), $LBCR[BMT]$ must not be set below 0x05 (or 40 bus cycles for time-out) under any circumstances.

10.4.2 General-Purpose Chip-Select Machine (GPCM)

The GPCM allows a minimal glue logic and flexible interface to SRAM, EPROM, FEPRM, ROM devices, and external peripherals. The GPCM contains two basic configuration register groups— BRn and ORn .

[Figure 10-23](#) shows a simple connection between an 8-bit port size SRAM device and the LBC in GPCM mode. Byte-write enable signals (LWE) are available for each byte written to memory. Also, the output

enable signal (\overline{LOE}) is provided to minimize external glue logic. On system reset, a global (boot) chip-select is available that provides a boot ROM chip-select ($\overline{LCS0}$) before to the system is fully configured.

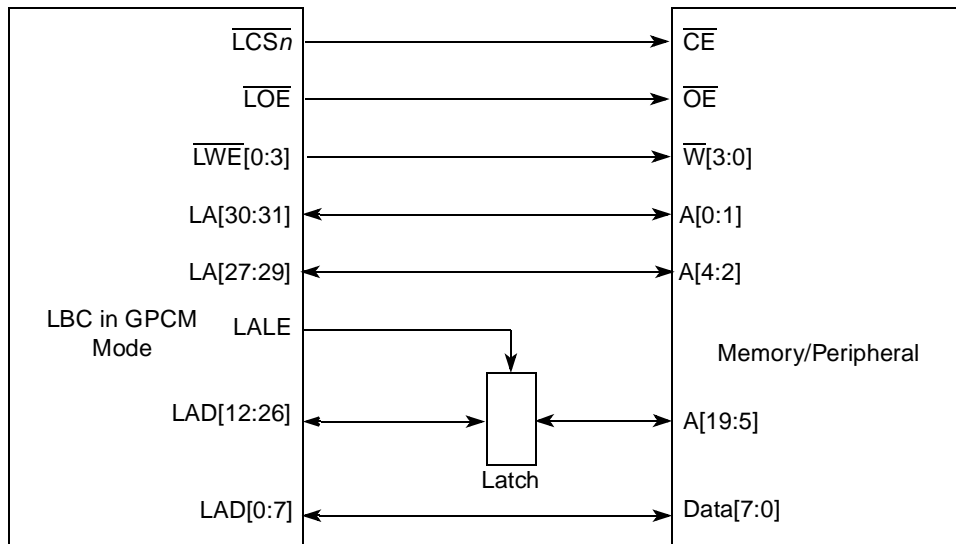


Figure 10-23. Local Bus to GPCM Device Interface

Figure 10-24 shows \overline{LCS} as defined by the setup time required between the address lines and \overline{CE} . The user can configure $ORn[ACS]$ to specify \overline{LCS} to meet this requirement.

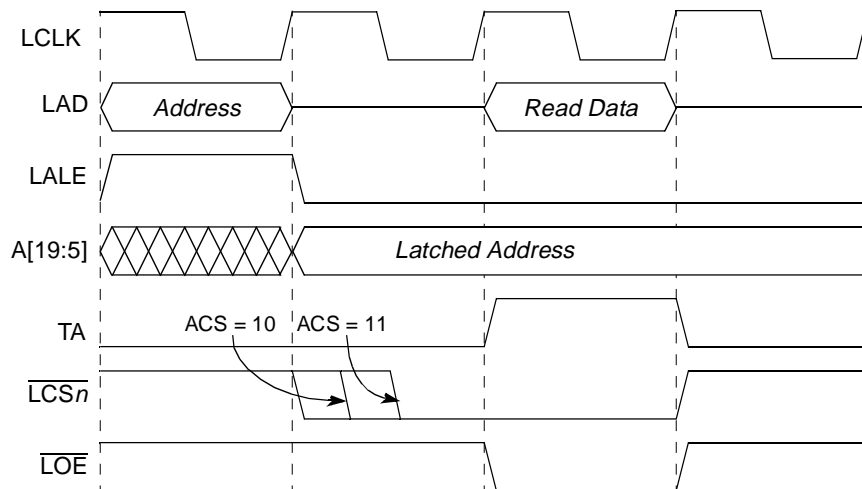


Figure 10-24. GPCM Basic Read Timing (XACS = 0, ACS = 1x, TRLX = 0, CLKDIV = 4,8)

10.4.2.1 Timing Configuration

If $BR_n[MSEL]$ selects the GPCM, the attributes for the memory cycle are taken from OR_n . These attributes include the CSNT, ACS, XACS, SCY, TR LX, EHTR and SETA fields. Table 10-23 shows signal behavior and system response for a write access with $LCRR[CLKDIV] = 4$ or $LCRR[CLKDIV] = 8$. Table 10-24 shows the signal behavior and system response for a read access with $LCRR[CLKDIV] = 4$ or $LCRR[CLKDIV] = 8$. Table 10-25 and Table 10-26 show the write and read signal behavior, respectively, when $LCRR[CLKDIV] = 2$.

Table 10-23. GPCM Write Control Signal Timing for $LCRR[CLKDIV] = 4$ or 8

Option Register Attributes				Signal Behavior (Bus Clock Cycles)			
TR LX	XACS	ACS	CSNT	Address to \overline{LCSn} Asserted	\overline{LCSn} Negated to Address Change	\overline{LWE} Negated to Address/Data Invalid	Total Cycles ¹
0	0	00	0	0	0	0	3+SCY
0	0	10	0	1/4	0	0	3+SCY
0	0	11	0	1/2	0	0	3+SCY
0	1	00	0	0	0	0	3+SCY
0	1	10	0	1	0	0	3+SCY
0	1	11	0	2	0	0	4+SCY
0	0	00	1	0	0	-1/4	3+SCY
0	0	10	1	1/4	-1/4	-1/4	3+SCY
0	0	11	1	1/2	-1/4	-1/4	3+SCY
0	1	00	1	0	0	-1/4	3+SCY
0	1	10	1	1	-1/4	-1/4	3+SCY
0	1	11	1	2	-1/4	-1/4	4+SCY
1	0	00	0	0	0	0	3+2*SCY
1	0	10	0	1+1/4	0	0	4+2*SCY
1	0	11	0	1+1/2	0	0	4+2*SCY
1	1	00	0	0	0	0	3+2*SCY
1	1	10	0	2	0	0	4+2*SCY
1	1	11	0	3	0	0	5+2*SCY
1	0	00	1	0	0	-1-1/4	4+2*SCY
1	0	10	1	1+1/4	-1-1/4	-1-1/4	5+2*SCY
1	0	11	1	1+1/2	-1-1/4	-1-1/4	5+2*SCY
1	1	00	1	0	0	-1-1/4	4+2*SCY
1	1	10	1	2	-1-1/4	-1-1/4	5+2*SCY
1	1	11	1	3	-1-1/4	-1-1/4	6+2*SCY

¹ Total cycles when LALE is asserted for one cycle only ($OR_n[EAD] = 0$; $OR_n[EAD] = 1$ and $LCRR[EADC] = 01$). Asserting LALE for more than one cycle increases the total cycle count accordingly.

Table 10-24. GPCM Read Control Signal Timing for LCRR[CLKDIV] = 4 or 8

Option Register Attributes				Signal Behavior (bus clock cycles)		
TRLX	EHTR	XACS	ACS	Address to $\overline{\text{LCS}}_n$ Asserted	$\overline{\text{LCS}}_n$ Negated to Address Change	Total Cycles ¹
0	0	0	00	0	1	4+SCY
0	0	0	10	1/4	1	4+SCY
0	0	0	11	1/2	1	4+SCY
0	0	1	00	0	1	4+SCY
0	0	1	10	1	1	4+SCY
0	0	1	11	2	1	5+SCY
0	1	0	00	0	2	5+SCY
0	1	0	10	1/4	2	5+SCY
0	1	0	11	1/2	2	5+SCY
0	1	1	00	0	2	5+SCY
0	1	1	10	1	2	5+SCY
0	1	1	11	2	2	6+SCY
1	0	0	00	0	5	8+2*SCY
1	0	0	10	1+1/4	5	9+2*SCY
1	0	0	11	1+1/2	5	9+2*SCY
1	0	1	00	0	5	8+2*SCY
1	0	1	10	2	5	9+2*SCY
1	0	1	11	3	5	10+2*SCY
1	1	0	00	0	9	12+2*SCY
1	1	0	10	1+1/4	9	13+2*SCY
1	1	0	11	1+1/2	9	13+2*SCY
1	1	1	00	0	9	12+2*SCY
1	1	1	10	2	9	13+2*SCY
1	1	1	11	3	9	14+2*SCY

¹ Total cycles when LALE is asserted for one cycle only (OR_n[EAD] = 0; OR_n[EAD] = 1 and LCRR[EADC] = 01). Asserting LALE for more than one cycle increases the total cycle count accordingly.

Table 10-25. GPCM Write Control Signal Timing for LCRR[CLKDIV] = 2

Option Register Attributes				Signal Behavior (Bus Clock Cycles)			
TRLX	XACS	ACS	CSNT	Address to $\overline{\text{LCSn}}$ Asserted	$\overline{\text{LCSn}}$ Negated to Address Change	$\overline{\text{LWE}}$ Negated to Address/Data Invalid	Total Cycles ¹
0	0	00	0	0	0	0	3+SCY
0	0	10	0	1/2	0	0	3+SCY
0	0	11	0	1/2	0	0	3+SCY
0	1	00	0	0	0	0	3+SCY
0	1	10	0	1	0	0	3+SCY
0	1	11	0	2	0	0	4+SCY
0	0	00	1	0	0	0	3+SCY
0	0	10	1	1/2	0	0	3+SCY
0	0	11	1	1/2	0	0	3+SCY
0	1	00	1	0	0	0	3+SCY
0	1	10	1	1	0	0	3+SCY
0	1	11	1	2	0	0	4+SCY
1	0	00	0	0	0	0	3+2*SCY
1	0	10	0	1+1/2	0	0	4+2*SCY
1	0	11	0	1+1/2	0	0	4+2*SCY
1	1	00	0	0	0	0	3+2*SCY
1	1	10	0	2	0	0	4+2*SCY
1	1	11	0	3	0	0	5+2*SCY
1	0	00	1	0	0	-1	4+2*SCY
1	0	10	1	1+1/2	-1	-1	5+2*SCY
1	0	11	1	1+1/2	-1	-1	5+2*SCY
1	1	00	1	0	0	-1	4+2*SCY
1	1	10	1	2	-1	-1	5+2*SCY
1	1	11	1	3	-1	-1	6+2*SCY

¹ Total cycles when LALE is asserted for one cycle only (OR η [EAD]=0; OR η [EAD]=1 and LCRR[EADC]=01). Asserting LALE for more than one cycle increases the total cycle count accordingly.

Table 10-26. GPCM Read Control Signal Timing for LCRR[CLKDIV] = 2

Option Register Attributes				Signal Behavior (Bus Clock cycles)		
TRLX	EHTR	XACS	ACS	Address to $\overline{\text{LCSn}}$ Asserted	$\overline{\text{LCSn}}$ Negated to Address Change	Total Cycles ¹
0	0	0	00	0	1	4+SCY
0	0	0	10	1/2	1	4+SCY
0	0	0	11	1/2	1	4+SCY
0	0	1	00	0	1	4+SCY
0	0	1	10	1	1	4+SCY
0	0	1	11	2	1	5+SCY
0	1	0	00	0	2	5+SCY
0	1	0	10	1/2	2	5+SCY
0	1	0	11	1/2	2	5+SCY
0	1	1	00	0	2	5+SCY
0	1	1	10	1	2	5+SCY
0	1	1	11	2	2	6+SCY
1	0	0	00	0	5	8+2*SCY
1	0	0	10	1+1/2	5	9+2*SCY
1	0	0	11	1+1/2	5	9+2*SCY
1	0	1	00	0	5	8+2*SCY
1	0	1	10	2	5	9+2*SCY
1	0	1	11	3	5	10+2*SCY
1	1	0	00	0	9	12+2*SCY
1	1	0	10	1+1/2	9	13+2*SCY
1	1	0	11	1+1/2	9	13+2*SCY
1	1	1	00	0	9	12+2*SCY
1	1	1	10	2	9	13+2*SCY
1	1	1	11	3	9	14+2*SCY

¹ Total cycles when LALE is asserted for one cycle only (OR_n[EAD]=0; OR_n[EAD]=1 and LCRR[EADC]=01). Asserting LALE for more than one cycle increases the total cycle count accordingly.

10.4.2.2 Chip-Select Assertion Timing

The banks selected to work with the GPCM support an option to drive the $\overline{\text{LCSn}}$ signal with different timings (with respect to the external address/data bus). $\overline{\text{LCSn}}$ can be driven in any of the following ways:

- Simultaneous with the latched memory address. (This refers to the externally latched address and not the address timing on LAD[0:31]. That is, the chip select does not assert during LALE).

- One quarter of a clock cycle later (for LCRR[CLKDIV] = 4, 8).
- One half of a clock cycle later (for LCRR[CLKDIV] = 2, 4, or 8).
- One clock cycle later (for LCRR[CLKDIV] = 4), when OR_n[XACS] = 1.
- Two clock cycles later (for LCRR[CLKDIV] = 2, 4, or 8), when OR_n[XACS] = 1.
- Three clock cycles later (for LCRR[CLKDIV] = 2, 4, or 8), when OR_n[XACS] = 1 and OR_n[TRLX] = 1.

Figure 10-24 shows two chip-select assertion timings for the case LCRR[CLKDIV] = 4 or 8. If LCRR[CLKDIV] = 2, $\overline{\text{LCS}}_n$ asserts identically for OR_n[ACS] = 10 or 11.

10.4.2.2.1 Programmable Wait State Configuration

The GPCM supports internal generation of transfer acknowledge. It allows between 0 and 30 wait states to be added to an access by programming OR_n[SCY] and OR_n[TRLX]. Internal generation of transfer acknowledge is enabled if OR_n[SETA] = 0. If $\overline{\text{LGTA}}$ is asserted externally two bus clock cycles or more before the wait state counter has expired (to allow for synchronization latency), the current memory cycle is terminated by $\overline{\text{LGTA}}$; otherwise it is terminated by the expiration of the wait state counter. Regardless of the setting of OR_n[SETA], wait states prolong the assertion duration of both $\overline{\text{LOE}}$ and $\overline{\text{LWE}}_n$ in the same manner. When TRLX = 1, the number of wait states inserted by the memory controller is doubled from OR_n[SCY] cycles to 2*OR_n[SCY] cycles, allowing a maximum of 30 wait states.

10.4.2.2.2 Chip-Select and Write Enable Negation Timing

Figure 10-23 shows a basic connection between the local bus and a static memory device. In this case, $\overline{\text{LCS}}_n$ is connected directly to $\overline{\text{CE}}$ of the memory device. $\overline{\text{LWE}}_n[0:3]$ are connected to the respective WE[3:0] signals on the memory device where each $\overline{\text{LWE}}_n[0:3]$ signal corresponds to a different data byte.

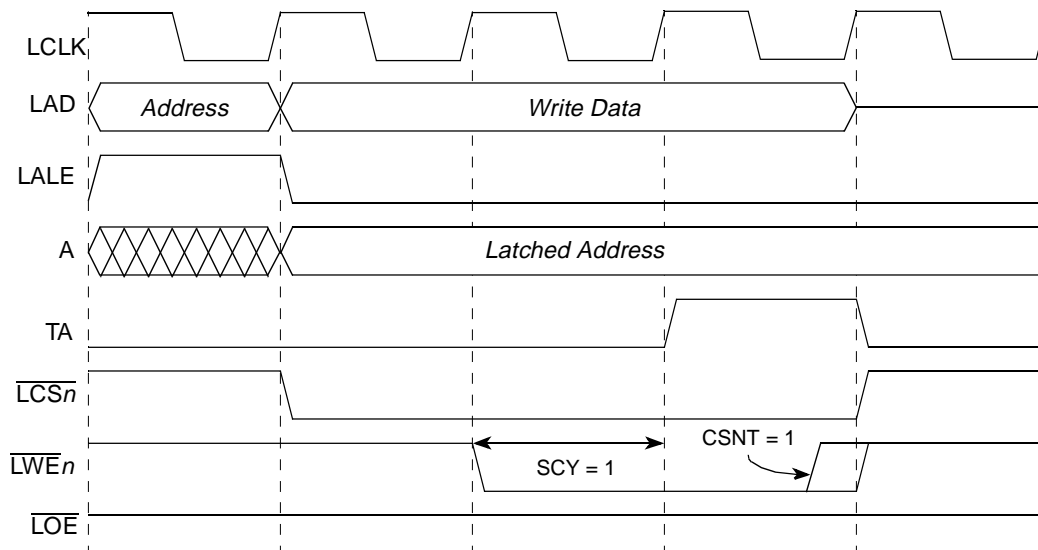


Figure 10-25. GPCM Basic Write Timing (XACS = 0, ACS = 00, CSNT = 1, SCY = 1, TRLX = 0, CLKDIV = 4, 8)

As Figure 10-25 shows, the timing for \overline{LCSn} is the same as for the latched address. The strobes for the transaction are supplied by \overline{LOE} or $\overline{LWE_n}$, depending on the transaction direction—read or write (write case shown in the figure). $OR_n[CSNT]$ controls the timing for the appropriate strobe negation in write cycles. When this attribute is asserted, the strobe is negated one quarter of a clock before the normal case provided that $LCRR[CLDIV] = 4$ or 8 . For example, when $ACS = 00$ and $CSNT = 1$, $\overline{LWE_n}$ is negated one quarter of a clock earlier, as shown in Figure 10-25. If $LCRR[CLDIV] = 2$, $\overline{LWE_n}$ is negated either coincident with \overline{LCSn} or one cycle earlier.

10.4.2.2.3 Relaxed Timing

$OR_n[TRLX]$ is provided for memory systems that require more relaxed timing between signals. Setting $TRLX = 1$ has the following effect on timing:

- An additional bus cycle is added between the address and control signals (but only if ACS is not equal to 00).
- The number of wait states specified by SCY is doubled, providing up to 30 wait states.
- The extended hold time on read accesses ($EHTR$) is extended further.
- \overline{LCSn} signals are negated 1-cycle earlier during writes (if $ACS \neq 00$).
- $\overline{LWE}[0:3]$ signals are negated one cycle earlier during writes.

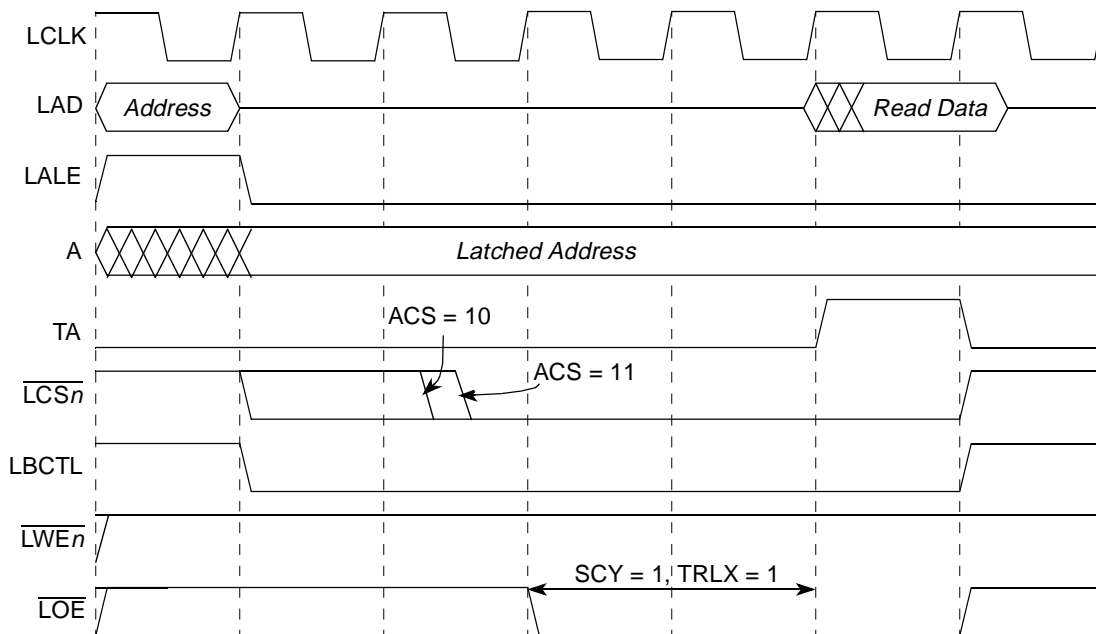


Figure 10-26. GPCM Relaxed Timing Read ($XACS = 0$, $ACS = 1x$, $SCY = 1$, $CSNT = 0$, $TRLX = 1$, $CLKDIV = 4, 8$)

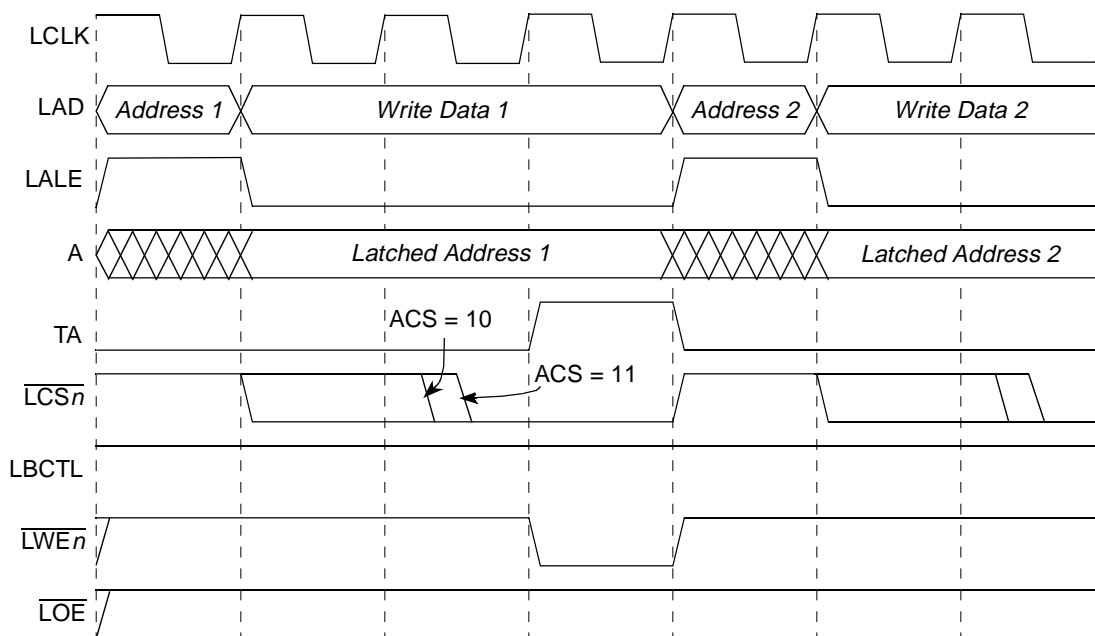


Figure 10-27. GPCM Relaxed Timing Back-to-Back Writes (XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1, CLKDIV = 4, 8)

Figure 10-26 and Figure 10-27 show relaxed timing read and write transactions. The effect of CLKDIV = 2 for these examples is only to delay the assertion of \overline{LCSn} in the ACS = 10 case to the ACS = 11 case. The example in Figure 10-27 also shows address and data multiplexing on LAD[0:31] for a pair of writes issued consecutively.

When TRLX and CSNT are set in a write access, the $\overline{LWE}[0:3]$ strobe signals are negated one clock earlier than in the normal case, as shown in Figure 10-28 and Figure 10-29. If ACS \neq 00, \overline{LCSn} is also negated one clock earlier.

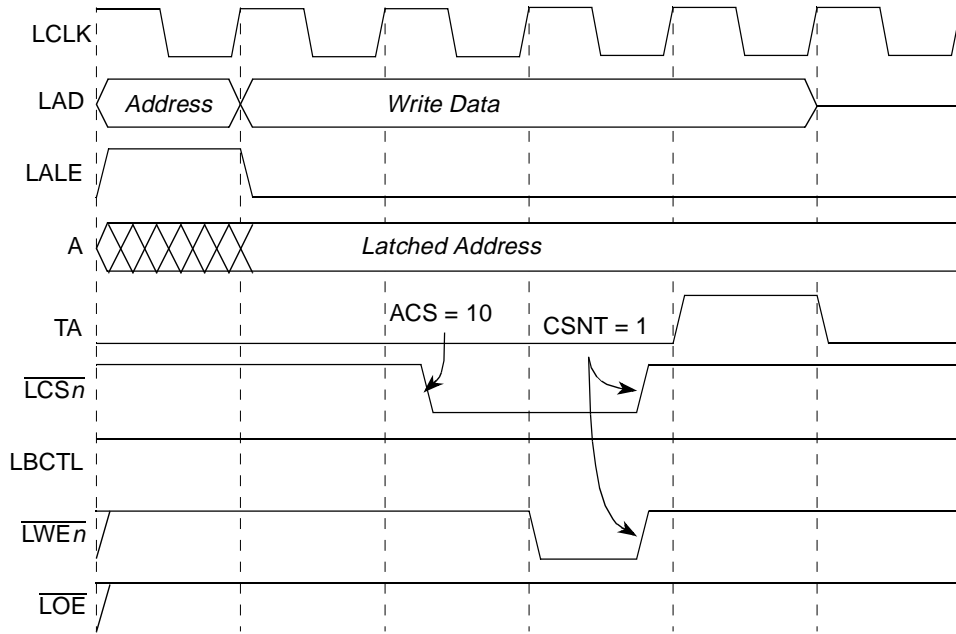


Figure 10-28. GPCM Relaxed Timing Write (XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TR LX = 1, CLKDIV = 4, 8)

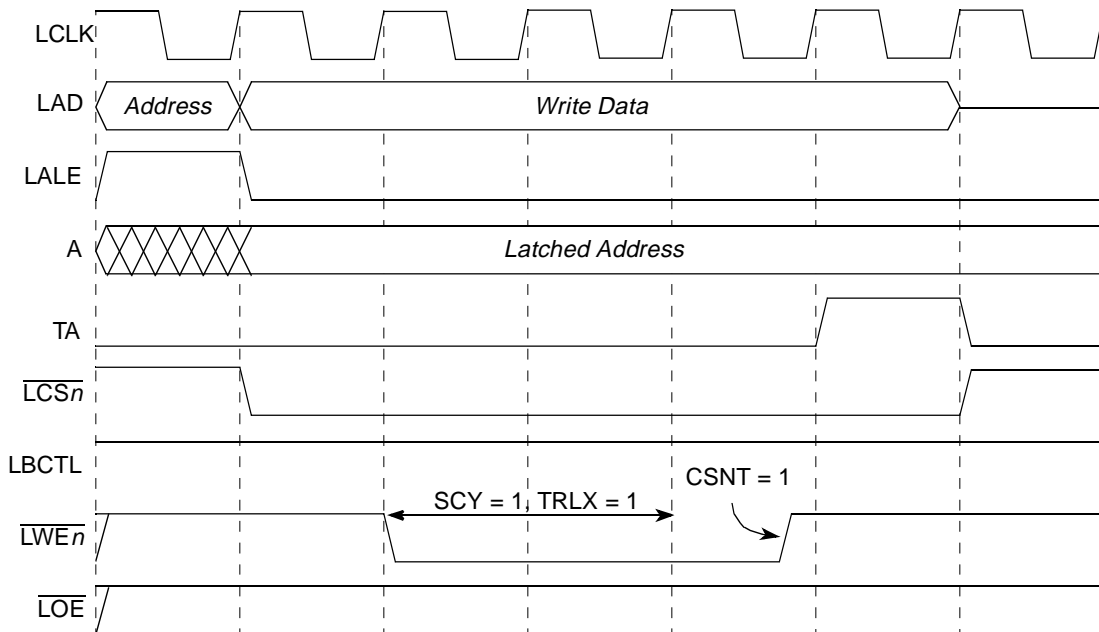


Figure 10-29. GPCM Relaxed Timing Write (XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TR LX = 1, CLKDIV = 4, 8)

10.4.2.2.4 Output Enable (\overline{LOE}) Timing

The timing of the \overline{LOE} is affected only by $TRLX$. It always asserts and negates on the rising edge of the bus clock. \overline{LOE} asserts either on the rising edge of the bus clock after \overline{LCSn} is asserted or coinciding with \overline{LCSn} (if $XACS = 1$ and $ACS = 10$ or $ACS = 11$). Accordingly, assertion of \overline{LOE} can be delayed (along with the assertion of \overline{LCSn}) by programming $TRLX = 1$. \overline{LOE} negates on the rising clock edge coinciding with \overline{LCSn} negation

10.4.2.2.5 Extended Hold Time on Read Accesses

Slow memory devices that take a long time to disable their data bus drivers on read accesses should choose some combination of $ORn[TRLX,EHTR]$. Any access following a read access to the slower memory bank is delayed by the number of clock cycles specified in [Table 10-6](#) in addition to any existing bus turnaround cycle. The final bus turnaround cycle is automatically inserted by the LBC for reads, regardless of the setting of $ORn[EHTR]$.

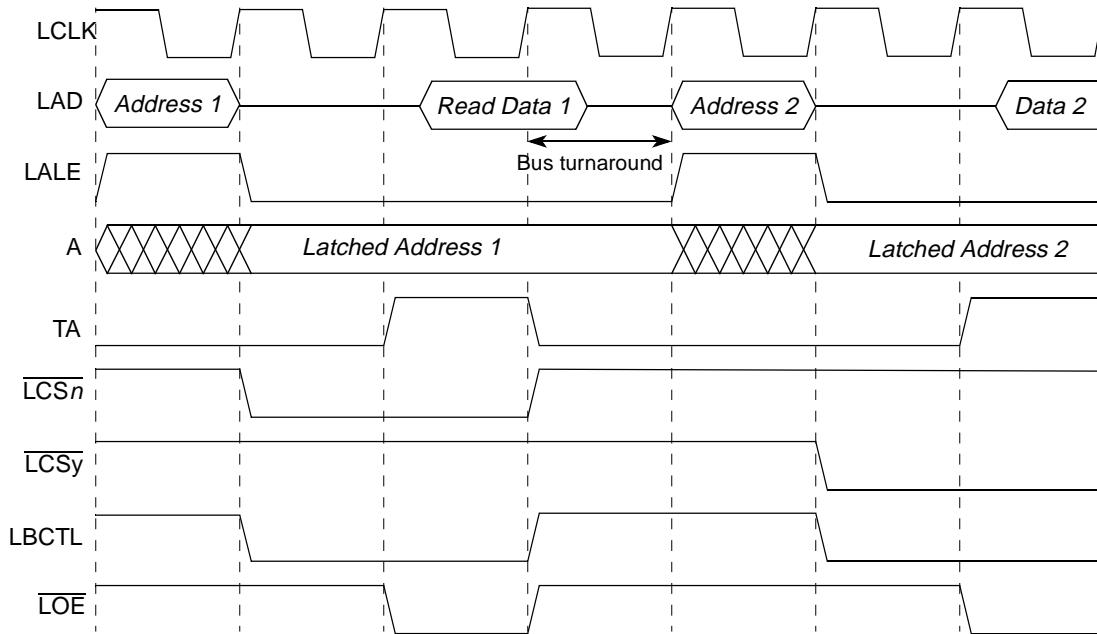


Figure 10-30. GPCM Read Followed by Read ($TRLX = 0$, $EHTR = 0$, Fastest Timing)

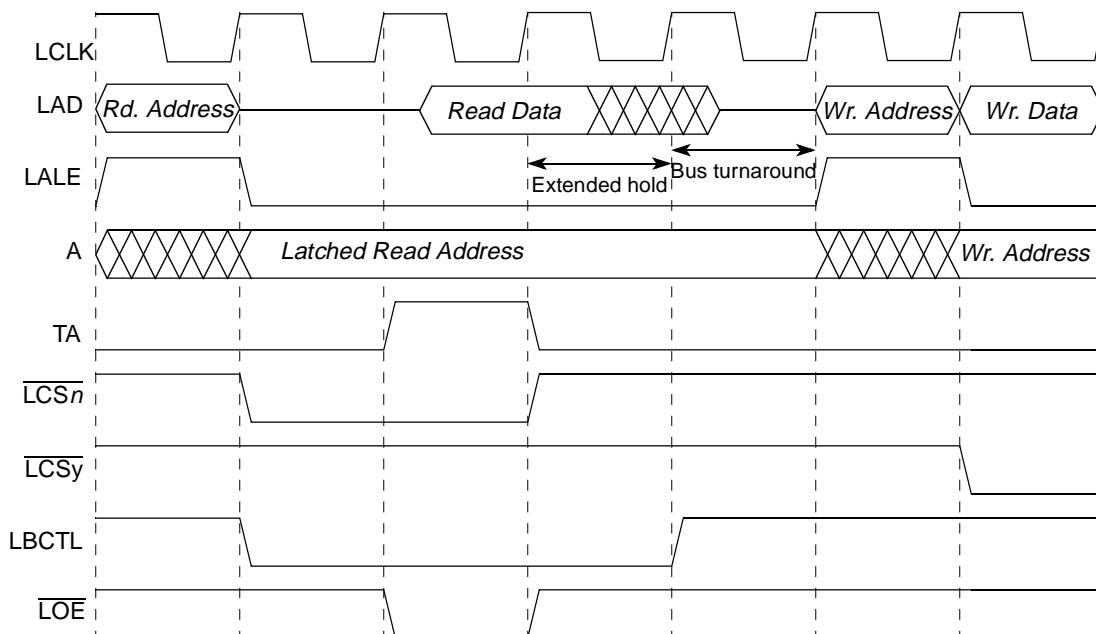


Figure 10-31. GPCM Read Followed by Write (TRLX = 0, EHTR = 1, One-Cycle Extended Hold Time on Reads)

10.4.2.3 External Access Termination ($\overline{\text{LGTA}}$)

External access termination is supported by the GPCM using the asynchronous $\overline{\text{LGTA}}$ input signal, which is synchronized and sampled internally by the local bus. If, during assertion of $\overline{\text{LCSn}}$, the sampled $\overline{\text{LGTA}}$ signal is asserted, it is converted to an internal generation of transfer acknowledge, which terminates the current GPCM access (regardless of the setting of $\text{ORn}[\text{SETA}]$). $\overline{\text{LGTA}}$ should be asserted for at least one bus cycle to be effective. Note that because $\overline{\text{LGTA}}$ is synchronized, bus termination occurs two cycles after $\overline{\text{LGTA}}$ assertion, so in case of read cycle, the device still must drive data as long as $\overline{\text{LOE}}$ is asserted.

The user selects whether transfer acknowledge is generated internally or externally ($\overline{\text{LGTA}}$) by programming $\text{ORn}[\text{SETA}]$. Asserting $\overline{\text{LGTA}}$ always terminates an access, even if $\text{ORn}[\text{SETA}] = 0$.

(internal transfer acknowledge generation), but it is the only means by which an access can be terminated if $OR_n[SETA] = 1$. The timing of \overline{LGTA} is illustrated in Figure 10-32.

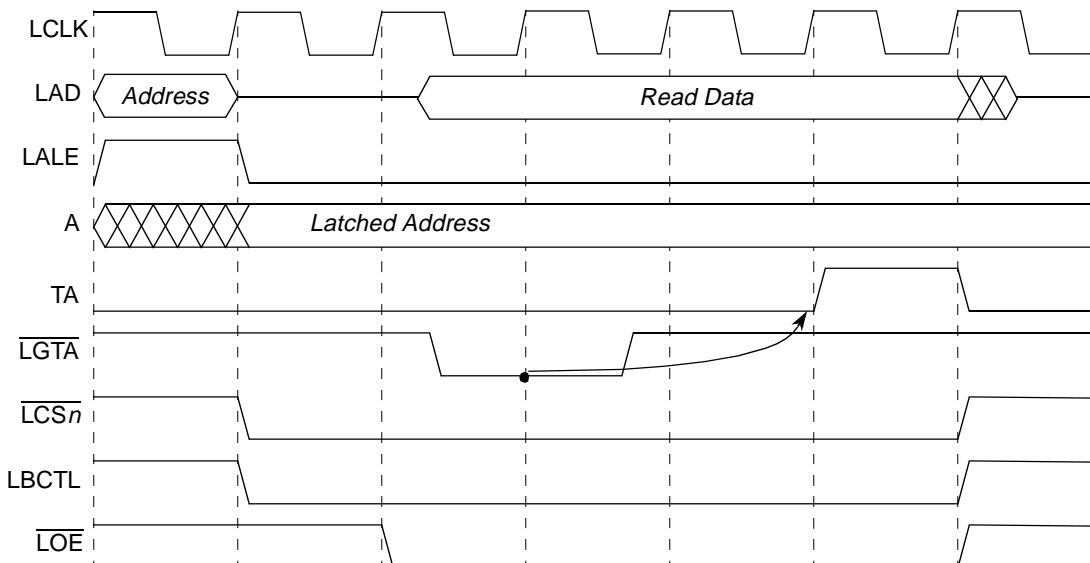


Figure 10-32. External Termination of GPCM Access (PLL Enabled Mode)

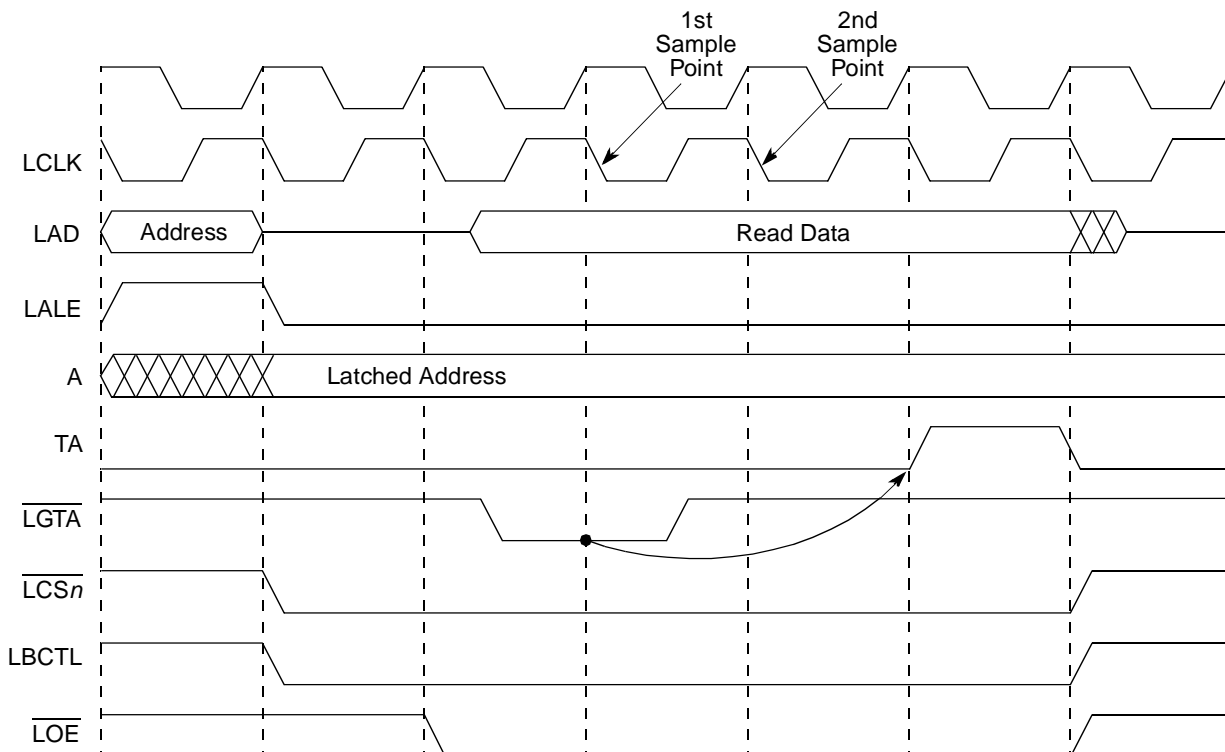


Figure 10-33. External Termination of GPCM Access (PLL Bypass Mode)

10.4.2.4 Boot Chip-Select Operation

Boot chip-select operation allows address decoding for a boot ROM before system initialization. $\overline{\text{LCS0}}$ is the boot chip-select output; its operation differs from other external chip-select outputs after a system reset. When the core begins accessing memory after system reset, $\overline{\text{LCS0}}$ is asserted for every local bus access until BR0 or OR0 is reconfigured.

The boot chip-select also provides a programmable port size, which is configured during reset. The boot chip-select does not provide write protection. $\overline{\text{LCS0}}$ operates this way until the first write to OR0 and it can be used as any other chip-select register after the preferred address range is loaded into BR0. After the first write to OR0, the boot chip-select can be restarted only with a hardware reset. Table 10-27 describes the initial values of the boot bank in the memory controller.

Table 10-27. Boot Bank Field Values After Reset

Register	Field	Setting
BR0	BA	0000_0000_0000_0000_0
	PS	From RCWH[ROMLOC]
	DECC	00
	WP	0
	MSEL	000
	ATOM	00
	V	1
OR0	AM	0000_0000_0000_0000_0
	BCTLD	0
	CSNT	1
	ACS	11
	XACS	1
	SCY	1111
	SETA	0
	TRLX	1
	EHTR	1
	EAD	1

10.4.3 SDRAM Machine

The LBC provides an SDRAM interface (machine) for the local bus. The machine provides the control functions and signals for Intel PC133 and JEDEC-compliant SDRAM devices. Each bank can control an SDRAM device on the local bus.

10.4.3.1 Supported SDRAM Configurations

The memory controller supports any SDRAM configuration with the restrictions that all SDRAM devices that reside on the bus should have the same port size and timing parameters (as defined in LSDMR).

Figure 10-34 shows an example connection between the LBC and a 32-bit SDRAM device with 12 address lines. Note that address signals A[2:0] of the SDRAM connect directly to LA[27:29], address signal A10

connects to the LBCs dedicated LSDA10 signal, while the remaining address bits (except A10) are latched from LAD[20:26].

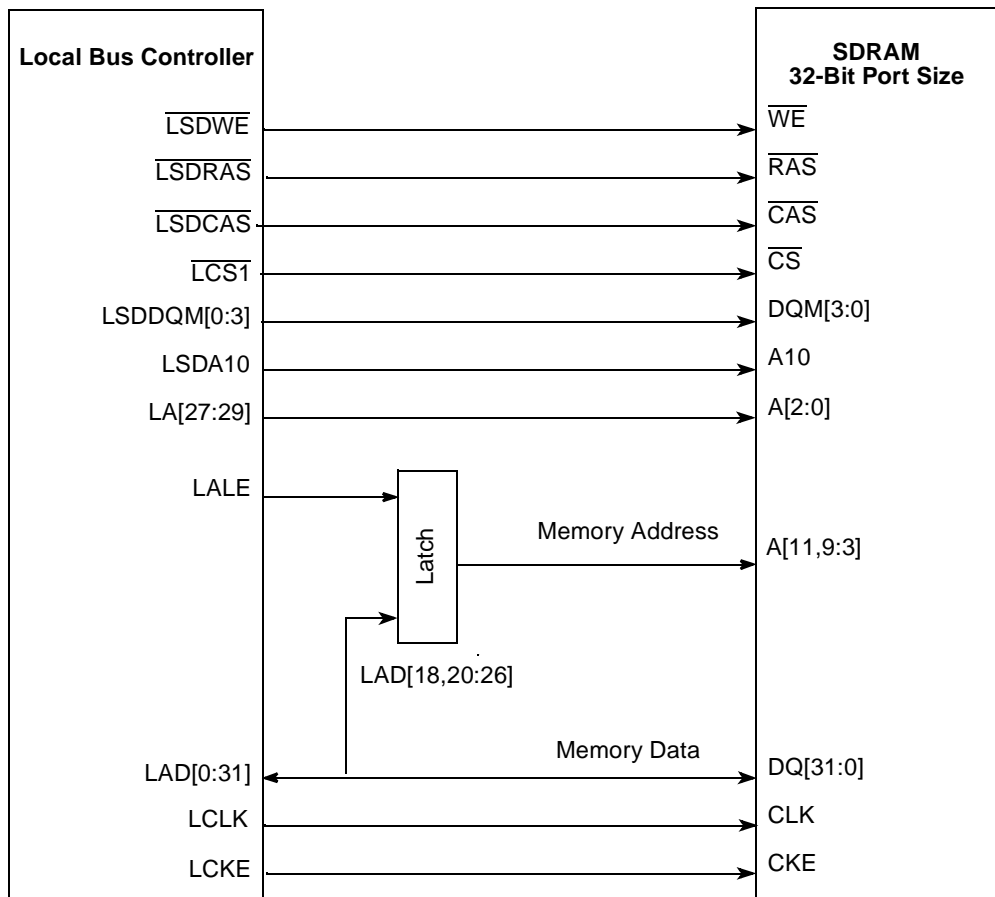


Figure 10-34. Connection to a 32-Bit SDRAM with 12 Address Lines

10.4.3.2 SDRAM Power-On Initialization

Following a system reset, initialization software must set up the programmable parameters in the memory controller banks registers (OR_n , BR_n , LSDMR). After all memory parameters are configured, system software should execute the following initialization sequence for each SDRAM device.

- Issue a PRECHARGE-ALL-BANKS command
- Issue eight AUTO-REFRESH commands
- Issue a MODE-SET command to initialize the mode register

The initial commands are executed by setting LSDMR[OP] and accessing the SDRAM with any write that hits the relevant bank. Since the result of any update to the LSDMR must be in effect before accessing the SDRAM with any write, a write to LSDMR should be followed immediately by a read from LSDMR, which must complete prior to an initial write to SDRAM. Further, the first write to SDRAM should be followed immediately by an SDRAM read, which must complete prior to additional LSDMR updates. This enforces a proper ordering between updates to the LSDMR and write accesses to the SDRAM. If the initialization is being done by the e300, this described protocol is guaranteed only if the SDRAM is

mapped as cache-inhibited and guarded, as the CCSR memory region containing LSDMR should be. If the initialization is from an external host, said host must ensure completion of LSDMR and SDRAM reads prior to subsequent writes, as described above.

Note that software should ensure that no memory operations begin until this process completes.

NOTE

In general (not only during power-on reset) the LSDMR/SDRAM access ordering protocol should be observed for proper operation.

10.4.3.3 Intel PC133 and JEDEC-Standard SDRAM Interface Commands

The SDRAM machine performs all accesses to SDRAM by using Intel PC133 and JEDEC-standard SDRAM interface commands. The SDRAM device samples the command and data inputs on the rising edge of the bus clock. Data at the output of the SDRAM device is sampled on the rising edge of the bus clock.

The following SDRAM interface commands are provided by setting LSDMR[OP] to a non-zero value (LSDMR[OP] = 000 sets normal read/write operation):

Table 10-28. SDRAM Interface Commands

Command (LSDMR[OP])	Description
ACTIVATE (110)	Latches the row address and initiates a memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored with a PRECHARGE command before another ACTIVATE is issued.
MODE-SET (011)	Allows setting of SDRAM options—CAS latency and burst length. CAS latency depends on the SDRAM device used. Although some SDRAMs provide burst lengths of 1, 2, 4, 8, or a page, the local bus memory controller supports only 8-beat bursts for 8-bit and 32-bit port size, or 4-beat bursts for 16-bit port size. The LBC does not support burst lengths of 1, 2 and a page for SDRAMs. The mode register data (CAS latency and burst length) is programmed into the LSDMR register by initialization software after reset. After the LSDMR is set, the LBC transfers the information to the SDRAM device by issuing a MODE-SET command.
PRECHARGE (100: single bank) (101: all-banks)	Restores data from the sense amplifiers to the appropriate row in the SDRAM device array. Also initializes the sense amplifiers to prepare for activating another row in the SDRAM device. Note that the LBC uses LSDA10 to distinguish between PRECHARGE-ALL-BANKS (LSDA10 is high) and PRECHARGE-SINGLE-BANK (LSDA10 is low). The SDRAMs must be compatible with this format.
READ (111)	Latches the column address and transfers data from the selected sense amplifier on the SDRAM device, to the output buffer as determined by the column address. During each successive clock, additional data is driven without additional read commands. At the end of the burst, the page remains open. Burst length is the one set for this bank. Read data is discarded by the LBC.
WRITE (111)	Latches the column address and transfers data from the data signals to the selected sense amplifier on the SDRAM device, as determined by the column address. During each successive clock, additional data is transferred to the sense amplifiers from the data signals without additional write commands. At the end of the burst, the page remains open. Burst length is the one set for this bank. LSDDQM[0:3] are inactive and write data is undefined.

Table 10-28. SDRAM Interface Commands (continued)

Command (LSDMR[OP])	Description
AUTO-REFRESH (001)	Causes a row to be read in all memory banks (JEDEC SDRAM) as determined by the refresh row address counter (similar to CBR). The refresh row address counter is internal to the SDRAM device. After being read, a row is automatically rewritten into the memory array. All banks must be in a precharged state before executing refresh.
SELF-REFRESH (010)	Allows data to be retained in the SDRAM device, even without any active clocks. When placed in this mode, the SDRAM device can issue its own refresh commands, without external clocking from the LBC and the LCKE signal from the LBC is negated. This command can be issued at any time. Normal operation can be resumed only by setting LSDMR[OP] = 000, and waiting a at least of 200 bus cycles before issuing reads or writes to the LBC.

10.4.3.4 Page Hit Checking

The SDRAM machine supports page-mode operation. Each time a page is activated on the SDRAM device, the SDRAM machine stores its address in a page register. The page information, which the user writes to the OR_n register, is used along with the bank size to compare page bits of the address to the page register each time a bus-cycle access is requested. If a match is found, together with a bank match, the bus cycle is defined as a page hit. An open page is automatically closed by the SDRAM machine if the bus becomes idle, unless $OR_n[PMSEL] = 1$.

10.4.3.5 Page Management

The LBC can manage at most four open pages (one page per SDRAM bank) for a single SDRAM device. After a page is opened, it remains open unless one of the following occurs:

- The next access is to a page in a different SDRAM device, in which case all open pages on the current device are closed with a PRECHARGE-ALL-BANKS command.
- The next access is to a page in an SDRAM bank that has a different page open on it, in which case the old page is closed with a PRECHARGE-SINGLE-BANK command.
- The current SDRAM device requires refresh services, in which case all open pages on the current device are closed with a PRECHARGE-ALL-BANKS command.
- The bus becomes idle and $OR_n[PMSEL] = 0$, in which case all open pages in the current device are closed with a PRECHARGE-ALL-BANKS command.

10.4.3.6 SDRAM Address Multiplexing

The lower address bus bits are connected to the memory device's address port with the memory controller multiplexing the row/column and the internal bank select lines. The position of the bank select lines are set according to LSDMR[BSMA]. [Figure 10-35](#) shows how the SDRAM controller shifts the row address down to the lower output address signals during activate and shifts the bank select bits up to the address signals specified by LSDMR[BSMA], supporting page-based interleaving. The lsb of the logical row

address (A_n in Figure 10-35) is aligned with the connected lsb of LAD (bits 29, 30, and 31 for port sizes of 32, 16, and 8 bits, respectively).

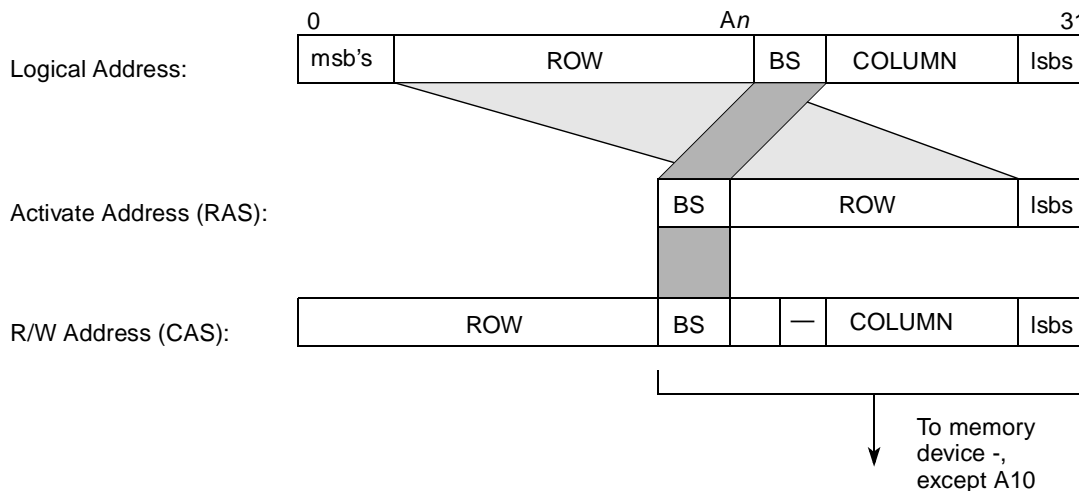


Figure 10-35. SDRAM Address Multiplexing

Note that during normal operation (read/write), a full 32-bit address that includes row and column is generated on LAD[0:31]. However, address/data signal multiplexing implies that the address must be latched by an external latch that is controlled by LALE. All SDRAM device address signals need to be connected to the latched address bits and burst address bits (LA[27:31]) of the LBC, with the exception of A10, which has a dedicated connection on LSDA10. LSDA10 is driven with the appropriate row address bit for SDRAM commands that require A10 to be an address.

10.4.3.7 SDRAM Device-Specific Parameters

The software is responsible for setting correct values for device-specific parameters that can be extracted from the device's data sheet. The values are stored in the OR_n and LSDMR registers. These parameters include the following:

- Precharge to activate interval (LSDMR[PRETOACT])
- Activate to read/write interval (LSDMR[ACTTORW])
- CAS latency, column address to first data out (LSDMR[CL] and LCRR[ECL])
- Write recovery, last data in to precharge (LSDMR[WRC])
- Refresh recovery interval (LSDMR[RFRC])
- External buffers on the control lines present (LSDMR[BUFCMD] and LCRR[BUFCMDC])

In addition, the LBC hardware ensures a default activate to precharge interval of 10 bus cycles. The following sections describe SDRAM parameters programmed in LSDMR.

10.4.3.7.1 Precharge-to-Activate Interval

The precharge-to-activate interval parameter, controlled by LSDMR[PRETOACT], defines the earliest timing for an ACTIVATE or REFRESH command after a PRECHARGE command to the same SDRAM bank.

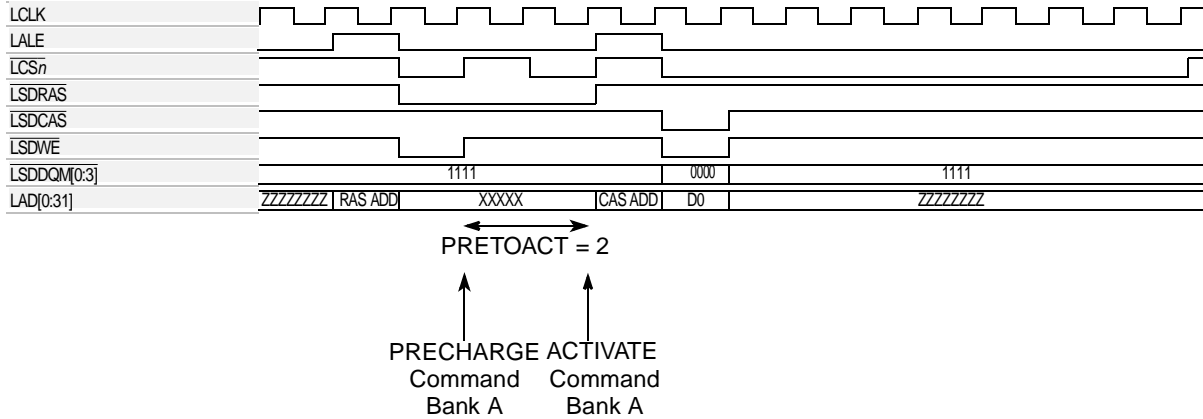


Figure 10-36. PRETOACT = 2 (2 Clock Cycles)

10.4.3.7.2 Activate-to-Read/Write Interval

This active-to-read/write interval parameter, controlled by LSDMR[ACTTORW], defines the earliest timing for a READ/WRITE command after an ACTIVATE command to the same SDRAM bank.

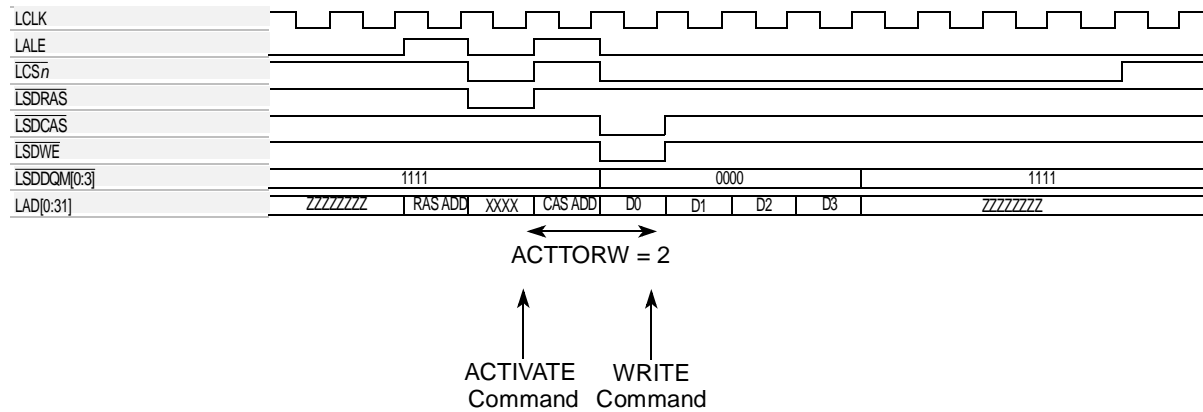


Figure 10-37. ACTTORW = 2 (2 Clock Cycles)

10.4.3.7.3 Column Address to First Data Out—CAS Latency

This parameter, controlled by LSDMR[CL] for latency of 1, 2, or 3 and by LCRR[ECL] for latency of more than 3, defines the timing for first read data after a column address is sampled by the SDRAM.

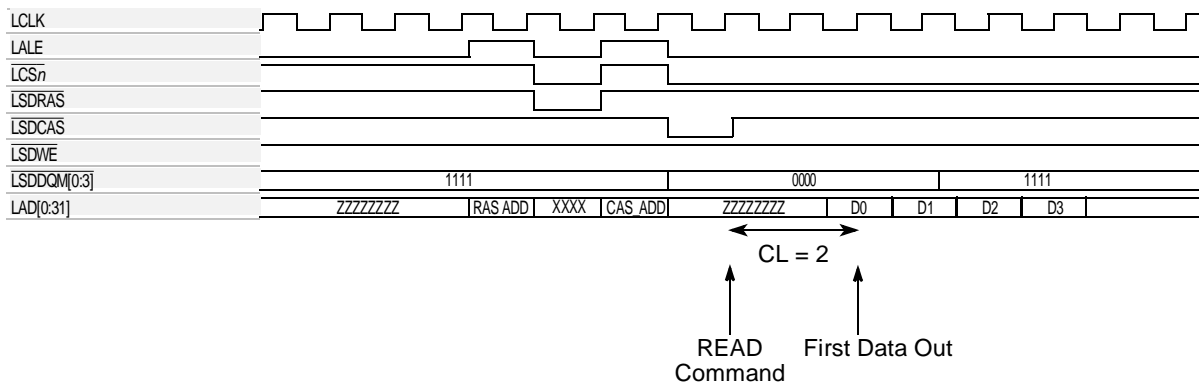


Figure 10-38. CL = 2 (2 Clock Cycles)

10.4.3.7.4 Last Data In to Precharge—Write Recovery

This parameter, controlled by LSDMR[WRC], defines the earliest timing for a PRECHARGE command after the last data was written to the SDRAM.

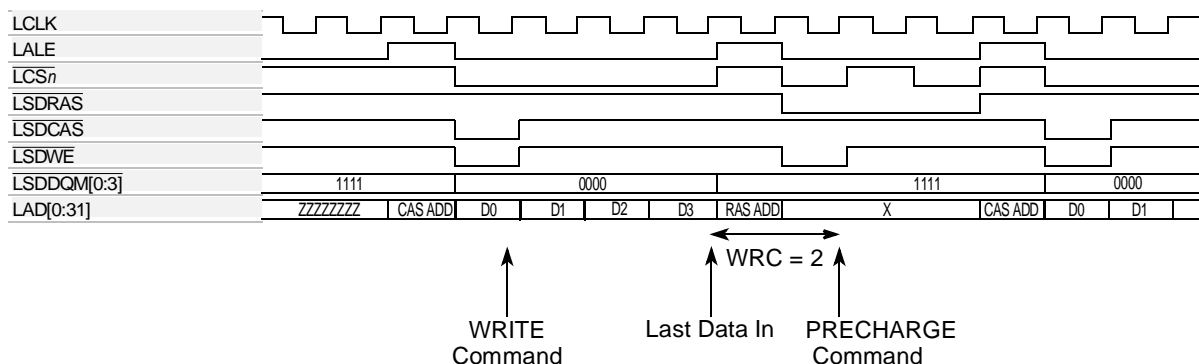


Figure 10-39. WRC = 2 (2 Clock Cycles)

10.4.3.7.5 Refresh Recovery Interval (RFRC)

This parameter, controlled by LSDMR[RFRC], defines the earliest timing for an ACTIVATE or REFRESH command after a REFRESH command to the same SDRAM device.

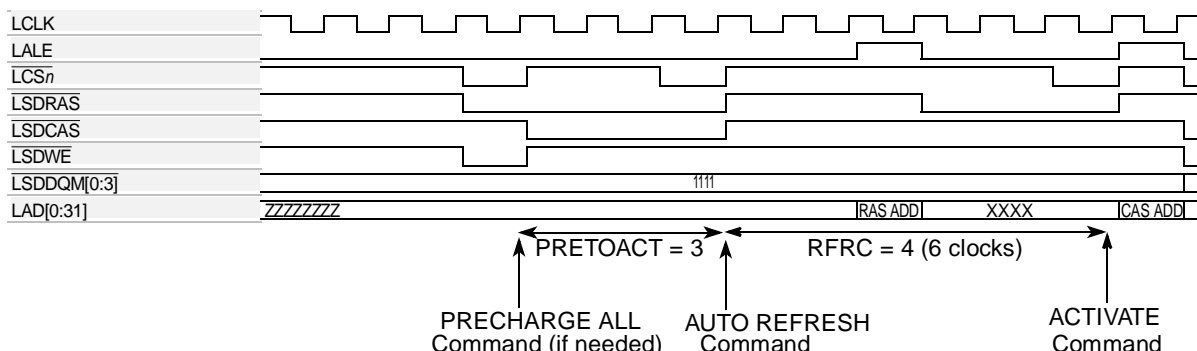


Figure 10-40. RFRC = 4 (6 Clock Cycles)

10.4.3.7.6 External Address and Command Buffers (BUFCMD)

If the additional delay of any buffers placed on the command strobes ($\overline{\text{LSDRAS}}$, $\overline{\text{LSDCAS}}$, $\overline{\text{LSDWE}}$ and $\overline{\text{LSDA10}}$), is endangering the device setup time, LSDMR[BUFCMD] should be set. Setting this bit causes the memory controller to add LCRR[BUFCMDC] extra bus cycles to the assertion of SDRAM control signals ($\overline{\text{LSDRAS}}$, $\overline{\text{LSDCAS}}$, $\overline{\text{LSDWE}}$ and $\overline{\text{LSDA10}}$) for each SDRAM command.

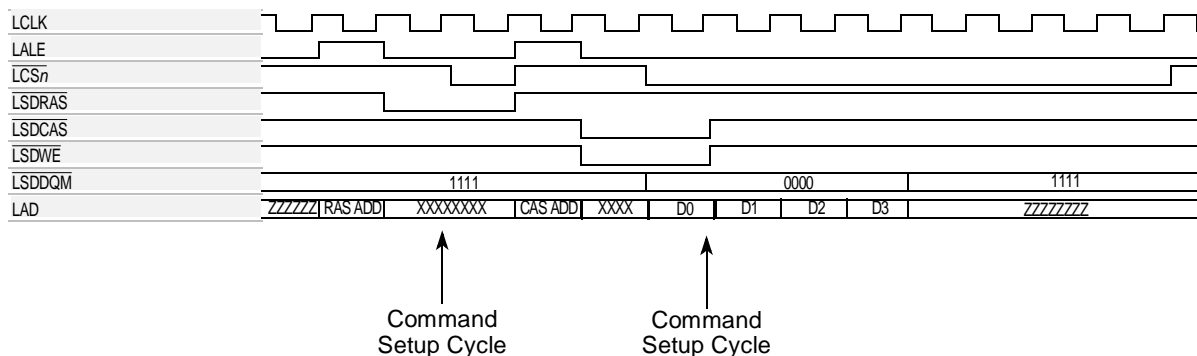


Figure 10-41. BUFCMD = 1, LCRR[BUFCMDC] = 2

10.4.3.8 SDRAM Interface Timing

The following figures show SDRAM timing for various types of accesses.

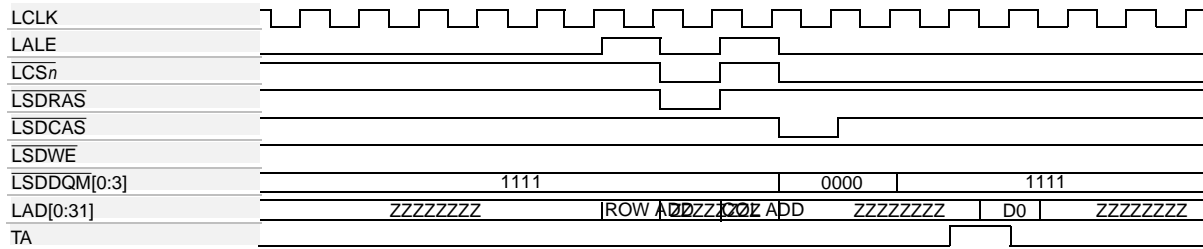


Figure 10-42. SDRAM Single-Beat Read, Page Closed, CL = 3

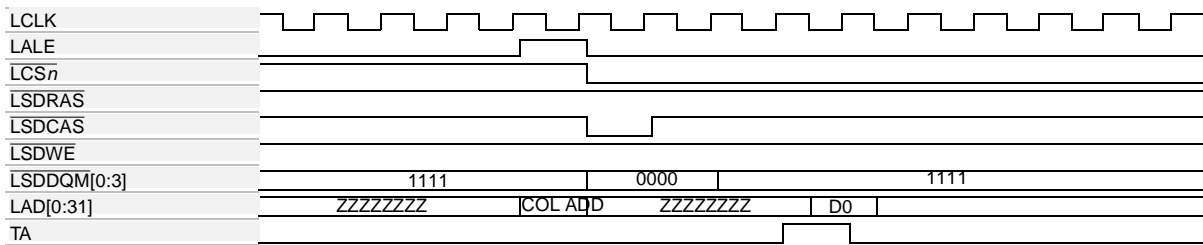


Figure 10-43. SDRAM Single-Beat Read, Page Hit, CL = 3

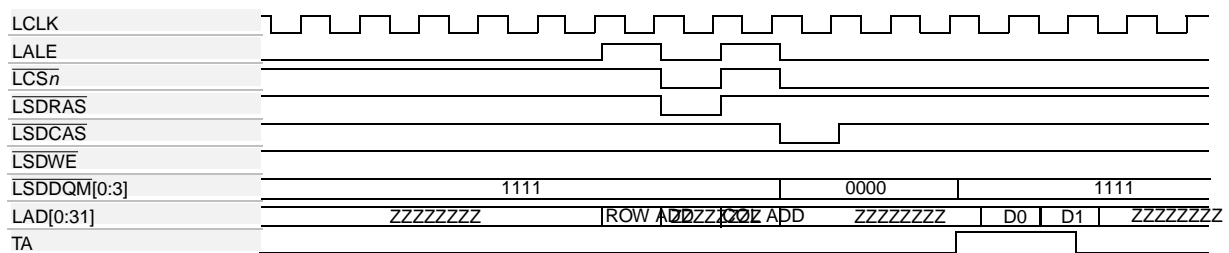


Figure 10-44. SDRAM Two-Beat Burst Read, Page Closed, CL = 3

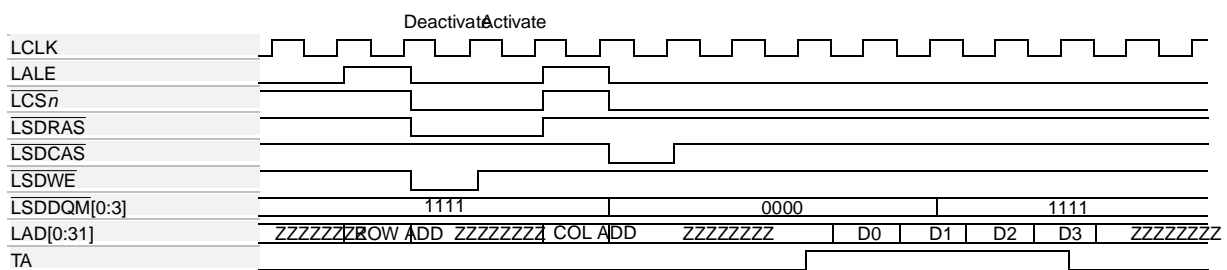


Figure 10-45. SDRAM Four-Beat Burst Read, Page Miss, CL = 3

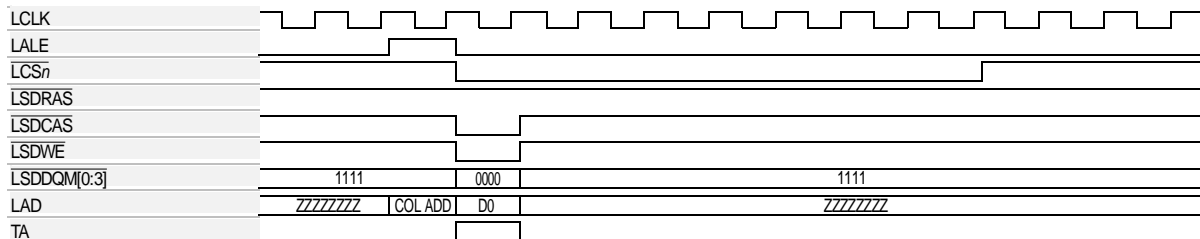


Figure 10-46. SDRAM Single-Beat Write, Page Hit

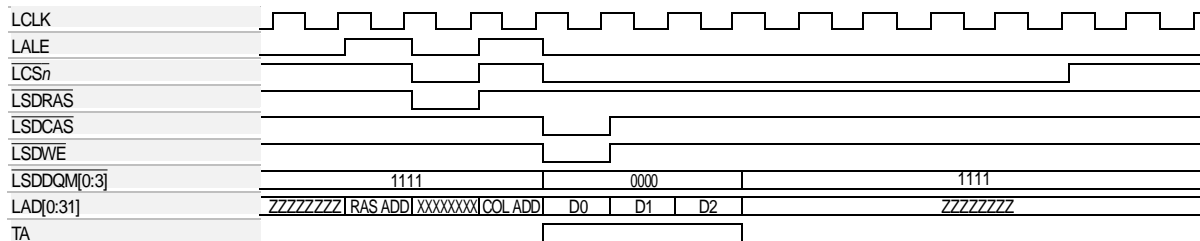


Figure 10-47. SDRAM Three-Beat Write, Page Closed

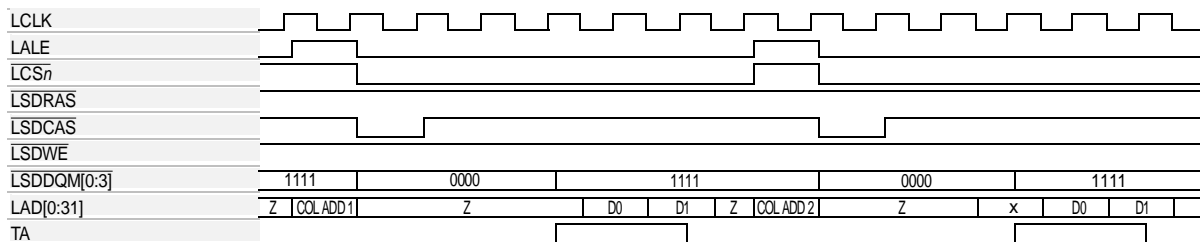


Figure 10-48. SDRAM Read-after-Read Pipelined, Page Hit, CL = 3

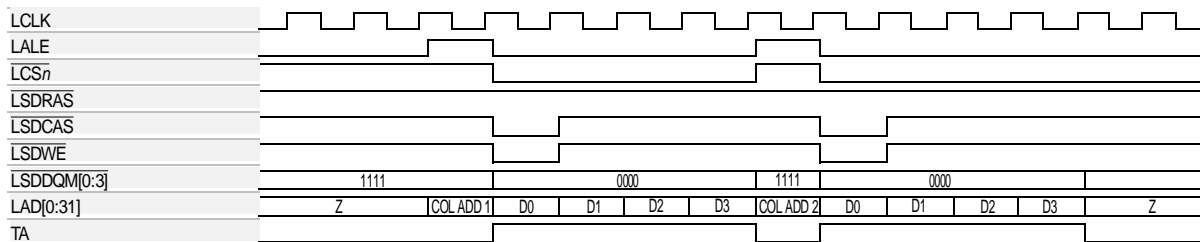


Figure 10-49. SDRAM Write-after-Write Pipelined, Page Hit

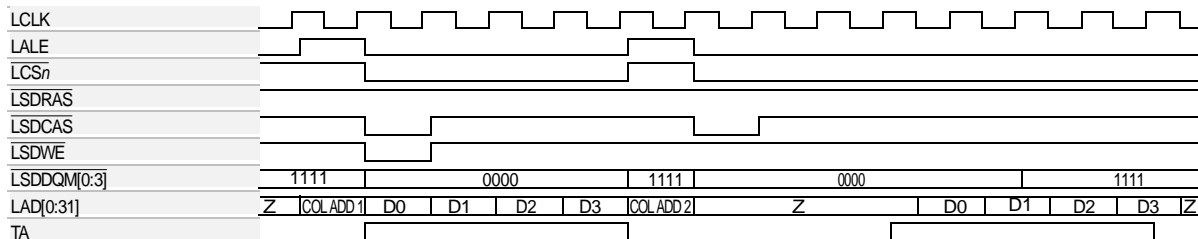


Figure 10-50. SDRAM Read-after-Write Pipelined, Page Hit

10.4.3.9 SDRAM Read/Write Transactions

The SDRAM interface supports read and write transactions of between 1 and 8 data beats for transaction sizes ranging from 1 to 32 bytes. A full burst is performed for each transaction, with the burst length dependent on the port size. A maximum burst of 8 beats is used for an 8- or 32-bit port size, while a maximum burst of 4 beats is used for a 16-bit port size, as programmed in LSDMR[BL]. For reads that require less than the full burst length, extraneous data in the burst is ignored and suppressed by the assertion of LSDDQM[0:3]. For writes that require less than the full burst length, the non-targeted addresses are protected by driving corresponding LSDDQM bits high (inactive) on the irrelevant cycles of the burst. However, system performance is not compromised because, if a new transaction is pending, the SDRAM controller begins executing it immediately, effectively terminating the burst early.

10.4.3.10 SDRAM MODE-SET Command Timing

The LBC transfers mode register data (CAS latency and burst length) stored in the LSDMR register to the SDRAM device by issuing the MODE-SET command, as shown in Figure 10-51. In this case, the latched address carries the mode bits for the command.

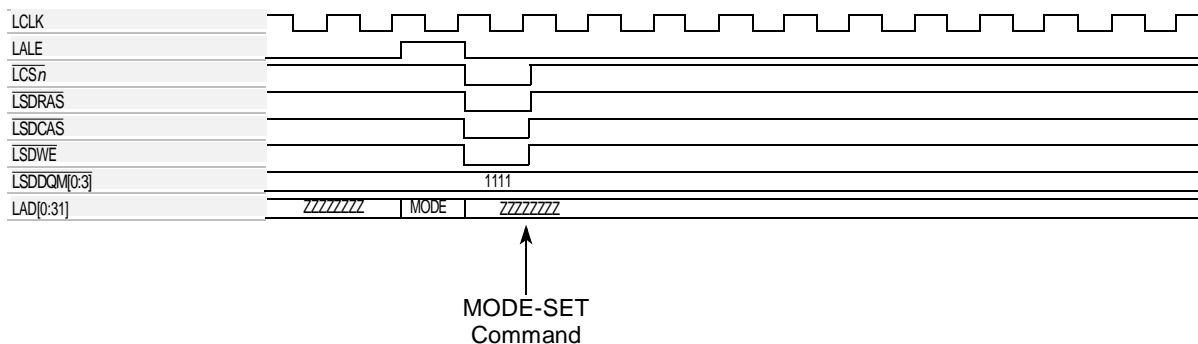


Figure 10-51. SDRAM MODE-SET Command

10.4.3.11 SDRAM Refresh

The memory controller supplies AUTO-REFRESH commands to any connected SDRAM device according to the interval specified in LSRT (and prescaled by MRTPR[PTP]). This interval represents the time period required between refreshes. The values of LSRT and MRTPR depend on the specific SDRAM devices used and the system clock frequency of the LBC. This value should allow for a potential collision

between memory accesses and refresh cycles. The refresh interval period must be greater than the access time. To ensure that read and write operations complete successfully.

There are two levels of refresh request priority—low and high. The low priority request is generated as soon as the refresh timer expires; this request is granted only if no other requests to the memory controller are pending. If the request is not granted (memory controller is busy) and the refresh timer expires two more times, the request becomes high priority and is served when the current memory controller operation finishes.

10.4.3.11.1 SDRAM Refresh Timing

The SDRAM memory controller implements bank staggering for the auto refresh function. This reduces instantaneous current consumption for memory refresh operations.

After a refresh request is granted, the memory controller begins issuing an AUTO-REFRESH command to each device associated with the refresh timer. After a refresh command is issued to an SDRAM device, the memory controller waits for the number of bus clock cycles programmed in the SDRAM machine’s mode register (LSDMR[RFRC]) before issuing any subsequent ACTIVATE command to the same device. To avoid violating SDRAM device timing constraints, the user should ensure that the refresh request interval, defined by LSRT and MRTPR, is greater than the refresh recovery interval, defined by LSDMR[RFRC].

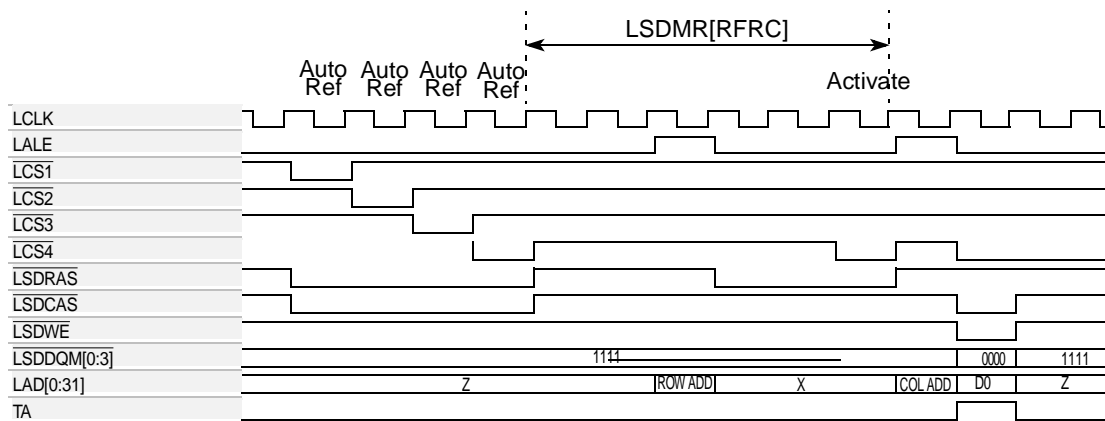


Figure 10-52. SDRAM Bank-Staggered Auto Refresh Timing

10.4.4 User-Programmable Machines (UPMs)

UPMs are flexible interfaces that connect to a wide range of memory devices. At the heart of each UPM is an internal RAM array that specifies the logical value driven on the external memory control signals (\overline{LCS}_n , $\overline{LBS}[0:3]$ and $\overline{LGPL}[0:5]$) for a given clock cycle. Each word in the RAM array provides bits that allow a memory access to be controlled with a resolution of up to one quarter of the external bus clock

period on the byte-select and chip-select lines. A gap of 2 dead LCLK cycles is present on the UPM interface between UPM transactions. Figure 10-53 shows the basic operation of each UPM.

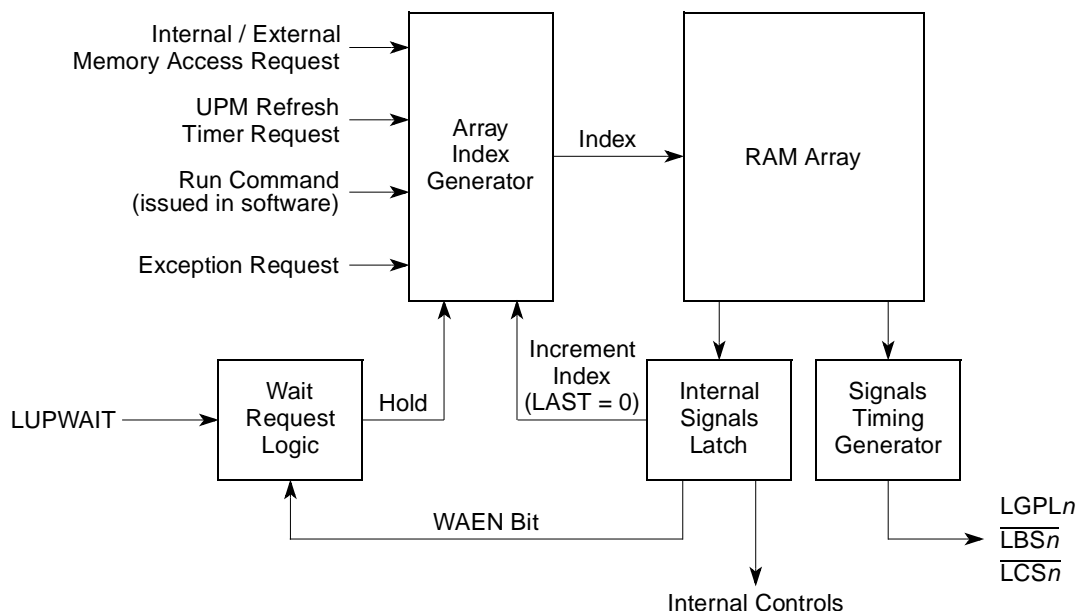


Figure 10-53. User-Programmable Machine Functional Block Diagram

The following events initiate a UPM cycle:

- Any internal device requests an external memory access to an address space mapped to a chip-select serviced by the UPM
- A UPM refresh timer expires and requests a transaction, such as a DRAM refresh
- A bus monitor time-out error during a normal UPM cycle redirects the UPM to execute an exception sequence

The RAM array contains 64 words of 32-bits each. The signal timing generator loads the RAM word from the RAM array to drive the general-purpose lines, byte-selects, and chip-selects. If the UPM reads a RAM word with WAEN set, the external LUPWAIT signal is sampled and synchronized by the memory controller and the current request is frozen.

10.4.4.1 UPM Requests

A special pattern location in the RAM array is associated with each possible UPM request. An internal device's request for a memory access initiates one of the following patterns ($M_nMR[OP] = 00$):

- Read single-beat pattern (RSS)
- Read burst cycle pattern (RBS)
- Write single-beat pattern (WSS)
- Write burst cycle pattern (WBS)

A UPM refresh timer request pattern initiates a refresh timer pattern (RTS).

An exception (caused by a bus monitor time-out error) occurring while another UPM pattern is running initiates an exception condition pattern (EXS).

Figure 10-54 and Table 10-29 show the start addresses of these patterns in the UPM RAM, according to cycle type. RUN commands ($MnMR[OP] = 11$), however, can initiate patterns starting at any of the 64 UPM RAM words.

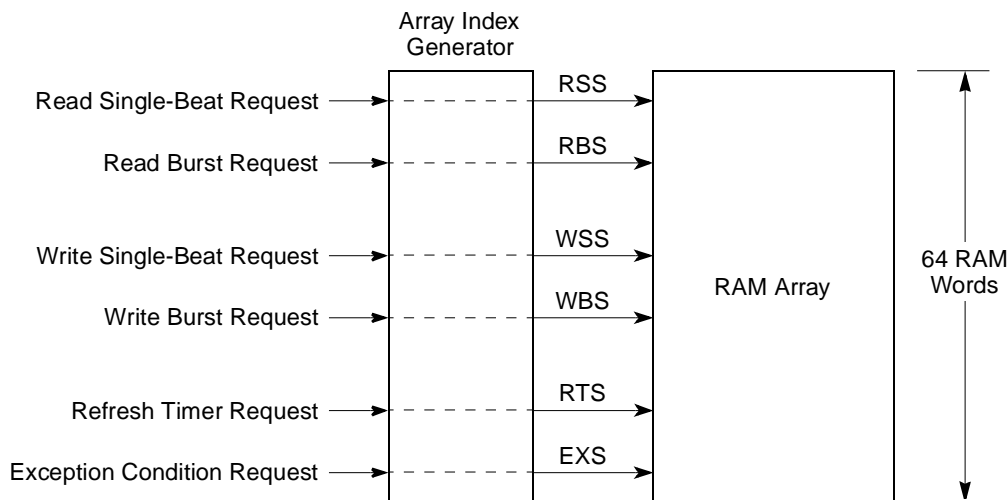


Figure 10-54. RAM Array Indexing

Table 10-29. UPM Routines Start Addresses

UPM Routine	Routine Start Address
Read single-beat (RSS)	0x00
Read burst (RBS)	0x08
Write single-beat (WSS)	0x18
Write burst (WBS)	0x20
Refresh timer (RTS)	0x30
Exception condition (EXS)	0x3C

10.4.4.1.1 Memory Access Requests

The user must ensure that the UPM is appropriately initialized before a request occurs.

The UPM supports two types of memory reads and writes:

- A single-beat transfer transfers one operand consisting of up to a single word (dependent on port size). A single-beat cycle starts with one transfer start and ends with one transfer acknowledge.
- A burst transfer transfers exactly four double words regardless of port size. For 32-bit accesses, the burst cycle starts with one transfer start but ends after eight transfer acknowledges, whereas an 8-bit device requires 32 transfer acknowledges.

The user must ensure that patterns for single-beat transfers contain one, and only one, transfer acknowledge (UTA bit in RAM word set high) and for a burst transfer, contain the exact number of transfer acknowledges required.

Any transfers that do not naturally fit single or burst transfers are synthesized as a series of single transfers. These accesses are treated by the UPM as back-to-back, single-beat transfers. Burst transfers can also be inhibited by setting $OR_n[BI]$. Burst performance can be achieved by ensuring that UPM transactions are 32-byte aligned with a transaction size being some multiple of 32-bytes, which is a natural fit for cache-line transfers, for example.

10.4.4.1.2 UPM Refresh Timer Requests

Each UPM contains a refresh timer that can be programmed to generate refresh service requests of a particular pattern in the RAM array. Figure 10-55 shows the clock division hardware associated with memory refresh timer request generation. The UPM refresh timer register (LURT) defines the period for the timers associated with all three UPMs.

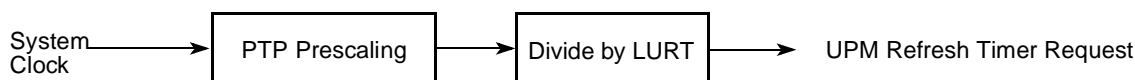


Figure 10-55. Memory Refresh Timer Request Block Diagram

By default, all local bus refreshes are performed using the refresh pattern of UPMA. This means that if refresh is required, $MAMR[RFEN]$ must be set. It also means that only one refresh routine should be programmed and be placed in UPMA, which serves as the refresh executor. Any banks assigned to a UPM are provided with the refresh pattern if the $RFEN$ bit of the corresponding UPM is set. UPMA assigned banks, therefore, always receive refresh services when $MAMR[RFEN]$ is set, while UPMB and UPMC assigned banks also receive (the same) refresh services if the corresponding $M_nMR[RFEN]$ bits are set.

Note that the UPM refresh timer request should not be used in a system with SDRAM refresh enabled. The system designer must choose to use either SDRAM refresh or UPM refresh. Using both may result in missing refresh periods to memory.

10.4.4.1.3 Software Requests—RUN Command

Software can start a request to the UPM by issuing a RUN command to the UPM. Some memory devices have their own signal handshaking protocol to put them into special modes, such as self-refresh mode. Other memory devices require special commands to be issued on their control signals, such as for SDRAM initialization.

For these special cycles, the user creates a special RAM pattern that can be stored in any unused areas in the UPM RAM. Then a RUN command is used to run the cycle. The UPM runs the pattern beginning at the specified RAM location until it encounters a RAM word with its $LAST$ bit set. The RUN command is issued by setting $M_nMR[OP] = 11$ and accessing UPM n memory region with any write transaction that hits the corresponding UPM machine. $M_nMR[MAD]$ determines the starting address in the RAM array for the pattern.

Note that transfer acknowledges (UTA bit in the RAM word) are ignored for software (RUN command) requests, and hence the LAD signals remain high-impedance unless the normal initial LALE occurs or the RUN pattern causes assertion of LALE to occur on changes to the RAM word AMX field.

10.4.4.1.4 Exception Requests

When the LBC under UPM control initiates an access to a memory device and an exception occurs (bus monitor time-out), the UPM provides a mechanism by which memory control signals can meet the device's timing requirements without losing data. The mechanism is the exception pattern that defines how the UPM negates its signals in a controlled manner.

10.4.4.2 Programming the UPMs

The UPM is a micro sequencer that requires microinstructions or RAM words to generate signal timings for different memory cycles. Follow these steps to program the UPMs:

1. Set up BR_n and OR_n registers.
2. Write patterns into the RAM array.
3. Program MRTPR, LURT, and MAMR[RFEN] if refresh is required.
4. Program M_nMR .

Patterns are written to the RAM array by setting $M_nMR[OP] = 01$ and accessing the UPM with any write transaction that hits the relevant chip select. The entire array is thus programmed by an alternating series of writes: to MDR (RAM word to be written) each time followed by a read from MDR and then followed by a (dummy) write transaction to the relevant UPM assigned bank. A read from MDR is required to ensure that the MDR update has occurred prior to the (dummy) write transaction.

RAM array contents may also be read for debug purposes, for example, by alternating dummy read transactions, each time followed by reads of MDR (when $M_nMR[OP] = 10$).

M_nMR / MDR registers should not be updated while dummy read/write access is still in progress. If the $M_nMR[MAD]$ is incremented, the previous dummy transaction is already completed. To enforce proper ordering between updates to the M_nMR /MDR register and the dummy accesses to the UPM memory region, two rules must be followed:

- Because the result of any update to the M_nMR /MDR register must be in effect before the dummy read or write to the UPM region, a write to M_nMR /MDR should be followed immediately by a read of M_nMR /MDR.
- The UPM memory region should have the same MMU settings as the memory region containing the M_nMR configuration register; both should be mapped by the MMU as Cache-Inhibited and Guarded. This prevents the e300c1 core from reordering a read of the UPM memory around the read of M_nMR . When the programming of the UPM array is complete, the MMU setting for the associated address range can be set to the proper mode for normal operation, such as cacheable and copyback.

For proper signalling, the following guidelines must be followed while programming UPM RAM words:

- For UPM reads, program UTA and LAST in the same or consecutive RAM words.
- For UPM burst reads, program last UTA and LAST in the same or consecutive RAM words.

- For UPM writes, program UTA and LAST in the same RAM word.
- For UPM burst writes, program last UTA and LAST in the same RAM word.

10.4.4.2.1 UPM Programming Example (Two Sequential Writes to the RAM Array)

The following example further illustrates the steps required to perform two writes to the RAM array at non-sequential addresses assuming that the relevant BR_x and OR_x registers have been previously set up:

1. Program M_nMR for the first write (with the desired RAM array address).
2. Write pattern/data to MDR to ensure that the M_nMR has already been updated with the desired configuration.
3. Read MDR to ensure that the MDR has already been updated with the desired pattern. (Or, read M_nMR register if step 2 is not performed.)
4. Perform a dummy write transaction. (Write transaction can now be performed.)
5. Read/check M_nMR[MAD]. If incremented, the previous dummy write transaction is completed; proceed to step 6. Repeat step 5 until incremented.
6. Program M_nMR for the second write with the desired RAM array address.
7. Write pattern/data to MDR to ensure that the M_nMR has already been updated with the desired configuration.
8. Read MDR to ensure that the MDR has already been updated with the desired pattern.
9. Perform a dummy write transaction. (Write transaction can now be performed.)
10. Read/check M_nMR[MAD]. If incremented, the previous dummy write transaction is completed.

Note that if step 1 (or 6) and 2 (or 7) are reversed, step 3 (or 8) is replaced by the following:

- Read M_nMR to ensure that the M_nMR has already been updated with the desired configuration.

10.4.4.2.2 UPM Programming Example (Two Sequential Reads from the RAM Array)

RAM array contents may also be read for debug purposes, for example, by alternating dummy read transactions, each time followed by reads of MDR (M_nMR[OP] = 0b10). The following example further illustrates the steps required to perform two reads from the RAM array at non-sequential addresses assuming that the relevant BR_x and OR_x registers have been previously set up:

1. Program M_nMR for the first read with the desired RAM array address.
2. Read M_nMR to ensure that the M_nMR has already been updated with the desired configuration, such as RAM array address.
3. Perform a dummy read transaction. (Read transaction can now be performed.)
4. Read/check M_nMR[MAD]. If incremented, the previous dummy read transaction is completed; proceed to step 5. Repeat step 4 until incremented.
5. Read MDR.
6. Program M_nMR for the second read with the desired RAM array address.

7. Read M_nMR to ensure that the M_nMR has already been updated with the desired configuration, such as RAM array address.
8. Perform a dummy read transaction. (Read transaction can now be performed.)
9. Read/check $M_nMR[MAD]$. If incremented, the previous dummy read transaction is completed; proceed to step 10. Repeat step 9 until incremented.
10. Read MDR.

10.4.4.3 UPM Signal Timing

RAM word fields specify the value of the various external signals at a granularity of up to four values for each bus clock cycle. The signal timing generator causes external signals to behave according to timing specified in the current RAM word. For $LCRR[CLKDIV] = 4$ or 8 , each bit in the RAM word relating to \overline{LCS}_n and \overline{LBS} timing specifies the value of the corresponding external signal at each quarter phase of the bus clock. If $LCRR[CLKDIV] = 2$, the external signal can change value only on each half phase of the bus clock. If the RAM word in this case ($LCRR[CLKDIV] = 2$) specifies a quarter phase signal change, the signal timing generator interprets this as a half cycle change.

The division of UPM bus cycles into phases is shown in [Figure 10-56](#) and [Figure 10-57](#). If $LCRR[CLKDIV] = 2$, the bus cycle comprises only two active phases, T1 and T3, which correspond with the first and second halves of the bus clock cycle, respectively. However, if $LCRR[CLKDIV] = 4$ or 8 , four phases, T1–T4, define four quarters of the bus clock cycle. Because T2 and T4 are inactive when $LCRR[CLKDIV] = 2$, UPM ignores signal timing programmed for assertion in either of these phases in the case $LCRR[CLKDIV] = 2$.

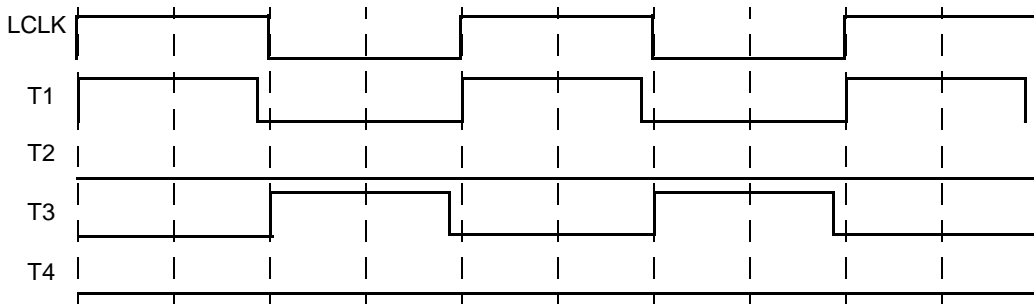


Figure 10-56. UPM Clock Scheme for $LCRR[CLKDIV] = 2$

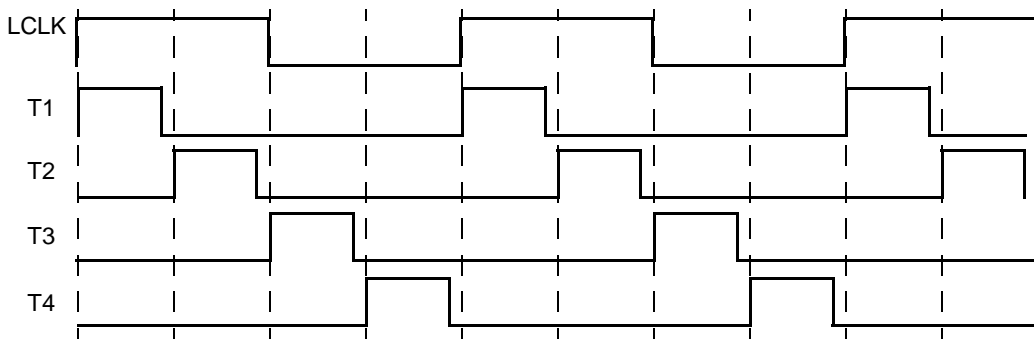


Figure 10-57. UPM Clock Scheme for $LCRR[CLKDIV] = 4$ or 8

10.4.4.4 UPM RAM Array

The RAM array for each UPM is 64 locations deep and 32 bits wide, as shown in Figure 10-58. The signals at the bottom of the figure are UPM outputs. The selected \overline{LCS}_n is for the bank that matches the current address. The selected \overline{LBS} is for the byte lanes read or written by the access.

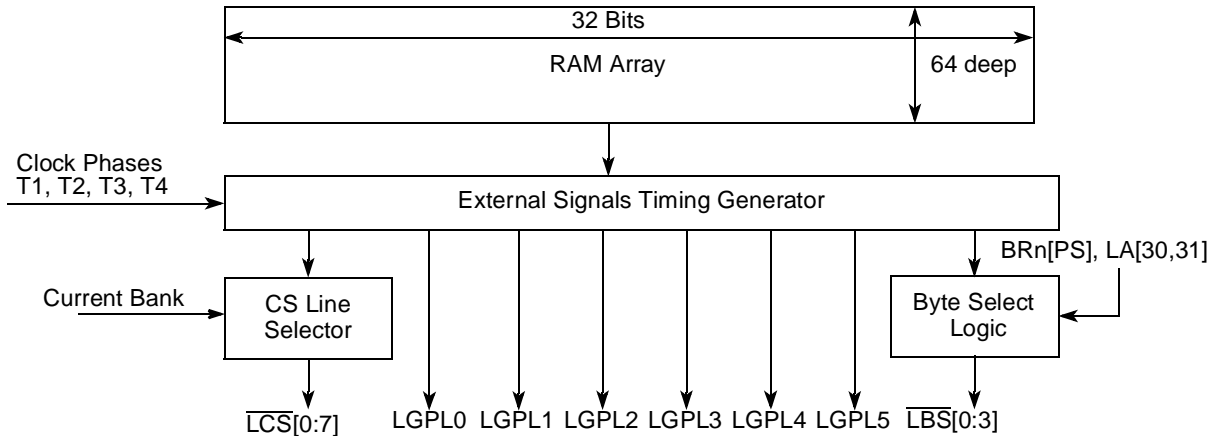


Figure 10-58. UPM RAM Array and Signal Generation

10.4.4.4.1 UPM RAM Words

The RAM word is a 32-bit microinstruction stored in one of 64 locations in the RAM array. It specifies timing for external signals controlled by the UPM. Figure 10-59 shows the RAM word fields. When $LCRR[CLKDIV] = 4$ or 8 , the CST_n and BST_n bits determine the state of UPM signals \overline{LCS}_n and $\overline{LBS}[0:3]$ at each quarter phase of the bus clock. When $LCRR[CLKDIV] = 2$, CST_2 and CST_4 are ignored and the external has the values defined by CST_1 and CST_3 but extended to half the clock cycle in duration. The same interpretation occurs for the BST_n bits when $LCRR[CLKDIV] = 2$.



Figure 10-59. UPM RAM Word Field Descriptions

Table 10-30 describes RAM word fields.

Table 10-30. UPM RAM Word Field Descriptions

Bits	Name	Description
0	CST1	Chip select timing 1. Defines the state (0 or 1) of \overline{LCS}_n during bus clock quarter phase 1 if $LCRR[CLKDIV] = 4$ or 8 . Defines the state (0 or 1) of \overline{LCS}_n during bus clock half phase 1 if $LCRR[CLKDIV] = 2$.

Table 10-30. UPM RAM Word Field Descriptions (continued)

Bits	Name	Description
1	CST2	Chip select timing 2. Defines the state (0 or 1) of \overline{LCSn} during bus clock quarter phase 2 if LCRR[CLKDIV] = 4 or 8. Ignored when LCRR[CLKDIV] = 2.
2	CST3	Chip select timing 3. Defines the state (0 or 1) of \overline{LCSn} during bus clock quarter phase 3 if LCRR[CLKDIV] = 4 or 8. Defines the state (0 or 1) of \overline{LCSn} during bus clock half phase 2 if LCRR[CLKDIV] = 2.
3	CST4	Chip select timing 4. Defines the state (0 or 1) of \overline{LCSn} during bus clock quarter phase 4 if LCRR[CLKDIV] = 4 or 8. Ignored when LCRR[CLKDIV] = 2.
4	BST1	Byte select timing 1. Defines the state (0 or 1) of \overline{LBS} during bus clock quarter phase 1 (LCRR[CLKDIV] = 4 or 8) or bus clock half phase 1 (LCRR[CLKDIV] = 2), in conjunction with BR η [PS] and LA[30:31].
5	BST2	Byte select timing 2. Defines the state (0 or 1) of \overline{LBS} during bus clock quarter phase 2 (LCRR[CLKDIV] = 4 or 8), in conjunction with BR η [PS] and LA[30:31]. Ignored when LCRR[CLKDIV] = 2.
6	BST3	Byte select timing 3. Defines the state (0 or 1) of \overline{LBS} during bus clock quarter phase 3 (LCRR[CLKDIV] = 4 or 8) or bus clock half phase 2 (LCRR[CLKDIV] = 2), in conjunction with BR η [PS] and LA[30:31].
7	BST4	Byte select timing 4. Defines the state (0 or 1) of \overline{LBS} during bus clock quarter phase 4 (LCRR[CLKDIV] = 4 or 8), in conjunction with BR η [PS] and LA[30:31]. Ignored when LCRR[CLKDIV] = 2.
8–9	G0L	General purpose line 0 lower. Defines the state of LGPL0 during the bus clock quarter phases 1 and 2 (first half phase). 00 Value defined by M η MR[G0CL] 01 Reserved 10 Asserted 11 Negated
10–11	G0H	General purpose line 0 higher. Defines the state of LGPL0 during the bus clock quarter phases 3 and 4 (second half phase). 00 Value defined by M η MR[G0CL] 01 Reserved 10 Asserted 11 Negated
12	G1T1	General purpose line 1 timing 1. Defines the state (0 or 1) of LGPL1 during bus clock quarter phases 1 and 2 (first half phase).
13	G1T3	General purpose line 1 timing 3. Defines the state (0 or 1) of LGPL1 during bus clock quarter phases 3 and 4 (second half phase)
14	G2T1	General purpose line 2 timing 1. Defines state (0 or 1) of LGPL2 during bus clock quarter phases 1 and 2 (first half phase).
15	G2T3	General purpose line 2 timing 3. Defines the state (0 or 1) of LGPL2 during bus clock quarter phases 3 and 4 (second half phase).
16	G3T1	General purpose line 3 timing 1. Defines the state (0 or 1) of LGPL3 during bus clock quarter phases 1 and 2 (first half phase).
17	G3T3	General purpose line 3 timing 3. Defines the state (0 or 1) of LGPL3 during bus clock quarter phases 3 and 4 (second half phase).

Table 10-30. UPM RAM Word Field Descriptions (continued)

Bits	Name	Description
18	G4T1/DLT3	General purpose line 4 timing 1/delay time 3. The function of this bit is determined by $MnMR[GPL4]$. If $MnMR[GPL4] = 0$ and $LGPL4/LUPWAIT$ signal functions as an output ($LGPL4$), $G4T1/DLT3$ defines the state (0 or 1) of $LGPL4$ during bus clock quarter phases 1 and 2 (first half phase). If $MnMR[GPL4] = 1$ and $LGPL4/LUPWAIT$ functions as an input ($LUPWAIT$), if a read burst or single read is executed, $G4T1/DLT3$ defines the sampling of the data bus as follows: 0 In the current word, the data bus should be sampled at the start of bus clock quarter phase 1 of the next bus clock cycle. 1 In the current word, the data bus should be sampled at the start of bus clock quarter phase 3 of the current bus clock cycle.
19	G4T3/WAEN	General purpose line 4 timing 3/wait enable. Bit function is determined by $MnMR[GPL4]$. If $MnMR[GPL4] = 0$ and $LGPL4/LUPWAIT$ signal functions as an output ($LGPL4$), $G4T3/WAEN$ defines the state (0 or 1) of $LGPL4$ during bus clock quarter phases 3 and 4 (second half phase). If $MnMR[GPL4] = 1$ and $LGPL4/LUPWAIT$ functions as an input ($LUPWAIT$), $G4T3/WAEN$ is used to enable the wait mechanism: 0 $LUPWAIT$ detection is disabled. 1 $LUPWAIT$ is enabled. If $LUPWAIT$ is detected as being asserted, a freeze in the 3 external signals logical values occurs until $LUPWAIT$ is detected as being negated.
20	G5T1	General purpose line 5 timing 1. Defines the state (0 or 1) of $LGPL5$ during bus clock quarter phases 1 and 2 (first half phase).
21	G5T3	General purpose line 5 timing 3. Defines the state (0 or 1) of $LGPL5$ during bus clock quarter phases 3 and 4 (second half phase).
22–23	REDO	Redo current RAM word. Defines the number of times to execute the current RAM word. 00 Once (normal operation) 01 Twice 10 Three times 11 Four times
24	LOOP	Loop start/end. The first RAM word in the RAM array where $LOOP$ is 1 is recognized as the loop start word. The next RAM word where $LOOP$ is 1 is the loop end word. RAM words between, and including the start and end words, are defined as part of the loop. The number of times the UPM executes this loop is defined in the corresponding loop fields of the $MnMR$. 0 The current RAM word is not the loop start word or loop end word. 1 The current RAM word is the start or end of a loop. Note: AMX must not be changed from its previous value in any RAM word which begins a loop.
25	EXEN	Exception enable. Allows branching to an exception pattern at the exception start address (EXS). When an internal bus monitor time-out exception is recognized and $EXEN$ in the RAM word is set, the UPM branches to the special exception start address (EXS) and begins operating as the pattern defined there specifies. The user should provide an exception pattern to negate signals controlled by the UPM in a controlled fashion. For DRAM control, a handler should negate RAS and CAS to prevent data corruption. If $EXEN = 0$, exceptions are ignored by UPM (but not by the local bus) and execution continues. After the UPM branches to the exception start address, it continues reading until the $LAST$ bit is set in the RAM word. 0 The UPM continues executing the remaining RAM words, ignoring any internal bus monitor time-out. 1 The current RAM word allows a branch to the exception pattern after the current cycle if an exception condition is detected.

Table 10-30. UPM RAM Word Field Descriptions (continued)

Bits	Name	Description
26–27	AMX	<p>Address multiplexing. Determines the source of LAD[0:31] during a LALE phase. Any change in the AMX field initiates a new LALE (address) phase.</p> <p>00 LAD[0:31] is the non-multiplexed address. For example, column address.</p> <p>01 Reserved</p> <p>10 LAD[0:31] is the address multiplexed according to $MnMR[AM]$. For example, row address.</p> <p>11 LAD[0:31] is the contents of MAR. Used, for example, to initialize a mode.</p> <p>Note: AMX must not be changed from its previous value in any RAM word which begins a loop.</p> <p>Note: Source ID debug mode is supported only for the AMX = 00 setting.</p>
28	NA	<p>Next burst address. Determines when the address is incremented during a burst access.</p> <p>0 The address increment function is disabled.</p> <p>1 The address is incremented in the next cycle. In conjunction with the $BRn[PS]$, the increment value of LA[27:31] is 1, 2, or 4 for port sizes of 8-bits, 16-bits and 32-bits, respectively.</p>
29	UTA	<p>UPM transfer acknowledge. Indicates assertion of transfer acknowledge in the current cycle.</p> <p>0 Transfer acknowledge is not asserted in the current cycle.</p> <p>1 Transfer acknowledge is asserted in the current cycle.</p>
30	TODT	<p>Turn-on disable timer. The disable timer associated with each UPM allows a minimum time to be guaranteed between two successive accesses to the same memory bank. This feature is critical when DRAM requires a RAS precharge time. TODT turns the timer on to prevent another UPM access to the same bank until the timer expires. The disable timer period is determined in $MnMR[DSn]$. The disable timer does not affect memory accesses to different banks. Note that TODT must be set together with LAST, otherwise it is ignored.</p> <p>0 The disable timer is turned off.</p> <p>1 The disable timer for the current bank is activated preventing a new access to the same bank (when controlled by the UPMs) until the disable timer expires. For example, precharge time.</p>
31	LAST	<p>Last word. When LAST is read in a RAM word, the current UPM pattern terminates and control signal timing set in the RAM word is applied to the current (and last) cycle. However, if the disable timer is activated and the next access is to the same bank, execution of the next UPM pattern is held off and the control signal values specified in the last word are extended in duration for the number of clock cycles specified in $MnMR[DSn]$.</p> <p>0 The UPM continues executing RAM words.</p> <p>1 Indicates the last RAM word in the program. Service to the UPM request is done after this cycle concludes.</p>

10.4.4.4.2 Chip-Select Signal Timing ($CSTn$)

If $BRn[MSEL]$ of the accessed bank selects a UPM on the currently requested cycle, the UPM manipulates the \overline{LCSn} for that bank with timing as specified in the UPM RAM word $CSTn$ fields. The selected UPM affects only the assertion and negation of the appropriate \overline{LCSn} signal. The state of the selected \overline{LCSn}

signal of the corresponding bank depends on the value of each CST_n bit. Figure 10-60 shows how UPMs control \overline{LCS}_n signals.

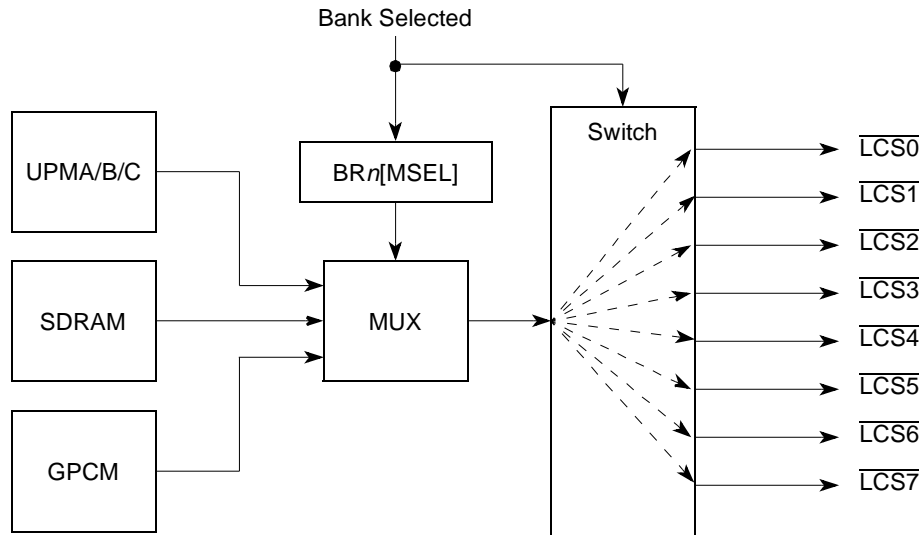


Figure 10-60. \overline{LCS}_n Signal Selection

10.4.4.4.3 Byte Select Signal Timing (BST_n)

If $BR_n[MSEL]$ of the accessed memory bank selects a UPM on the currently requested cycle, the selected UPM affects the assertion and negation of the appropriate $\overline{LBS}[0:3]$ signal. The timing of all four byte-select signals is specified in the RAM word. However, $\overline{LBS}[0:3]$ are also controlled by the port size of the accessed bank, the number of bytes to transfer, and the address accessed.

Figure 10-61 shows how UPMs control $\overline{LBS}[0:3]$.

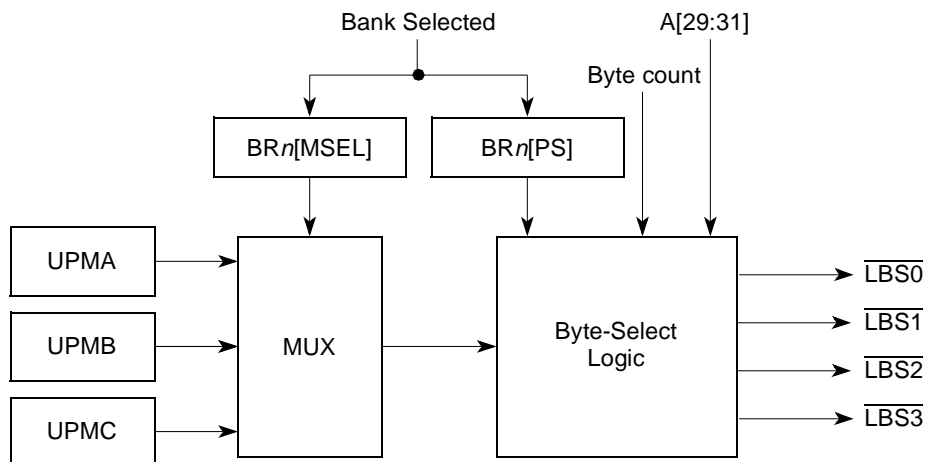


Figure 10-61. \overline{LBS} Signal Selection

The uppermost byte select (\overline{LBS}_0), when asserted, indicates that $LAD[0:7]$ contains valid data during a cycle. Likewise, \overline{LBS}_1 indicates that $LAD[8:15]$ contains valid data, \overline{LBS}_2 indicates that $LAD[16:23]$ contains valid data, and \overline{LBS}_3 indicates that $LAD[24:31]$ contains valid data. For a UPM refresh timer

request, all $\overline{\text{LBS}}[0:3]$ signals are asserted/negated by the UPM according to the refresh pattern only. Following any internal bus monitor exception, the $\overline{\text{LBS}}[0:3]$ signals are negated regardless of the exception handling provided by any UPM exception pattern to prevent spurious writes to external RAM.

10.4.4.4.4 General-Purpose Signals ($GnTn$, GO_n)

The general-purpose signals ($LGPL[0:5]$) each have two bits in the RAM word that define the logical value of the signal to be changed at the rising edge of the bus clock and/or at the falling edge of the bus clock. $LGPL0$ offers enhancements beyond the other $LGPLn$ lines.

$GPL0$ can be controlled by an address line specified in $MnMR[GOCL]$. To use this feature, GOH and GOL should be set in the RAM word. For example, for a SIMM with multiple banks, this address line can be used to switch between internal memory device banks.

10.4.4.4.5 Loop Control (LOOP)

The LOOP bit in the RAM word specifies the beginning and end of a set of UPM RAM words that are to be repeated. The first time $LOOP = 1$, the memory controller recognizes it as a loop start word and loads the memory loop counter with the corresponding contents of the loop field shown in Table 10-31. The next RAM word for which $LOOP = 1$ is recognized as a loop end word. When it is reached, the loop counter is decremented by one.

Continued loop execution depends on the loop counter. If the counter is not zero, the next RAM word executed is the loop start word. Otherwise, the next RAM word executed is the one after the loop end word. Loops can be executed sequentially but cannot be nested. Also, special care must be taken: LAST and LOOP must not be set together.

Table 10-31. $MnMR$ Loop Field Use

Request Serviced	Loop Field
Read single-beat cycle	RLF
Read burst cycle	RLF
Write single-beat cycle	WLF
Write burst cycle	WLF
Refresh timer expired	TLF
RUN command	RLF

10.4.4.4.6 Repeat Execution of Current RAM Word (REDO)

The REDO function is useful for wait-state insertion in a long routine that would otherwise need too many RAM words. Setting the REDO bits of the RAM word to a nonzero value causes the UPM to re-execute the current RAM word up to three more times, as defined in the REDO field of the current RAM word.

Special care must be taken in the following cases:

- When UTA and REDO are set together, TA is asserted the number of times specified by the REDO function.
- When NA and REDO are set together, the address is incremented the number of times specified by the REDO function.

- When LOOP and REDO are set together, the loop mechanism works as usual and the line is repeated according to the REDO function.
- LAST and REDO must not be set together.
- REDO should not be used within the exception routine.

10.4.4.4.7 Address Multiplexing (AMX)

The address lines can be controlled by the pattern the user provides in the UPM. The address multiplex bits can choose between driving the transaction address, driving it according to the multiplexing specified by the $M_nMR[AM]$ field, or driving the MAR contents on the address signals. In all cases, LA[27:31] of the LBC are driven by the five lsb's of the address selected by AMX, regardless of whether the NA bit of the RAM word is used to increment the current address. The effect of NA = 1 is visible only when AMX = 00 chooses the column address.

Table 10-32 shows how $M_nMR[AM]$ settings affect address multiplexing when the RAM word AMX = 10. The 16 msbs of the LAD[0:31] bus during an address phase are driven with zero in the AMX = 10 case.

Table 10-32. UPM Address Multiplexing

AM	LAD[0:31] as Address Signals	A0– A15	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
000	Signal driven on external signal when address multiplexing is enabled— RAM word AMX = 10	0	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23
001		0	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22
010		0	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21
011		0	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20
100		0	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
101		0	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18

Note that any change to AMX from one RAM word to the next RAM word executed results in an address phase on LAD[0:31] with the assertion of LALE for the number of cycles set for LALE in the OR $_n$ and LCRR registers. LGPL[0:5] signals maintain the value specified in the RAM word during the LALE phase.

NOTE

AMX must not be changed from its previous value in any RAM word which begins a loop.

10.4.4.4.8 Data Valid and Data Sample Control (UTA)

When a read access is handled by the UPM, and the UTA bit is 1 (data is to be sampled by the LBC), the value of the DLT3 bit in the same RAM word, in conjunction with $M_nMR[GPL_n4DIS]$, determines when the data input is sampled by the LBC as follows:

- If $M_nMR[GPL_n4DIS] = 1$ (G4T4/DLT3 functions as DLT3) and DLT3 = 1 in the RAM word, data is latched on the falling edge of the bus clock instead of the rising edge. The LBC samples the data on the next falling edge of the bus clock, which is during the middle of the current bus cycle. This feature should be used only in systems without external synchronous bus devices that require mid-cycle sampling.

- If $GPL_n4DIS = 0$ (G4T4/DLT3 functions as G4T4), or if $GPL_n4DIS = 1$ but $DLT3 = 0$, data is latched on the rising edge of the bus clock, which occurs at the end of the current bus clock cycle (normal operation).

Figure 10-62 shows how data sampling is controlled by the UPM.

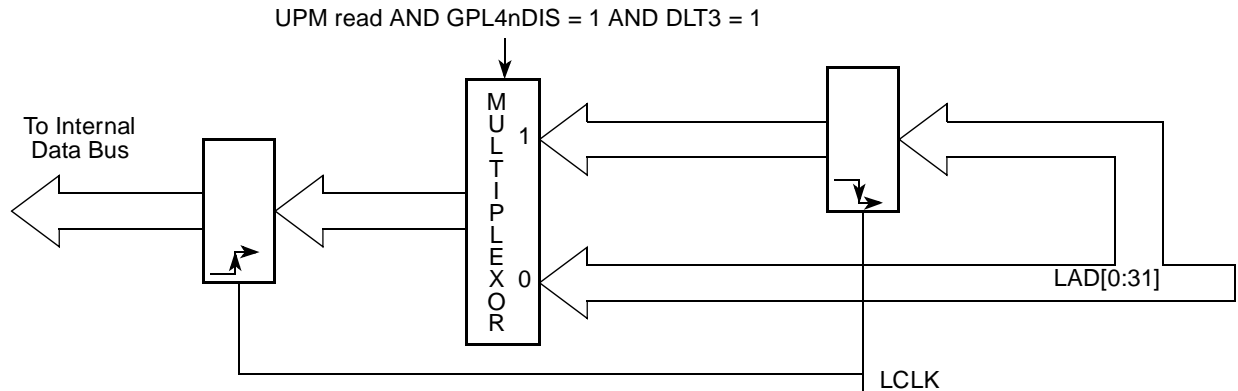


Figure 10-62. UPM Read Access Data Sampling

10.4.4.4.9 $\overline{LGPL}[0:5]$ Signal Negation (LAST)

When the LAST bit is read in a RAM word, the current UPM pattern is terminated at the end of the current cycle. On the next cycle (following LAST) all UPM signals are negated unconditionally (driven to logic one), unless there is a back-to-back UPM request pending. In this case, the signal values for the cycle following the one in which the LAST bit was set are taken from the first RAM word of the pending UPM routine.

10.4.4.4.10 Wait Mechanism (WAEN)

The WAEN bit in the RAM array word can be used to enable the UPM wait mechanism in selected UPM RAM words. If the UPM reads a RAM word with WAEN set, the external LUPWAIT signal is sampled and synchronized by the memory controller as if it were an asynchronous signal. The WAEN bit is ignored if $LAST = 1$ in the same RAM word.

Synchronization of LUPWAIT starts at the rising edge of the bus clock and takes at least 1 bus cycle to complete. If LUPWAIT is asserted and $WAEN = 1$ in the current UPM word, the UPM is frozen until LUPWAIT is negated. The value of external signals driven by the UPM remains as indicated in the previous RAM word. Setting the WAEN bit for the first RAM word does not have any effect since the first RAM word signifies the start of a new bus cycle and the initial values of the signals driven onto the bus should be present. When LUPWAIT is negated, the UPM continues normal functions. Note that during WAIT cycles, the UPM does not handle data.

Figure 10-63 shows how the WAEN bit in the word read by the UPM and the LUPWAIT signal are used to hold the UPM in a particular state until LUPWAIT is negated. As the example shows, the \overline{LCS}_n and \overline{LGPL}_I states and the WAEN value are frozen until LUPWAIT is recognized as negated. WAEN is typically set before the line that contains $UTA = 1$. Note that if WAEN and NA are both set in the same

RAM word, NA causes the burst address to increment once as normal regardless of whether the UPM freezes.

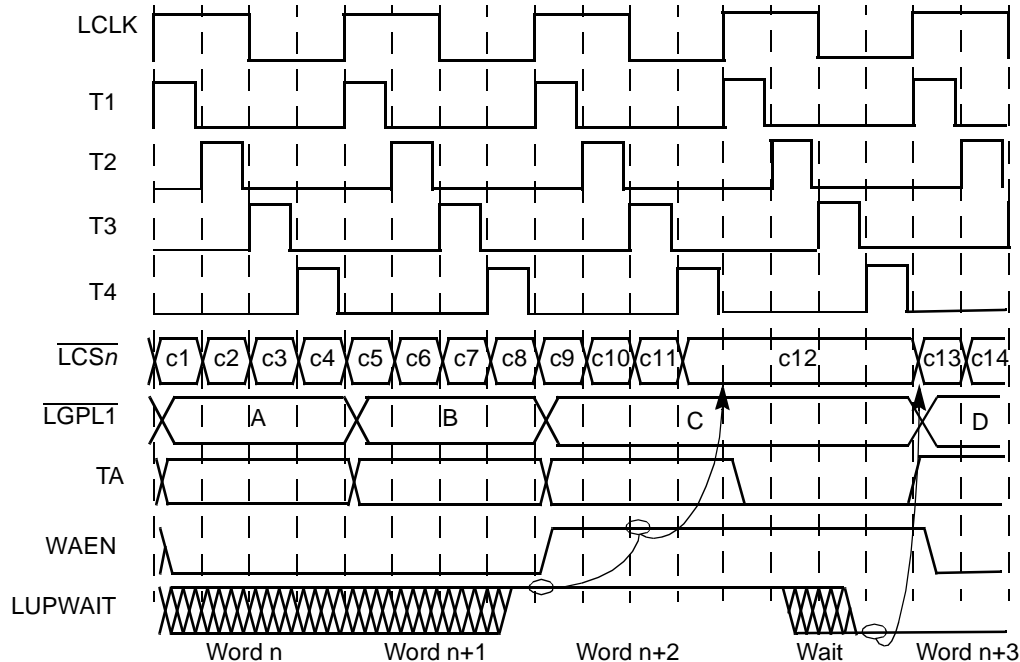


Figure 10-63. Effect of LUPWAIT Signal

10.4.4.5 Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge

If LUPWAIT is to be considered an asynchronous signal, which can be asserted/negated at any time, no UPM RAM word must contain both WAEN = 1 and UTA = 1 simultaneously.

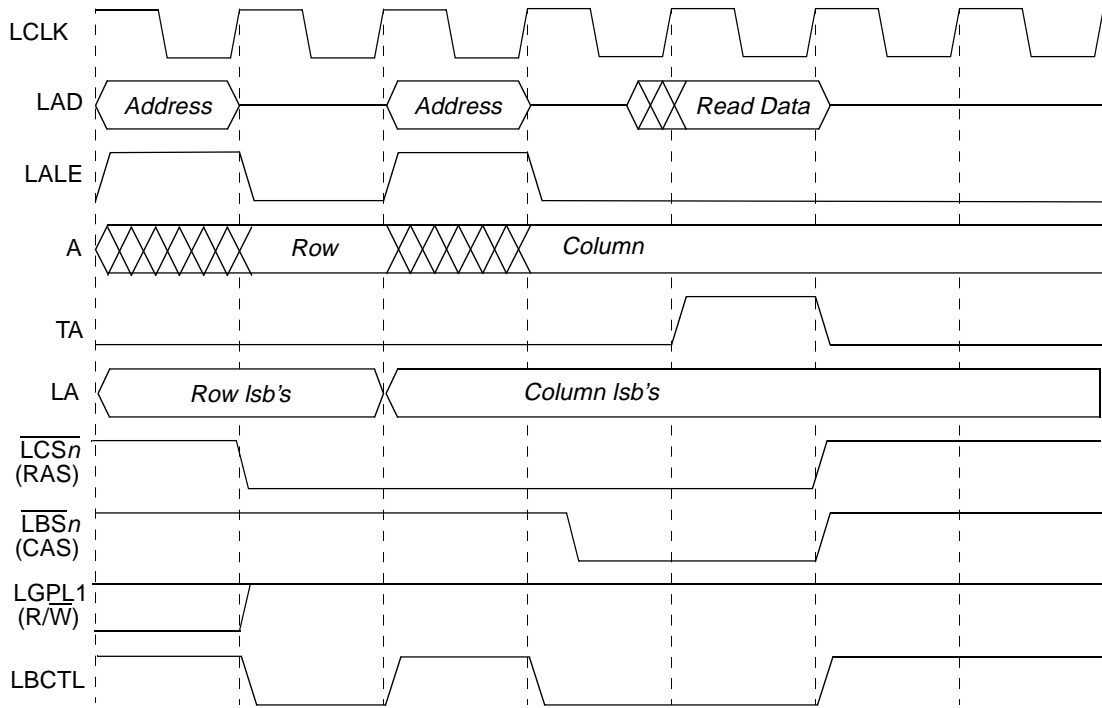
However, programming WAEN = 1 and UTA = 1 in the same RAM word allows the UPM to treat LUPWAIT as a synchronous signal, which must meet set-up and hold times in relation to the rising edge of the bus clock. In this case, as soon as UPM samples LUPWAIT negated on the rising edge of the bus clock, it immediately generates an internal transfer acknowledge, which allows a data transfer one bus clock cycle later. The generation of transfer acknowledge is early because LUPWAIT is not re-synchronized, and the acknowledge occurs regardless of whether the UPM was already frozen in WAIT cycles or not. This feature allows the synchronous negation of LUPWAIT to effect a data transfer, even if UTA, WAEN, and LAST are set simultaneously.

10.4.4.6 Extended Hold Time on Read Accesses

Slow memory devices that take a long time to turn off their data bus drivers on read accesses should choose some non-zero combination of ORn[TRLX] and ORn[EHTR]. The next access after a read access to the slow memory device is delayed by the number of clock cycles specified in the ORn register in addition to any existing bus turnaround cycle.

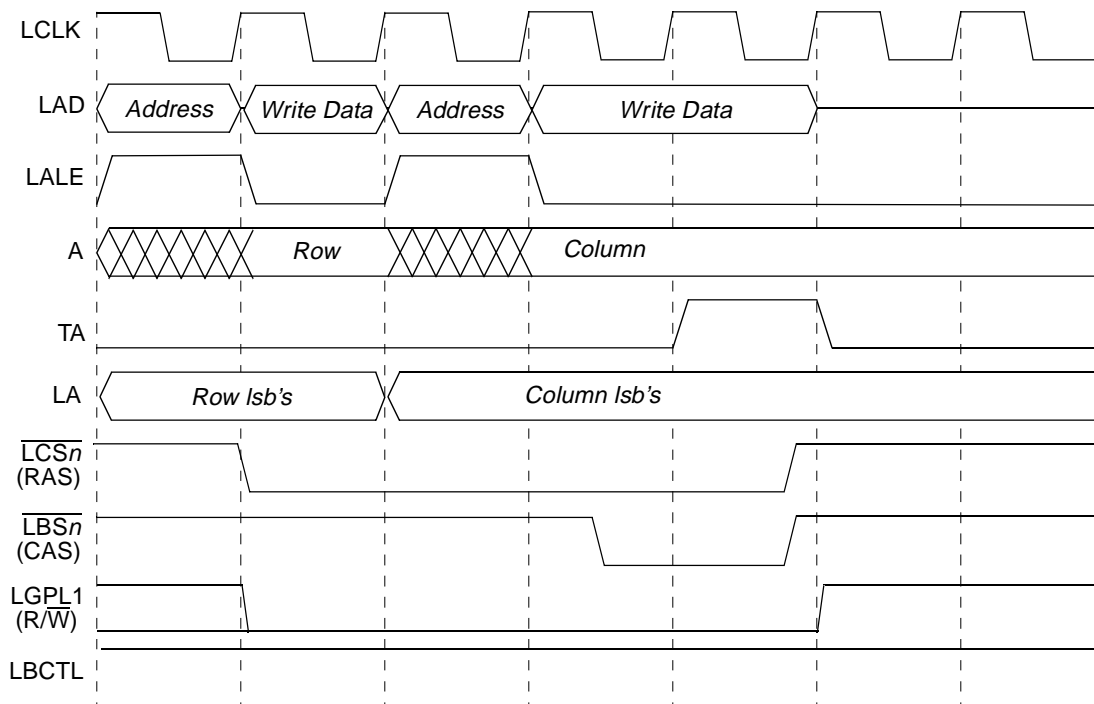
10.4.4.7 Memory System Interface Example Using UPM

Connecting the local bus UPM controller to a DRAM device requires a detailed examination of the timing diagrams representing the possible memory cycles that must be performed when accessing this device. This section presents timing diagrams for various UPM configurations, using fast-page mode DRAM as an example, with $LCRR[CLKDIV] = 4$ or 8 . These examples may not represent the timing necessary for any specific device used with the LBC. Here, $LGPL1$ is programmed to drive R/\overline{W} of the DRAM, although any $LGPLn$ signal can be used for this purpose.



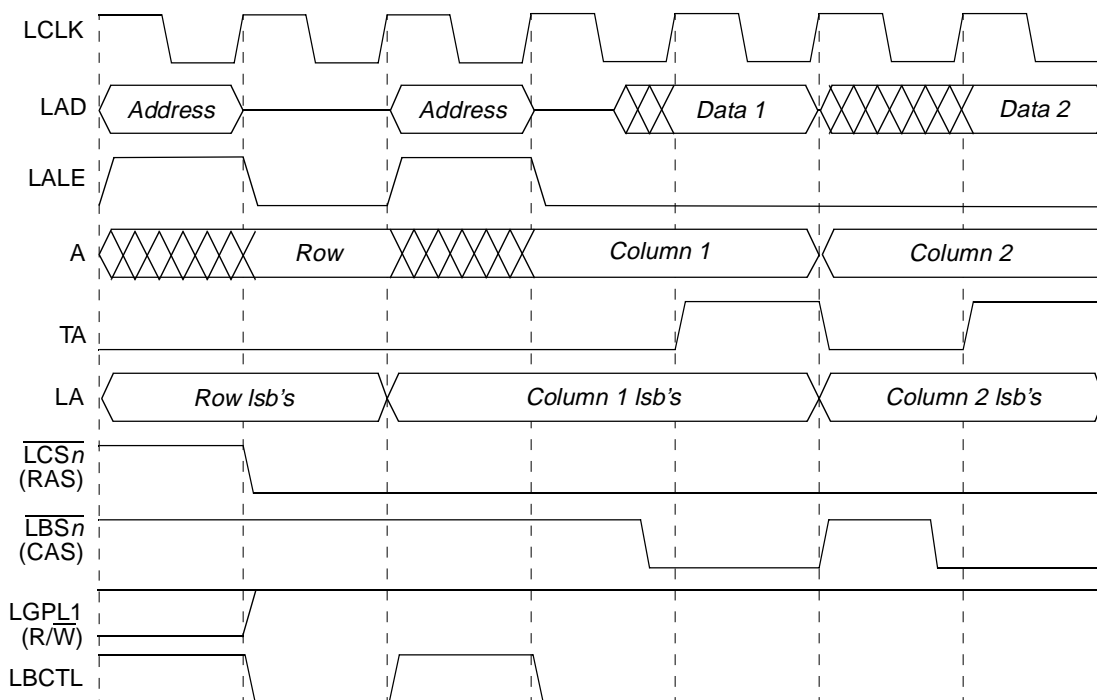
cst1	0	LALE pause (due to change in AMX)	0	0	Bit 0	
cst2	0		0	0	Bit 1	
cst3	0		0	0	Bit 2	
cst4	0		0	0	Bit 3	
bst1	1		1	0	Bit 4	
bst2	1		0	0	Bit 5	
bst3	1		0	0	Bit 6	
bst4	1		0	0	Bit 7	
g0i0					Bit 8	
g0i1					Bit 9	
g0h0					Bit 10	
g0h1					Bit 11	
g1t1	1		1	1	Bit 12	
g1t3	1		1	1	Bit 13	
g2t1					Bit 14	
g2t3					Bit 15	
g3t1					Bit 16	
g3t3					Bit 17	
g4t1					Bit 18	
g4t3					Bit 19	
g5t1					Bit 20	
g5t3					Bit 21	
redo[0]					Bit 22	
redo[1]					Bit 23	
loop	0			0	0	Bit 24
exen	0			0	0	Bit 25
amx0	1			0	0	Bit 26
amx1	0			0	0	Bit 27
na	0			0	0	Bit 28
uta	0			0	1	Bit 29
todt	0			0	1	Bit 30
last	0		0	1	Bit 31	
	RSS	RSS+1	RSS+1	RSS+2		

Figure 10-64. Single-Beat Read Access to FPM DRAM



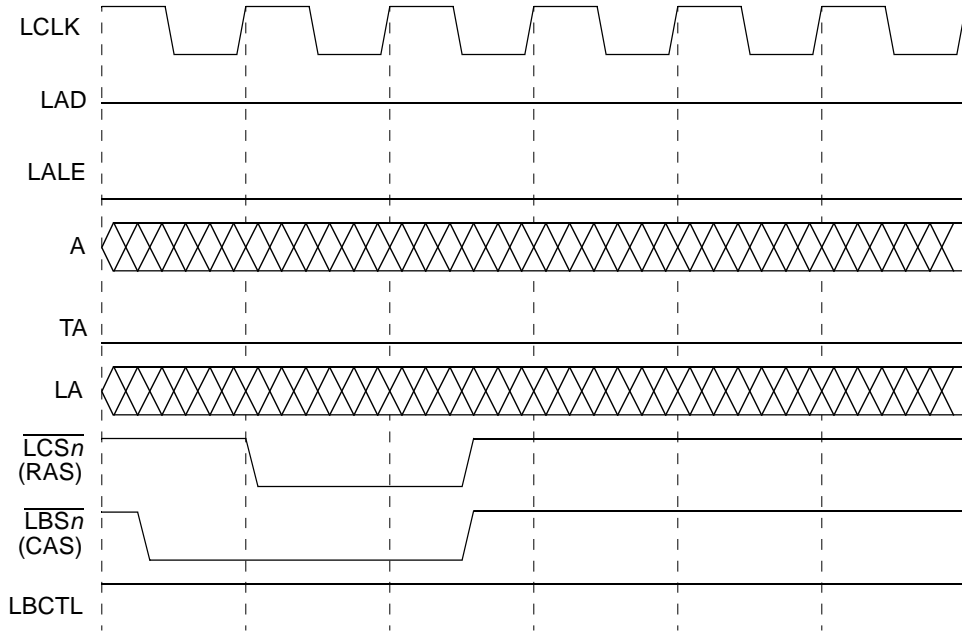
cst1	0	LALE pause (due to change in AMX)	0	0	Bit 0
cst2	0		0	0	Bit 1
cst3	0		0	0	Bit 2
cst4	0		0	1	Bit 3
bst1	1		1	0	Bit 4
bst2	1		1	0	Bit 5
bst3	1		0	0	Bit 6
bst4	1		0	1	Bit 7
g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1	0		0	0	Bit 12
g1t3	0		0	0	Bit 13
g2t1					Bit 14
g2t3					Bit 15
g3t1				Bit 16	
g3t3				Bit 17	
g4t1				Bit 18	
g4t3				Bit 19	
g5t1				Bit 20	
g5t3				Bit 21	
redo[0]				Bit 22	
redo[1]				Bit 23	
loop	0	0	0	Bit 24	
exen	0	0	0	Bit 25	
amx0	1	0	0	Bit 26	
amx1	0	0	0	Bit 27	
na	0	0	0	Bit 28	
uta	0	0	1	Bit 29	
todt	0	0	1	Bit 30	
last	0	0	1	Bit 31	
	WSS	WSS+1	WSS+1	WSS+2	

Figure 10-65. Single-Beat Write Access to FPM DRAM



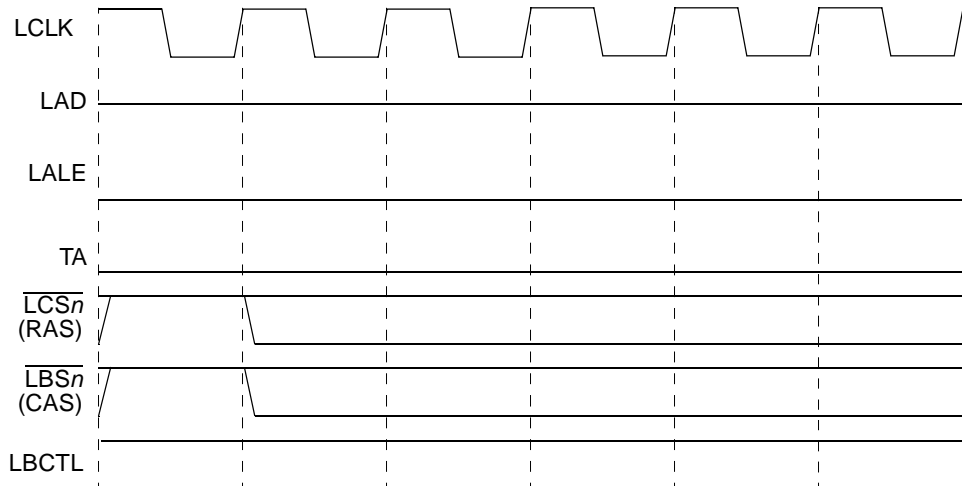
cst1	0	LALE pause (due to change in AMX)	0	0	1	Bit 0
cst2	0		0	0	1	Bit 1
cst3	0		0	0	1	Bit 2
cst4	0		0	0	1	Bit 3
bst1	1		1	0	1	Bit 4
bst2	1		1	0	1	Bit 5
bst3	1		1	0	1	Bit 6
bst4	1		0	0	1	Bit 7
g0i0						Bit 8
g0i1						Bit 9
g0h0						Bit 10
g0h1						Bit 11
g1t1	1		1	1	1	Bit 12
g1t3	1		1	1	1	Bit 13
g2t1						Bit 14
g2t3						Bit 15
g3t1						Bit 16
g3t3						Bit 17
g4t1						Bit 18
g4t3						Bit 19
g5t1						Bit 20
g5t3						Bit 21
redo[0]						Bit 22
redo[1]						Bit 23
loop	0		1	1	0	Bit 24
exen	0		0	1	0	Bit 25
amx0	1		0	0	0	Bit 26
amx1	0		0	0	0	Bit 27
na	0	0	1	0	Bit 28	
uta	0	0	1	0	Bit 29	
todt	0	0	0	1	Bit 30	
last	0	0	0	1	Bit 31	
	RBS		RBS+1	RBS+2	RBS+3	

Figure 10-66. Burst Read Access to FPM DRAM Using LOOP (Two Beats Shown)



cst1	1	0	0	Bit 0
cst2	1	0	0	Bit 1
cst3	1	0	1	Bit 2
cst4	1	0	1	Bit 3
bst1	1	0	0	Bit 4
bst2	0	0	0	Bit 5
bst3	0	0	1	Bit 6
bst4	0	0	1	Bit 7
g0l0				Bit 8
g0l1				Bit 9
g0h0				Bit 10
g0h1				Bit 11
g1t1				Bit 12
g1t3				Bit 13
g2t1				Bit 14
g2t3				Bit 15
g3t1				Bit 16
g3t3				Bit 17
g4t1				Bit 18
g4t3				Bit 19
g5t1				Bit 20
g5t3				Bit 21
redo[0]				Bit 22
redo[1]				Bit 23
loop	0	0	0	Bit 24
exen	0	0	0	Bit 25
amx0	0	0	0	Bit 26
amx1	0	0	0	Bit 27
na	0	0	0	Bit 28
uta	0	0	0	Bit 29
todt	0	0	1	Bit 30
last	0	0	1	Bit 31
	PTS	PTS+1	PTS+2	

Figure 10-67. Refresh Cycle (CBR) to FPM DRAM



cst1	1	Bit 0
cst2	1	Bit 1
cst3	1	Bit 2
cst4	1	Bit 3
bst1	1	Bit 4
bst2	1	Bit 5
bst3	1	Bit 6
bst4	1	Bit 7
g0i0		Bit 8
g0i1		Bit 9
g0h0		Bit 10
g0h1		Bit 11
g1t1		Bit 12
g1t3		Bit 13
g2t1		Bit 14
g2t3		Bit 15
g3t1		Bit 16
g3t3		Bit 17
g4t1		Bit 18
g4t3		Bit 19
g5t1		Bit 20
g5t3		Bit 21
redo[0]		Bit 22
redo[1]		Bit 23
loop	0	Bit 24
exen	0	Bit 25
amx0	0	Bit 26
amx1	0	Bit 27
na	0	Bit 28
uta	0	Bit 29
todt	1	Bit 30
last	1	Bit 31
EXS		

Figure 10-68. Exception Cycle

10.5 Initialization/Application Information

These sections provide detailed information on interfacing the LBC to peripheral devices.

10.5.1 Interfacing to Peripherals in Multiplexed Address/Data Mode

The local bus can be used either with separate address- and data buses or with a multiplexed address/data bus. These sections provide guidelines for interfacing peripherals in the multiplexed mode.

10.5.1.1 Multiplexed Address/Data Bus and Unmultiplexed Address Signals

To save signals on the local bus, address and data are multiplexed onto the same 32-bit bus. An external latch is needed to demultiplex and reconstruct the original address. No external intelligence is needed, because LALE provides the correct timing to control a standard logic latch. The LAD signals can be directly connected to the data signals of the memory/peripheral.

Transactions on the local bus start with an address phase, where the LBC drives the transaction address on the LAD signals and asserts the LALE signal. This can be used to latch the address and then the LBC can continue with the data phase.

The LBC supports port sizes of 8, 16, and 32 bits. For devices smaller than 32 bits, transactions must be broken down. For this reason, LA[30:31] are driven unmultiplexed. For 8-bit devices, LA[30:31] should be used and for 16 bit devices, LA30 should be used. 32-bit devices use neither of these signals.

In addition, the LBC supports burst transfers (not in the GPCM machine). LA[27:29] are the burst addresses within a natural 32-byte burst. To minimize the amount of address phases needed on the local bus and to optimize the throughput, those signals are driven separately and should be used whenever a device requires the 5 least-significant addresses. Those should not be used from LAD[27:31].

All other addresses, A[0:26], must be reconstructed through the latch.

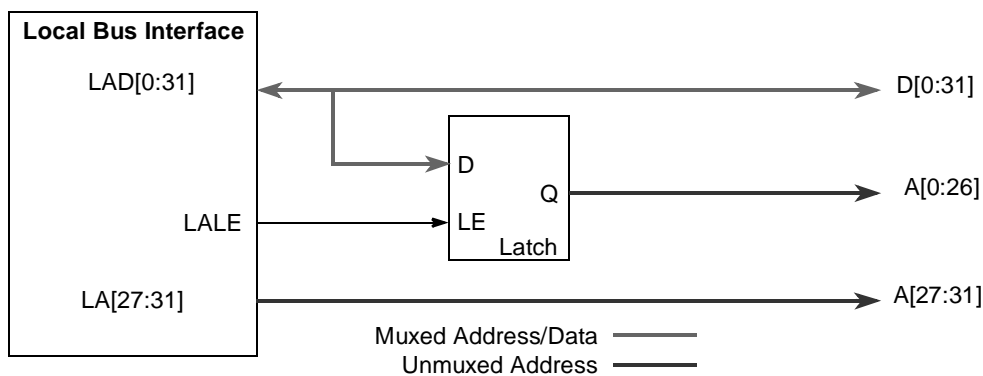


Figure 10-69. Multiplexed Address/Data Bus

10.5.1.2 Peripheral Hierarchy on the Local Bus

To achieve high bus speed interfaces for synchronous SRAMs or SDRAMs, a hierarchy of the memories/peripherals connected to the local bus is suggested, as shown in [Figure 10-70](#).

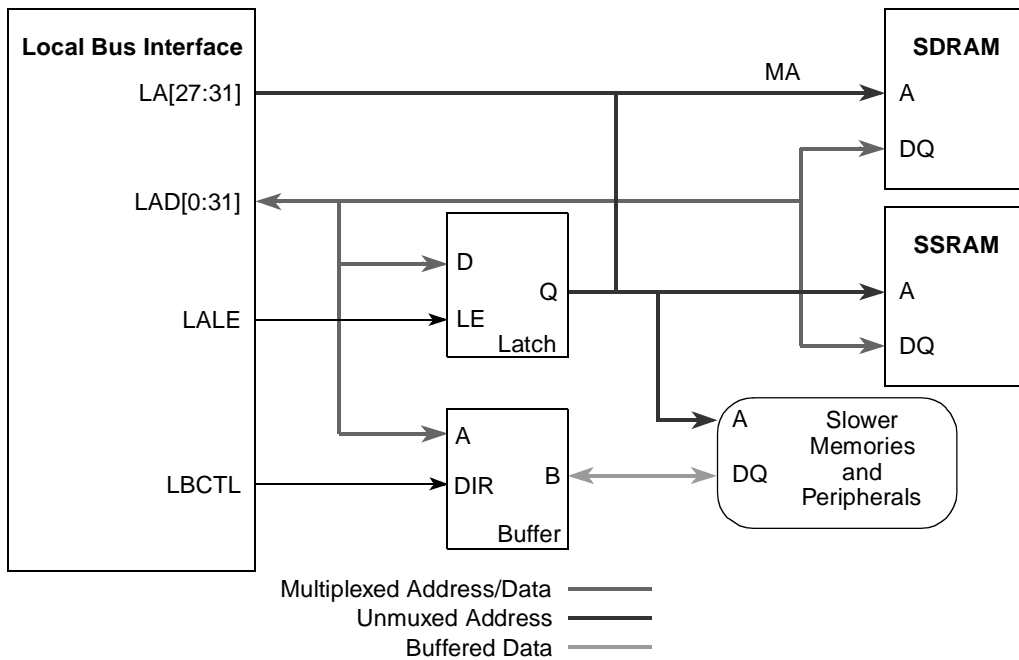


Figure 10-70. Local Bus Peripheral Hierarchy

The multiplexed address/data bus sees the capacitive loading of the data signals of the fast SDRAMs or synchronous SRAMs plus one load for an address latch plus one load for a buffer to the slow memories. The loadings of all other memories and peripherals are hidden behind the buffer and the latch. The system designer needs to investigate the loading scenario and ensure that I/O timings can be met with the loading determined by the connected components.

10.5.1.3 Peripheral Hierarchy on the Local Bus for Very High Bus Speeds

To achieve the highest possible bus speeds on the local bus, it is recommended to reduce the number of devices connected directly to the local bus even further. For those cases probably only one bank of synchronous SRAMs or SDRAMs should be used and instead of using a separate latch and a separate bus transceiver, a bus demultiplexer combining those two functions into one device should be used.

[Figure 10-71](#) shows an example of such a hierarchy. This section is only a guideline; the board designer

must simulate the electric characteristics of the chosen scenario to determine the maximum operating frequency.

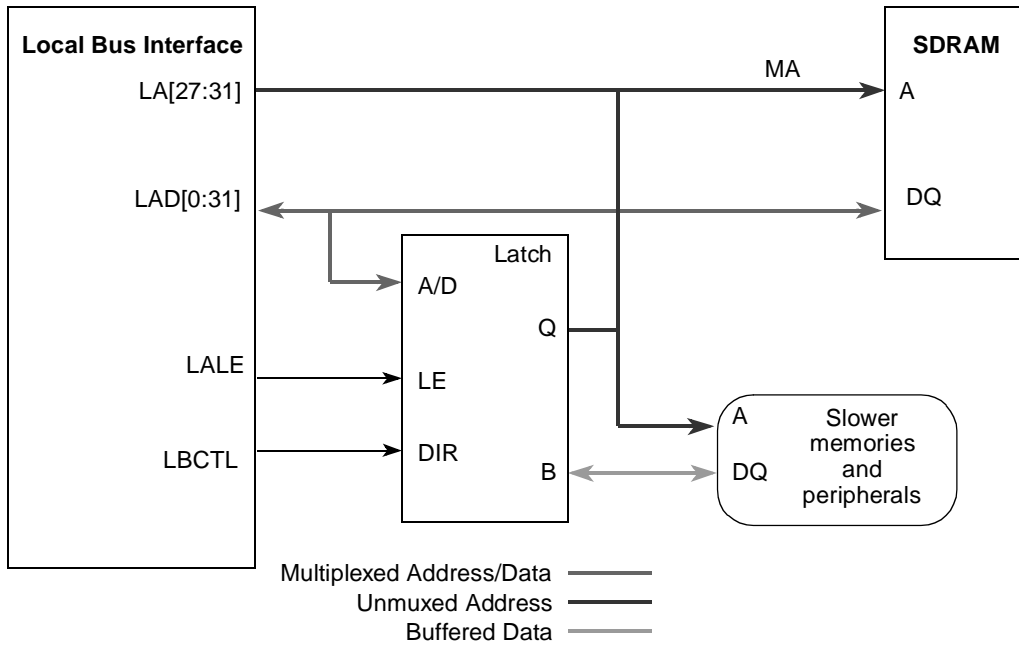


Figure 10-71. Local Bus Peripheral Hierarchy for Very High Bus Speeds

10.5.1.4 GPCM Timings

If a system contains a memory hierarchy with high-speed synchronous memories (SDRAM, synchronous SRAM) and lower speed asynchronous memories (FLASH EPROM and peripherals) the GPCM-controlled memories should be decoupled by buffers to reduce capacitive loading on the bus. Those buffers have to be taken into account for the timing calculations. The GPCM address timings is shown in [Figure 10-72](#).

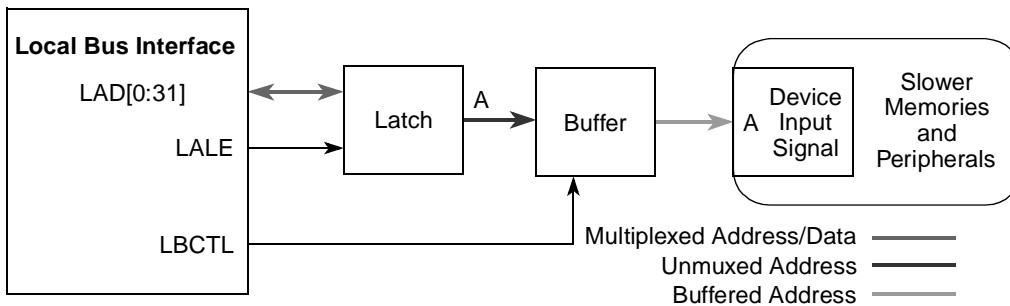


Figure 10-72. GPCM Address Timings

To calculate address setup timing for a slower peripheral/memory device, several parameters have to be added: propagation delay for the address latch, propagation delay for the buffer and the address setup for the actual peripheral. Typical values for the two propagation delays are in the order of 3–6 ns, so for a 166-MHz bus frequency, \overline{LCS} should arrive roughly 3 bus clocks later.

For data timings, only the propagation delay of one buffer plus the actual data setup time must be considered.

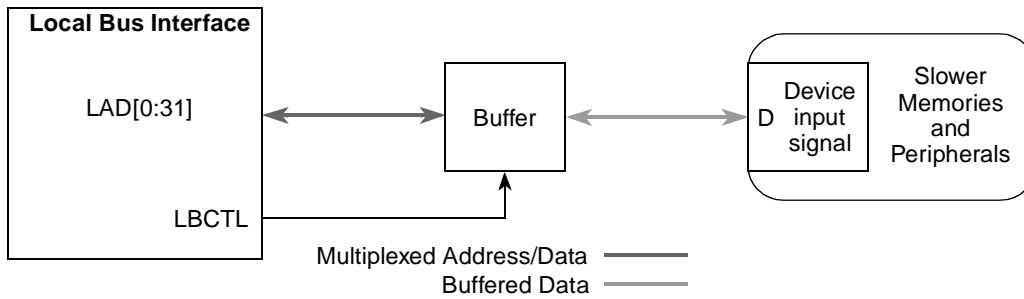


Figure 10-73. GPCM Data Timings

10.5.2 Bus Turnaround

Because the local bus uses multiplexed address and data, special consideration must be given to avoid bus contention at bus turnaround. The following cases must be examined:

- Address phase after previous read
- Read data phase after address phase
- Read-modify-write cycle for parity protected memory banks
- UPM cycles with additional address phases

The bus does not change direction for the following cases, so they need no special attention:

- Continued burst after the first beat
- Write data phase after address phase
- Address phase after previous write

10.5.2.1 Address Phase after Previous Read

During a read cycle, the memory/peripheral drives the bus and the bus transceiver drives LAD. After the data has been sampled, the output drivers of the external device must be disabled. This can take some time; for slow devices, the EHTR feature of the GPCM or the programmability of the UPM should be used to guarantee that those devices have stopped driving the bus when the LBC memory controller ends the bus cycle.

In this case, after the previous cycle ends, LBCTL goes high and changes the direction of the bus transceiver. The LBC then inserts a bus turnaround cycle to avoid contention. The external device has now already placed its data signals in high impedance and no bus contention will occur.

10.5.2.2 Read Data Phase after Address Phase

During the address phase, LAD actively drives the address and LBCTL is high, driving the bus transceivers in the same direction as during a write. After the end of the address phase, LBCTL goes low and changes the direction of the bus transceiver. The LBC places the LAD signals in high impedance after its $t_{dis}(LB)$. The LBCTL will have its new state after $t_{en}(LB)$ and, because this is an asynchronous input,

the transceiver starts to drive those signals after its t_{en} (transceiver) time. The system designer has to ensure, that $[t_{en}(LB) + t_{en}(\text{transceiver})]$ is larger than $t_{dis}(LB)$ to avoid bus contention.

10.5.2.3 Read-Modify-Write Cycle for Parity Protected Memory Banks

Principally, a read-modify-write cycle is a read cycle immediately followed by a write cycle. Because the write cycle has a new address phase in any case, this basically is the same case as an address phase after a previous read.

10.5.2.4 UPM Cycles with Additional Address Phases

The flexibility of the UPM allows the user to insert additional address phases during read cycles by changing the AMX field, therefore turning around the bus during one pattern. The LBC automatically inserts a single bus turnaround cycle if the bus (LAD) was previously high impedance for any reason, such as a read, before LALE is driven and LAD is driven with the new address. The turnaround cycle is not inserted on a write, because the bus was already driven to begin with.

However, bus contention could potentially still occur on the far side of a bus transceiver. It is the responsibility of the designer of the UPM pattern to guarantee that enough idle cycles are inserted in the UPM pattern to avoid this.

10.5.3 Interface to Different Port-Size Devices

The LBC supports 8-, 16-, and 32-bit data port sizes. However, the bus requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 32-bit port must reside on D[0:31], a 16-bit port must reside on D[0:15], and an 8-bit port must reside on D[0:7]. The local bus always tries to

transfer the maximum amount of data on all bus cycles. [Figure 10-74](#) shows the device connections on the data bus.

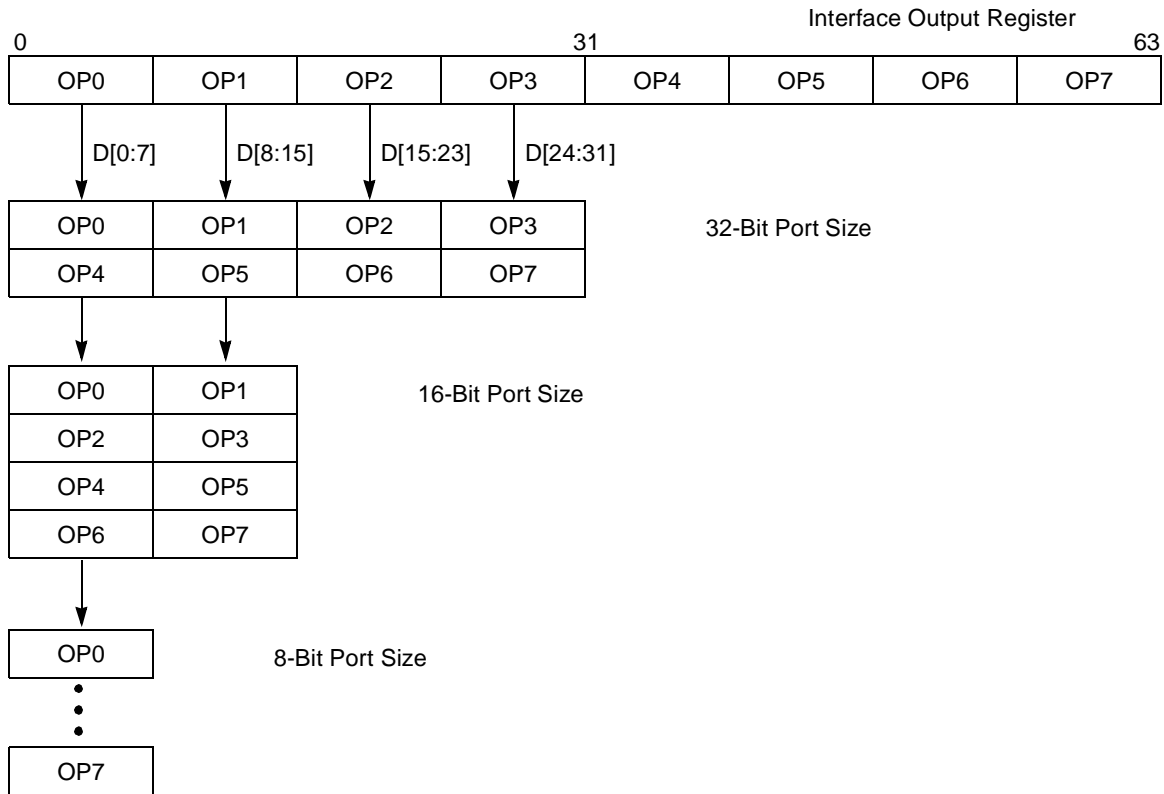


Figure 10-74. Interface to Different Port-Size Devices

[Table 10-33](#) lists the bytes required on the data bus for read cycles.

Table 10-33. Data Bus Requirements For Read Cycle

Transfer Size	Address State ¹ A[29:31]	Port Size/Data Bus Assignments						
		32-Bit				16-Bit		8-Bit
		0-7	8-15	16-23	24-31	0-7	8-15	0-7
Byte	000	OP0 ²	— ³	—	—	OP0	—	OP0
	001	—	OP1	—	—	—	OP1	OP1
	010	—	—	OP2	—	OP2	—	OP2
	011	—	—	—	OP3	—	OP3	OP3
	100	OP4	—	—	—	OP4	—	OP4
	101	—	OP5	—	—	—	OP5	OP5
	110	—	—	OP6	—	OP6	—	OP6
	111	—	—	—	OP7	—	OP7	OP7

Table 10-33. Data Bus Requirements For Read Cycle (continued)

Transfer Size	Address State ¹ A[29:31]	Port Size/Data Bus Assignments						
		32-Bit				16-Bit		8-Bit
		0-7	8-15	16-23	24-31	0-7	8-15	0-7
Half Word	000	OP0	OP1	—	—	OP0	OP1	OP0
	001	—	OP1	OP2	—	—	OP1	OP1
	010	—	—	OP2	OP3	OP2	OP3	OP2
	100	OP4	OP5	—	—	OP4	OP5	OP4
	101	—	OP5	OP6	—	—	OP5	OP5
	110	—	—	OP6	OP7	OP6	OP7	OP6
Word	000	OP0	OP1	OP2	OP3	OP0	OP1	OP0
	100	OP4	OP5	OP6	OP7	OP4	OP5	OP4

¹ Address state is the calculated address for port size.

² OP n : These lanes are read or written during that bus transaction. OP0 is the most-significant byte of a word operand and OP3 is the least-significant byte.

³ — Denotes a byte not driven during that write cycle.

10.5.4 Interfacing to SDRAM

The following subsections provide application information on interfacing to SDRAM.

10.5.4.1 Basic SDRAM Capabilities of the Local Bus

The LBC provides one SDRAM machine for the local bus. Although there is only one machine, multiple chip selects (\overline{LCSn}) can be programmed to support multiple SDRAM devices. Note that no limitation exists on the number of chip selects that can be programmed for SDRAM. This means that $\overline{LCS}[1:7]$ can be programmed to support SDRAM, assuming $\overline{LCS0}$ is reserved for the GPCM to connect to Flash memory.

If multiple chip selects are configured to support SDRAM on the local bus, each SDRAM device should have the same port size and timing parameters. This means that all option registers (OR n) for the SDRAM chip selects should be programmed the same.

NOTE

Although in principle it is possible to mix different port sizes and timing parameters, combinations are limited and this operation is not recommended.

All the chip selects share the same local bus SDRAM mode register (LSDMR) for initialization along with the local bus-assigned SDRAM refresh timer register (LSRT) and the memory refresh timer prescaler register (MPTPR) for refresh.

For refresh, the memory controller supplies auto refresh to SDRAM according to the time interval specified in LSRT and MPTPR as follows:

$$\text{Refresh Period} = \frac{\text{LSRT} \times (\text{MPTPR}[\text{PTP}])}{\text{System Frequency}}$$

This represents the time period required between refreshes. When the refresh timer expires, the memory controller issues a CBR to each chip select. Each CBR is separated by one clock. A refresh timing diagram for multiple chip selects is shown in [Figure 10-52](#) in [Section 10.4.3.11.1](#), “[SDRAM Refresh Timing](#).”

During a memory transaction dispatched to the local bus, the memory controller compares the memory address with the address information of each chip select (programmed with BR_n and OR_n). If the comparison matches a chip select that is controlled by SDRAM, the memory controller requests service to the local bus SDRAM machine, depending on the information in BR_n. Although multiple chip selects may be programmed for SDRAM, only one chip select is active at any given time; thus, multiple chip selects can share the same SDRAM machine.

10.5.4.2 Maximum Amount of SDRAM Supported

[Table 10-34](#) summarizes information based on SDRAM data sheets supplied by Micron.

Table 10-34. Micron SDRAM Devices

SDRAM Device	64 Mbit				128 Mbit				256 Mbit				512 Mbit			
	x4	x8	x16	x32	x4	x8	x16	x32	x4	x8	x16	x32	x4	x8	x16	x32
I/O Port	x4	x8	x16	x32	x4	x8	x16	x32	x4	x8	x16	x32	x4	x8	x16	x32
Bank	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Row	12	12	12	11	12	12	12	12	13	13	13	13	13	13	13	TBD
Column	10	9	8	8	11	10	9	8	11	10	9	8	12	11	10	TBD

The data port size is programmable, but the following examples use all 32 bits of the local bus. The 32-bit port size requires four SDRAM devices (with 8-bit I/O ports) connected in parallel to a single chip select. If 128-Mbit devices are used, one chip select provides 128-Mbit/device×4 devices = 64 Mbytes. If 4 chip selects are programmed for SDRAM use, the result is 64 Mbytes×4 = 256 Mbyte. If 256-Mbit SDRAM devices are used, the total available memory is 512 Mbyte. Consequently 512-Mbit devices allow for 1 Gbyte.

Although there is no technical difficulty in supporting multiple chip select configurations, in practice, the user may want to maximize the amount of SDRAM assigned to each chip select to minimize cost.

10.5.4.3 SDRAM Machine Limitations

This section describes limitations of the local bus SDRAM machine.

10.5.4.3.1 Analysis of Maximum Row Number Due to Bank Select Multiplexing

LSDMR[BSMA] is used to multiplex the bank select address. The BSMA field and corresponding multiplexed address are shown below:

```
000 LA12–LA13
001 LA13–LA14
...
111 LA19–LA20
```

Note that LA12 is the latched value of LAD12.

The highest address signals that the bank selects can be multiplexed with are LA[12:13], which limits the signals for the row address to LA[14:31]. For a 32-bit port, the maximum width of the local bus, LA[30:31] are not connected, and the maximum row is LA[14:29]. The local bus SDRAM machine supports 15 rows, which is sufficient for all devices.

10.5.4.3.2 Bank Select Signals

Page-based interleaving allows bank signals to be multiplexed to the higher-order address signals to leave room for future upgrades. For example, a user could multiplex the bank select signals to LA[14:15], leaving LA16 to connect to the address signal for a larger memory size.

This allows the system designer to design one board that can be used with a current generation of SDRAM devices and upgraded to the next generation without requiring a new board layout.

10.5.4.3.3 128-Mbyte SDRAM

Figure 10-75 shows the connection to an SDRAM of 128 Mbytes. Note that all circuit diagrams are principal connection diagrams and do not show any means of signal integrity.

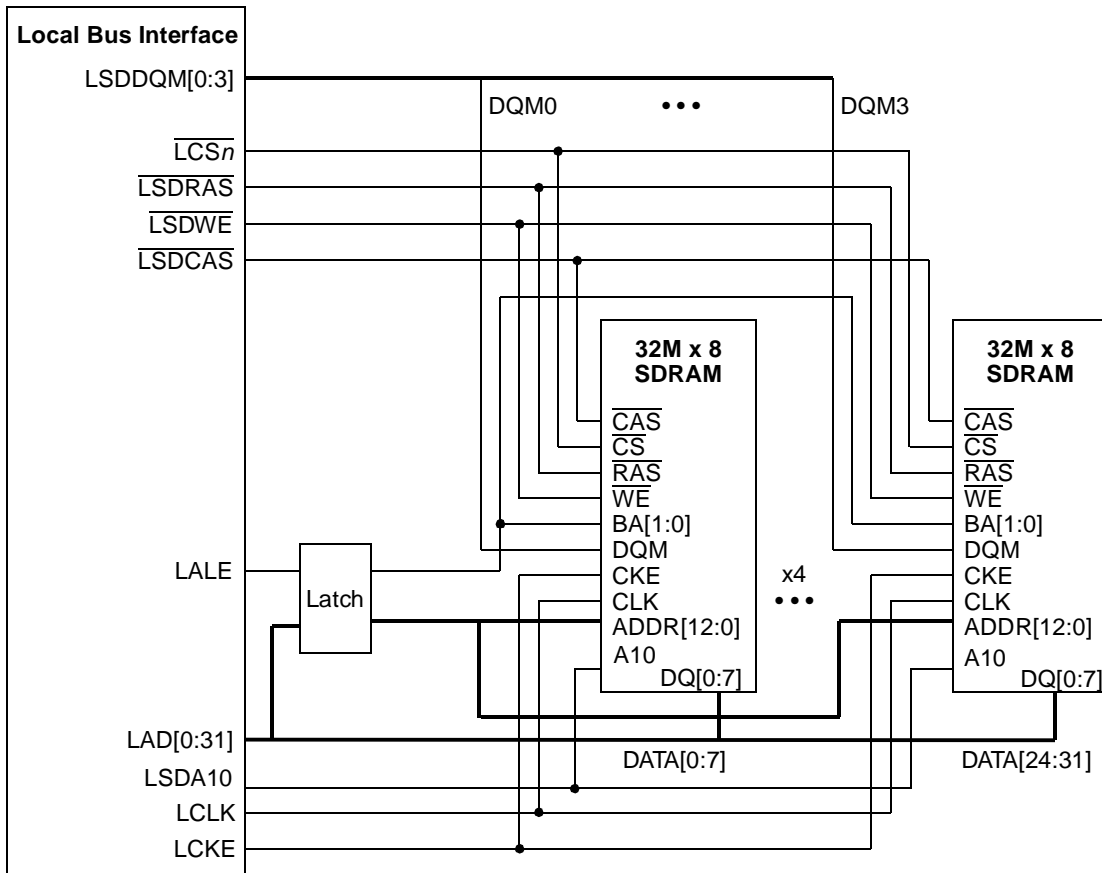


Figure 10-75. 128-Mbyte SDRAM Diagram

Table 10-35 shows details about LAD n signal connections for the example in Figure 10-75.

Table 10-35. LAD n Signal Connections to 128-Mbyte SDRAM

LAD (Latch Address)	SDRAM Address Signal
LAD29	A0
LAD28	A1
LAD27	A2
LAD26	A3
LAD25	A4
LAD24	A5
LAD23	A6
LAD22	A7

Table 10-35. LAD_n Signal Connections to 128-Mbyte SDRAM (continued)

LAD (Latch Address)	SDRAM Address Signal
LAD21	A8
LAD20	A9
LAD19 (no connect)	A10 is connected to LSDA10
LAD18	A11
LAD17	A12
LAD16	BA0 (if LSDMR[BSMA] = 011)
LAD15	BA1 (if LSDMR[BSMA] = 011)

Consider the following SDRAM organization:

- The 32-bit port size is organized as 8 x 8 x 32 Mbit.
- Each device has four internal banks, 13 row address lines, and 10 column address lines.

The logical address is partitioned as shown in [Table 10-36](#).

Table 10-36. Logical Address Bus Partitioning

A[0:4]	A[5:17]	A[18:19]	A[20:29]	A[30:31]
msb of start address	Row	Bank select	Column	lsb

The following parameters are extracted:

- COLS = 011, 10 column lines
- ROWS = 100, 13 row lines

During the address phase, the SDRAM address port is set as shown in [Table 10-37](#).

Table 10-37. SDRAM Device Address Port During Address Phase

LA[0:14]	LA[15:16]	LA[17:29]	LA[30:31]
—	Internal bank select A[18:19]	Row A[5:17]	no-connect

Because the internal bank selects are multiplexed over LA[15:16], LSDMR[BSMA] must be set to 011.

[Table 10-38](#) shows the address port configuration during a READ/WRITE command.

Table 10-38. SDRAM Device Address Port During READ/WRITE Command

LA[0:14]	LA[15:16]	LA[17:18]	LA[19]	LA[20:29]	LA[30:31]
msb of start address	Internal bank select	Don't care	AP	Column	no-connect

Table 10-39 shows the register configuration for this example. PSRT and MPTPR are not shown but should be programmed according to the device’s specific refresh requirements.

Table 10-39. Register Settings for 128-Mbytes SDRAMs

Register	Field	Value
BR n	BA	Base address
	PS	11 = 32-bit port size
	MS	011 = SDRAM-local bus
	V	1
OR n	AM	11_1111_1000_0000_0000_0
	COLS	011
	ROWS	100
LSDMR	RFEN	1
	OP	000
	BSMA	011
	RFRC	From device data sheet
	PRETOACT	From device data sheet
	ACTTOROW	From device data sheet
	BL	0
	WRC	From device data sheet
	BUFCMD	0
	CL	From device data sheet

10.5.4.3.4 256-Mbyte SDRAM

This example uses the same SDRAM as the previous example, but doubles the number of devices connected and therefore uses two chip selects.

10.5.4.3.5 512-Mbyte SDRAM

This example uses the MT48LC64M4A2FB from Micron to implement 512 Mbytes.

In this SDRAM organization:

- The 32-bit port size is 8 x 4 x 64 Mbit x 2 chip select lines.
- Each device has 4 internal banks, 13 row address lines, and 11 column address lines.

The logical address is partitioned as shown in Table 10-40.

Table 10-40. Logical Address Partitioning

A[0:3]	A[4:16]	A[17:18]	A[19:29]	A[30:31]
msb of start address	Row	Bank select	Column	lsb

The following parameters can be extracted:

- COLS = 100, 11 column lines
- ROWS = 100, 13 row lines

During the address phase, the SDRAM address port is set as in [Table 10-41](#).

Table 10-41. SDRAM Device Address Port During Address Phase

LA[0:13]	LA[15:16]	LA[17:29]	LA[30:31]
—	Internal bank select (A[17:18])	Row (A[4:16])	no-connect

Because the internal bank selects are multiplexed over LA[15:16], LSDMR[BSMA] must be set to 011.

[Table 10-42](#) shows the address port settings during a READ/WRITE command.

Table 10-42. SDRAM Device Address Port During READ/WRITE Command

LA[0:14]	LA[15:16]	LA[17]	LA[18]	LA[19:29]	LA[30:31]
msb of start address	Internal bank select	Don't care	AP	Column	no-connect

[Table 10-43](#) shows the register configuration. PSRT and MPTPR are not shown, but they should be programmed according to the specific device's refresh requirements.

Table 10-43. Register Settings for 512-Mbyte SDRAMs

Register	Field	Value
BR _n	BA	Base address
	PS	11 = 32-bit port size
	MS	011 = SDRAM-local bus
	V	1
OR _n	AM	11_1110_0000_0000_0000_0
	BPD	01
	COLS	100
	ROWS	100
PSDMR	RFEN	1
	OP	000
	BSMA	011
	RFRC	From device data sheet
	PRETOACT	From device data sheet
	ACTTOROW	From device data sheet
	BL	0
	WRC	From device data sheet
	BUFCMD	0
	CL	From device data sheet

10.5.4.4 Parity Support for SDRAM

Unlike older DRAM technologies, SDRAM devices typically are organized either x4, x8, x16 or x32. No mainstream devices include parity support. To allow for error protection on the local bus an additional SDRAM for the 4 parity bits must be used. Because the local bus allows for SDRAM accesses with less than the full port size, read-modify-write cycles are supported for SDRAM write cycles.

Figure 10-76 shows a connection diagram.

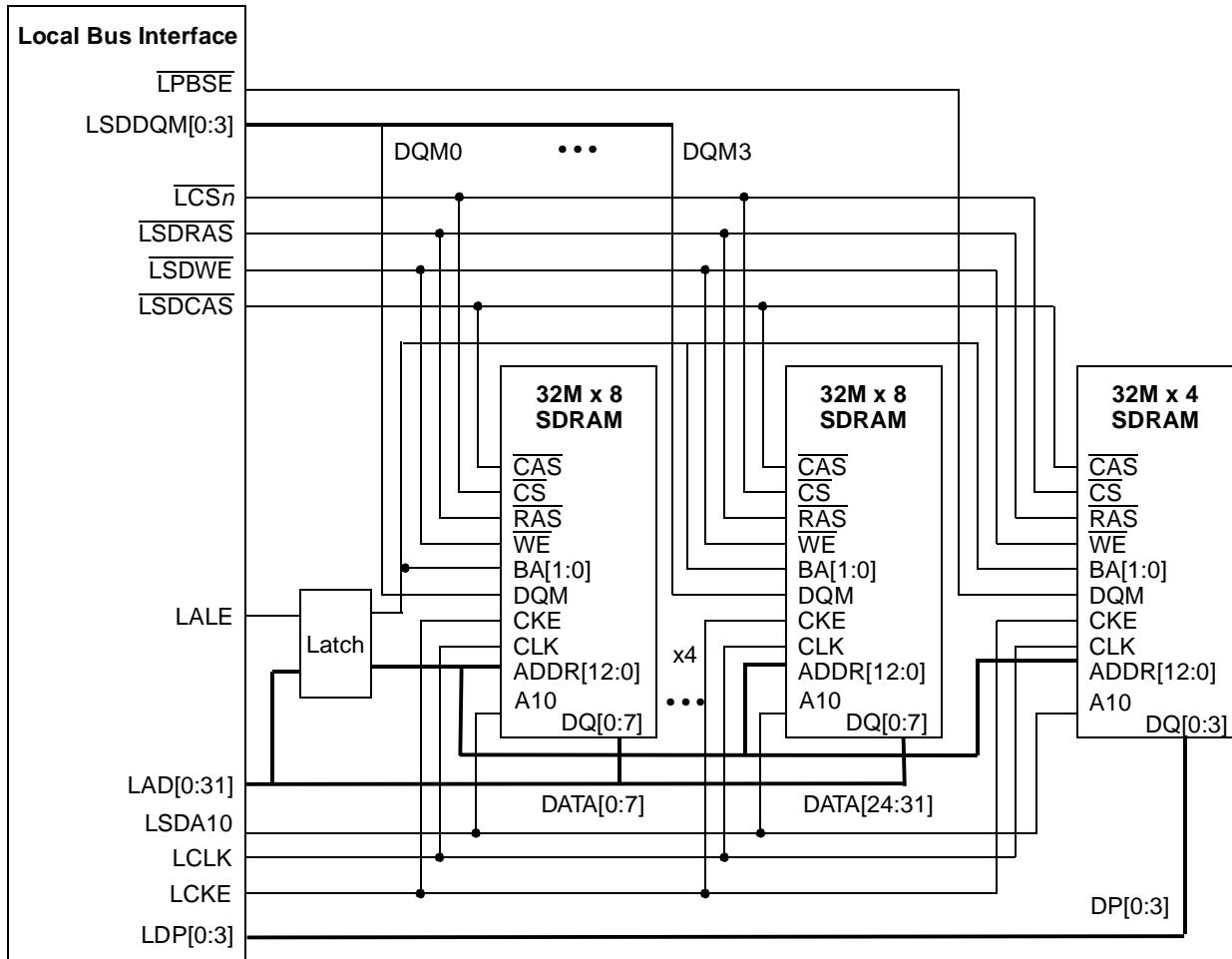


Figure 10-76. Parity Support for SDRAM

10.5.5 Interfacing to ZBT SRAM

In many applications, SDRAM provides sufficient performance for the local bus. However, especially in networking applications, memory access patterns are often random and SDRAM is not optimized for that case. ZBT SRAMs have been designed to optimize the performance in networking applications. This section describes how to interface to ZBT SRAMs. Figure 10-77 shows the connections. The UPM is used to generate control signals. The same interfacing is used for pipelined and flow-through versions of ZBT SRAMs. However different UPM patterns must be generated for those cases. Because ZBT SRAMs are used mostly by performance-critical applications, a 32-bit maximum local bus width is assumed.

ZBT SRAMs allow different configurations. For the local bus, the burst order should be set to linear burst order by tying the mode signal to GND; $\overline{\text{CKE}}$ should also be tied to ground.

ZBT SRAMs perform four-beat bursts. Because the LBC generates eight-beat transactions (for 32-bit ports) the UPM breaks down each burst into two consecutive four-beat bursts. The internal address generator of the LBC generates the new A27 for the second burst.

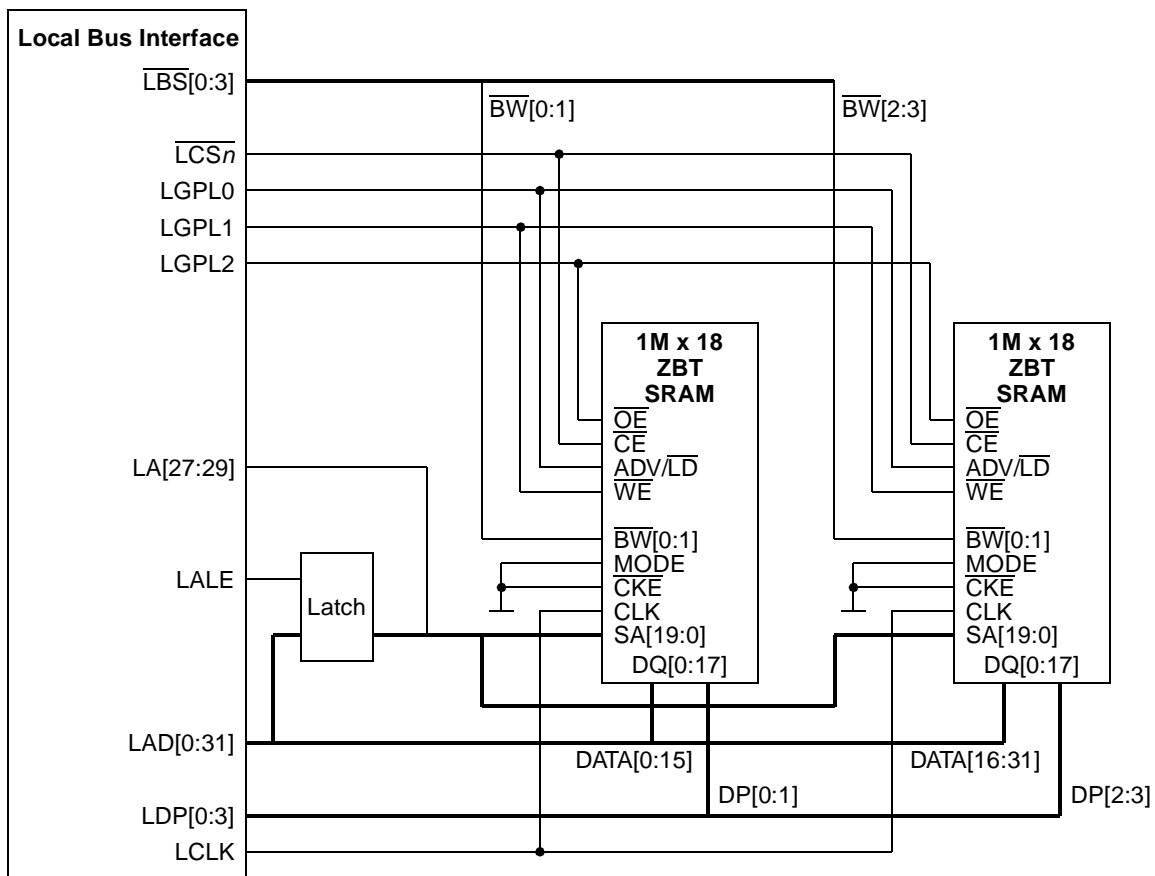


Figure 10-77. Interface to ZBT SRAM

Because linear burst on the SRAM is used, the device itself bursts with the burst addresses of [0:1:2:3]. The local bus always generates linear bursts and expects [0:1:2:3:4:5:6:7]. Therefore, two consecutive linear bursts of the ZBT SRAM with A27 = 0 for the first burst and A27 = 1 for the second burst give the desired burst pattern.

The UPM also supports single beat accesses. Because the ZBT SRAM does not support this and always responds with a burst, the UPM pattern must take care that data for the critical beat is provided (for write) or sampled (for read), and that the rest of the burst is ignored (by negating \overline{WE}). The UPM controller must wait for the end of the SRAM burst to avoid bus contention with further bus activities.

Note that SRAMs are available with natural parity. In the example, a $\times 18$ SRAM, which holds two data bytes and two parity bits, is used. While for the support of parity on SDRAM banks, the local bus must use read-modify-write cycles and compromise performance, SRAM banks can be used with natural parity and do not compromise performance for parity support.

10.5.5.1 Interfacing to MSC8122 DSI

The MSC8122 direct-slave interface (DSI) gives an external host direct access to the MSC8122. It provides the following slave interfaces to an external host:

- Asynchronous SRAM-like interface giving the host single accesses (with no external clock).
- Synchronous SSRAM-like interface giving the host single or burst accesses of 256 bits (eight beats of 32 bits or four beats of 64 bits) with its external clock decoupled from the MSC8122 internal bus clock.

The DSI supports 32- or 64-bit data bus modes. For connection to the local bus, the DSI must be configured in 32-bit mode. As part of the DSP reset.

The DSI supports two addressing modes, which are determined during the MSC8122 boot sequence and are used in both 32- and 64-bit data modes. Refer to details in the MSC8122 documentation.

- Full address bus mode with HA[11–29]
- Sliding window mode with HA[14–29]

10.5.5.2 DSI in Asynchronous SRAM-Like Mode

The local bus supports the DSI single- and double-strobe of operation. The dual strobe configuration in [Figure 10-78](#) shows the interface to the MSC8122 DSI for asynchronous mode.

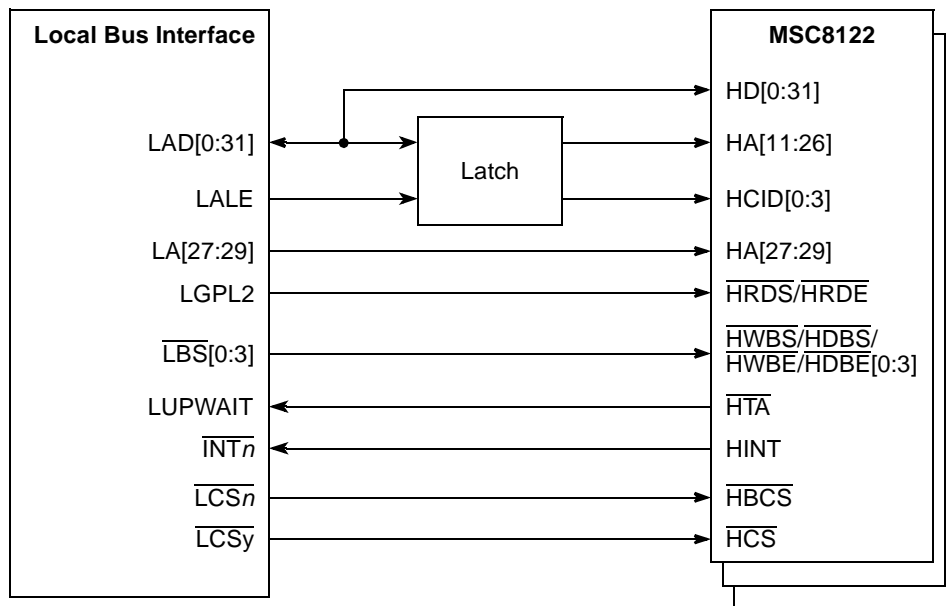


Figure 10-78. Interface to MSC8122 DSI in Asynchronous Mode

The asynchronous SRAM-like mode of the DSI is inherently slower than the synchronous mode and should be used if only relatively small amounts of data are transferred between the communications controller and the MSC8122. For maximum timing flexibility, the UPM machine of the LBC should be used.

The UPM programmer is responsible for ensuring correct setup and hold timings for all signals. The UPM allows sufficient control to satisfy any requirements here.

[Figure 10-79](#) shows an asynchronous write access. The DSI samples the host chip ID signals (HCID[0:3]) on the first falling edge of the host write byte strobe signals (\overline{HWBS}) on which the host chip select signal (\overline{HCS}) is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. By asserting \overline{HTA} , the DSI signals whether it is ready to sample the host data bus (HD), and the host can terminate the access by immediately negating \overline{HWBS} . The WAEN feature of the UPM must be used to insert wait states while the DSI is busy. $MnMR[UWPL]$ must be cleared to interpret the correct polarity of \overline{HTA} . The DSI samples the host address bus (HA) and the host data bus (HD) on the rising edge of \overline{HWBS} . In addition, the assertion of $\overline{HWBS}[0:3]$ are sampled at the end and are part of the access attributes.

Because the UPM is used for this mode, the DCR[HTAAD] should be set 1 and DCR[HTADT] should be defined to a non zero value. This mode is to be used in implementations with a pull-up resistor on \overline{HTA} . The host can start its next access (back-to-back accesses) without negating \overline{HCS} between accesses. If the next access is not to the same MSC8122, to prevent contention on \overline{HTA} , the host must wait until the previous DSI stops driving \overline{HTA} before it accesses the next device. If the next access is to the same MSC8122, the host must not start consecutive accesses before \overline{HTA} is asserted by the previous access. The

easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ is inactive.

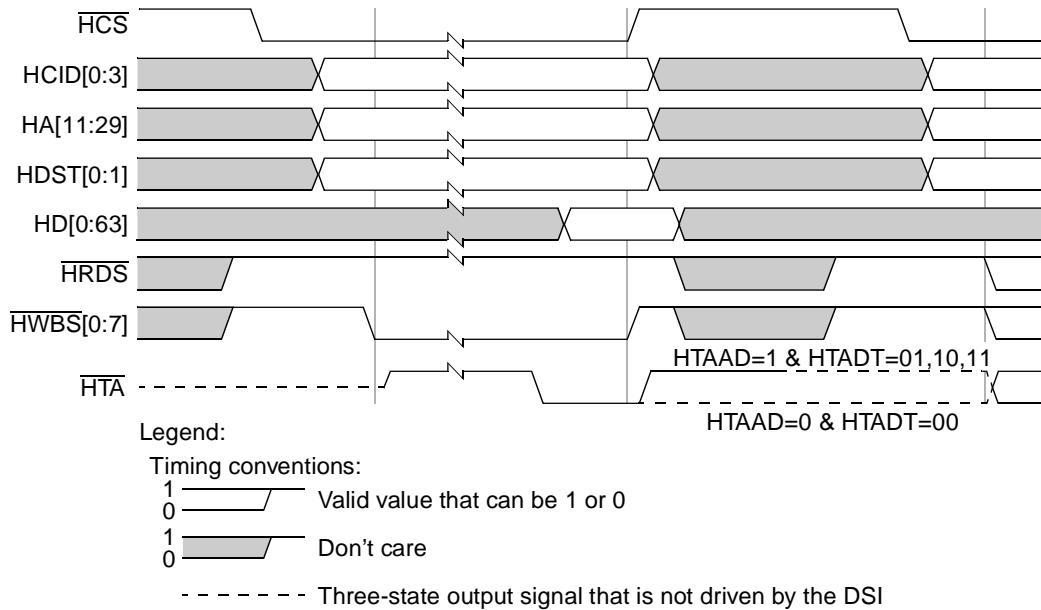


Figure 10-79. Asynchronous Write to MSC8122 DSI

Figure 10-80 shows an asynchronous read access. The DSI samples the host address bus (HA) and the HCID on the first falling edge of the host read strobe signal ($\overline{\text{HRDS}}$) on which the $\overline{\text{HCS}}$ is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. When the DCR[RPE] bit is set, read access to the memory space (not to the register space) initiates data prefetching from consecutive addresses in the internal memory space. By asserting $\overline{\text{HTA}}$, the DSI signals that the data is valid and that the host can sample the host data bus (HD) and terminate the access by negating $\overline{\text{HRDS}}$. If the data for this access is already in the read buffer due to the prefetch mechanism, assertion time of $\overline{\text{HTA}}$ is improved. The WAEN feature of the UPM must be used to insert wait states while the DSI is busy. $MnMR[\text{UWPL}]$ has to be cleared to interpret the correct polarity of the $\overline{\text{HTA}}$ signal.

Because the UPM is used in the mode, the DCR[HTAAD] should be set 1 and the drive time control field, DCR[HTADT], should be defined to a non zero value. This mode is specially designed to be used for implementations with a pull-up resistor on $\overline{\text{HTA}}$.

The host can start its next access (back-to-back accesses) without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8122, then to prevent contention on $\overline{\text{HTA}}$, the host must wait until the previous DSI stops driving $\overline{\text{HTA}}$ before it accesses the next device. If the next access is to the same MSC8122, the host must not start consecutive access before $\overline{\text{HTA}}$ is actively driven to 1 by the previous access. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ is inactive.

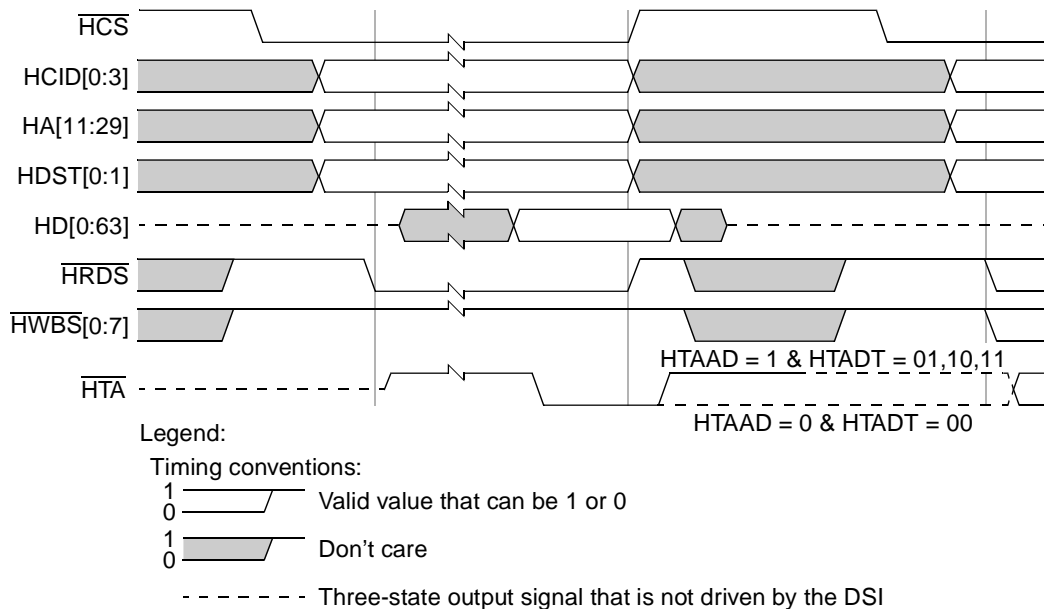


Figure 10-80. Asynchronous Read from MSC8122 DSI

10.5.5.3 DSI in Synchronous Mode

The synchronous SSRAM-like mode of the DSI is inherently faster than the asynchronous mode and should be used if larger amounts of data are transferred between the communications controller and the MSC8122. This will optimize the bus utilization, especially if several MSC8122s are connected to one local bus. The UPM machine of the LBC must be used to implement this interface.

Figure 10-81 shows the interface for synchronous mode. Because the DSI asserts and negates \overline{HTA} in synchronous mode even within a burst transfer on a clock-by-clock basis, and because the DSI expects the host to react within one clock cycle, some tricks can be implemented to support the synchronous mode.

\overline{HTA} drives LUPWAIT of the UPM. $MnMR[UWPL]$ must be cleared to interpret the correct polarity of \overline{HTA} . Because this signal influences the internal state machine of the local bus clock, the local bus cannot react to \overline{HTA} changes correctly within one local bus clock. Refer to Section 10.4.4.4.10, “Wait Mechanism (WAEN).”

The solution to this lies in that the local bus operates at a higher frequency than the DSI interface of the DSP. The local bus clock can be divided by an integer divider (1:2, 1:3 or 1:4) to generate the DSI clock. This should not be a problem because the local bus is designed for much higher frequencies than the DSI. Because all timings are given in DSP DSI clock cycles, the UPM patterns must be adjusted appropriately and need to assert a signal for 2, 3, or 4 clocks (as many as the divider ratio) instead of one. Fortunately, the UPM has the REDO feature, which allows every UPM RAM entry to be executed 1x, 2x, 3x, or 4x, which should be sufficient for any likely divider ratio.

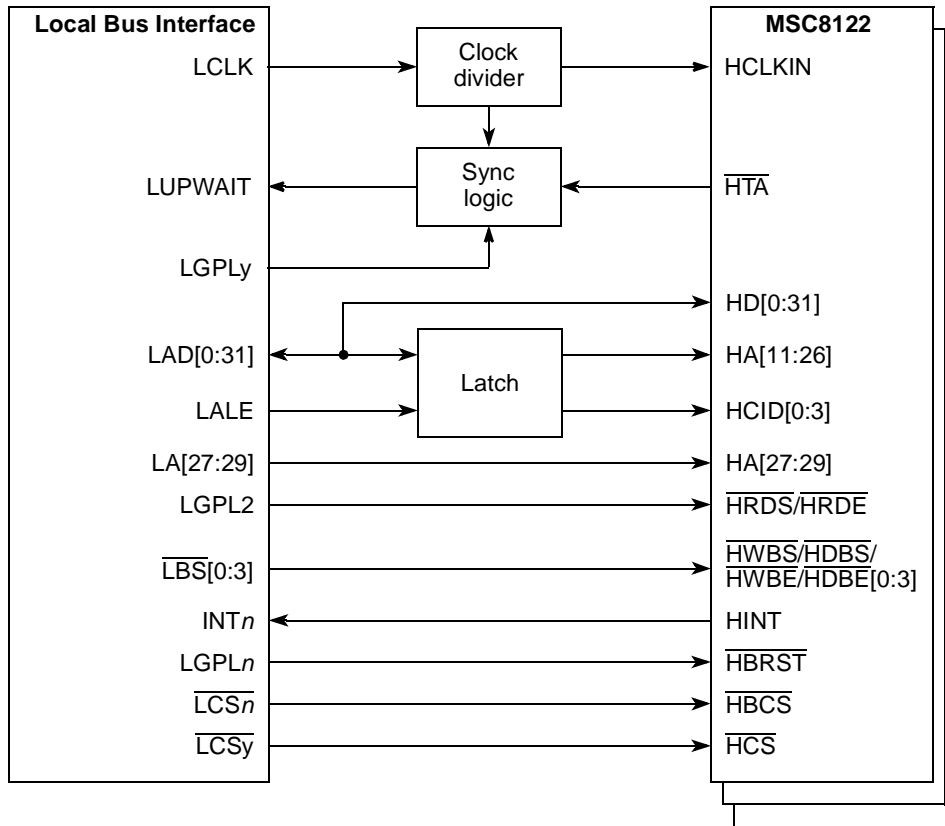


Figure 10-81. Interface to MSC8122 DSI in Synchronous Mode

This solution allows the local bus to react within multiple local bus clocks to the \overline{HTA} signal and be still within one DSI clock. Typically, LUPWAIT is synchronized internal, and only 2 clocks after LUPWAIT changes, new data can be sampled or presented. For example, if the local bus clock ratio is 3x the DSI clock, data can be sampled in the third local bus sub-clock, which is the last third of the DSI clock. If the local bus clock ratio is only 2x the DSI clock, there is a special mode, where the LUPWAIT is not synchronized. Refer to [Section 10.4.4.5, “Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge.”](#) In this mode, data is sampled in the 2nd subclock, which is the second half of the DSI clock. AC timing of LUPWAIT must be met in this mode; otherwise behavior is indeterminate.

The remaining issue is the synchronization of the UPM cycles to the beginning of the DSI clock cycle. Because the UPM executes n cycles for every DSI cycle, UPM transitions must be synchronized to the DSI clock. The solution is to use a special synchronize cycle at the beginning of the pattern. A GPL signal controls a multiplexer and activates external synchronization logic, which uses the DSI clock to stall the UPM by asserting LUPWAIT until the beginning of the next DSI cycle. After that, this GPL signal must be negated and the multiplexer connects LUPWAIT to \overline{HTA} instead, for the rest of the bus cycle. Note that the GPL signals should be used in the inverted state of their inactive state (LGPLn are 1 when inactive) to start the synchronization process.

[Figure 10-82](#) shows such a synchronization mechanism for a clock divider of 3. Note that the length of the synchronization cycle depends on the relative start of the synchronization process and varies with every access. It can vary from one to n (clock ratio) local bus clocks.

The second column (compensation cycle) is intended to compensate for the reaction time of LUPWAIT to get in lock step with the DSI clock. For example, if the clock divider ratio is 1:3 and the LUPWAIT reaction time is two local bus clocks, because LUPWAIT is synchronized, then one local bus clock should be inserted.

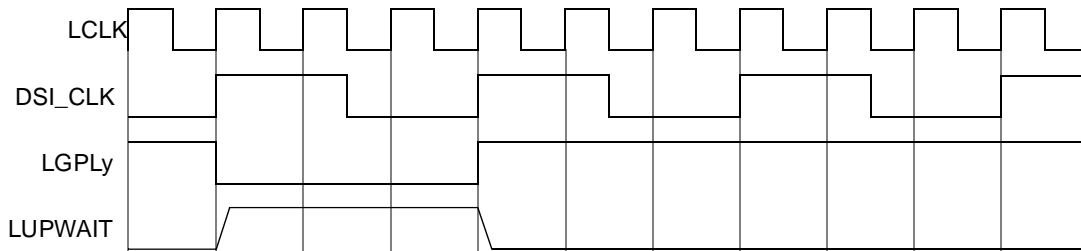


Figure 10-82. UPM Synchronization Cycle

Table 10-44. UPM Synchronization Cycles

	Synchronization Cycle	Compensation Cycle	DSI Cycle 1	Bits
cst1–cst4				0–3
bst1–bst4				4–7
g0xx				8–11
g1tx				12–13
g2tx				14–15
g3tx				16–17
g4t1				18
g4t3	1	0		19
g5tx				20–21
redo[0]	0	0	1	22
redo[1]	0	0	0	23
loop	0	0		24
exen	0			25
amx0	0	0		26
amx1	0	0		27
na	0			28
uta	0			29
todt	0			30
last	0			31

This section describes synchronous single write and read, and synchronous burst write and read operations. The local bus supports the DSI single strobe as well as the DSI double strobes of operation. The dual strobe configuration is shown as an example.

10.5.5.3.1 Synchronous Single Write

Figure 10-83 shows a synchronous single write access.

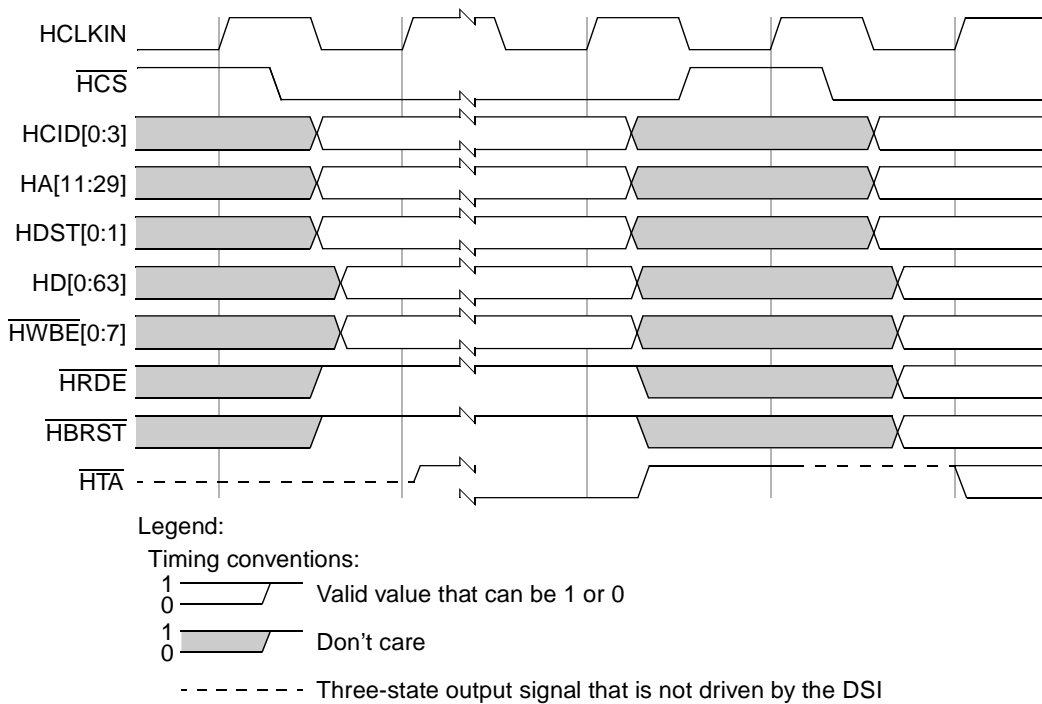


Figure 10-83. Synchronous Single Write to MSC8122 DSI

The DSI samples HA, HDST, HCID, HD, HWBE, HRDE, and HBRST on the first HCLKIN rising edge on which HCS is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. At least one HWBE signal is asserted, and HRDE and HBRST are negated. Assertion of HTA indicates that the DSI is ready to complete the current access and the host must terminate this access. Because HTA is connected to the LUPWAIT signal of the UPM, all local bus signals are frozen until HTA goes to 0 and then the UPM continues in its pattern. Typically, HTA is asserted immediately. If the write buffer is full, HTA assertion is delayed. HTA is asserted for one HCLKIN cycle, driven to logic 1 in the next cycle, and stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8122 immediately on the next HCLKIN rising edge without negating HCS between accesses. If the next access is not to the same MSC8122, then, to prevent contention on HTA, the host must wait to access the next device until the previous DSI stops driving HTA. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that HTA is inactive.

10.5.5.3.2 Synchronous Single Read

Figure 10-84 shows a synchronous single read access.

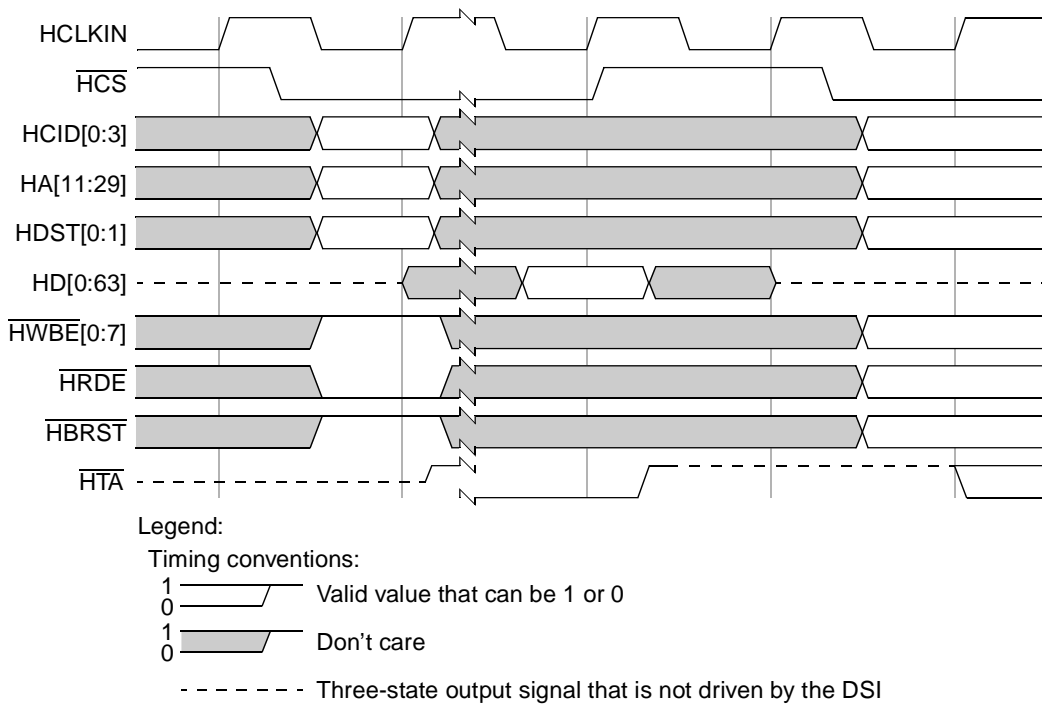


Figure 10-84. Synchronous Single Read from MSC8122 DSI

The DSI samples HA, HDST, HCID, \overline{HWBE} , \overline{HRDE} , and \overline{HBRST} on the first HCLKIN rising edge on which \overline{HCS} is asserted. If the HCID[0:3] signals match the CHIPID value, the DSI is accessed. \overline{HRDE} is asserted, and \overline{HWBE} and \overline{HBRST} are negated. If DCR[RPE] is set (see MSC8122 documentation), read access to the memory space (not to the register space) initiates prefetching data from consecutive addresses in the internal memory space. Assertion of \overline{HTA} indicates that data is valid and the host must sample the HD and terminate the access. Because \overline{HTA} is connected to the UPM LUPWAIT signal, all local bus signals are frozen until \overline{HTA} goes to 0; then the UPM continues in its pattern. \overline{HTA} is asserted earlier when the data for this access is already prefetched to the read buffer. It asserted for one HCLKIN cycle and driven to logic 1 in the next cycle. It stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8122 immediately in the next HCLKIN rising edge without negating \overline{HCS} between accesses. If the next access is not to the same MSC8122, then, to prevent contention on \overline{HTA} , the host must wait to access the next device until the previous DSI stops driving \overline{HTA} . The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that \overline{HTA} is inactive.

10.5.5.3.3 Synchronous Burst Write

Figure 10-85 shows a synchronous burst write access.

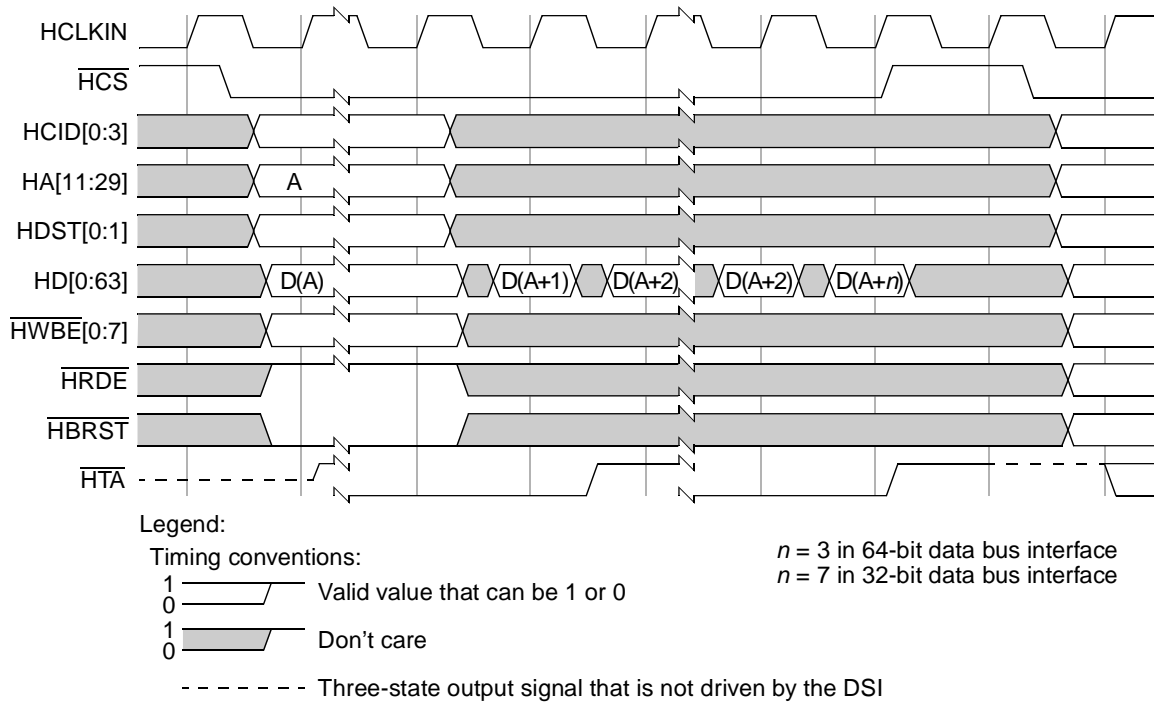


Figure 10-85. Synchronous Burst Write to MSC8122 DSI

The DSI samples HA, HDST, HCID, HD, HWBE, HRDE, and HBRST on the first HCLKIN rising edge on which HCS is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. HWBE are asserted, HBRST is asserted, and HRDE is negated. Assertion of HTA indicates that the DSI is ready to complete the current beat of the access and the host must proceed to the next beat of this access. When the host reaches the last beat of the access, it must terminate the burst access. Typically HTA is asserted immediately for each beat of the access. If the write buffer is full, HTA assertion is delayed. Because HTA is connected to the LUPWAIT signal of the UPM, all local bus signals are frozen until HTA goes to 0 and then the UPM continues in its pattern. After the last beat of the access, HTA is driven to logic 1 and stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8122 immediately in the next HCLKIN rising edge without negating HCS between accesses. If the next access is not to the same MSC8122, then, to prevent contention on HTA, the host must wait to access the next device until the previous DSI stops driving HTA. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that HTA is inactive.

10.5.5.3.4 Synchronous Burst Read

Figure 10-86 shows a synchronous burst read access. The DSI samples HA, HDST, HCID, $\overline{\text{HWBE}}$, $\overline{\text{HRDE}}$, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed.

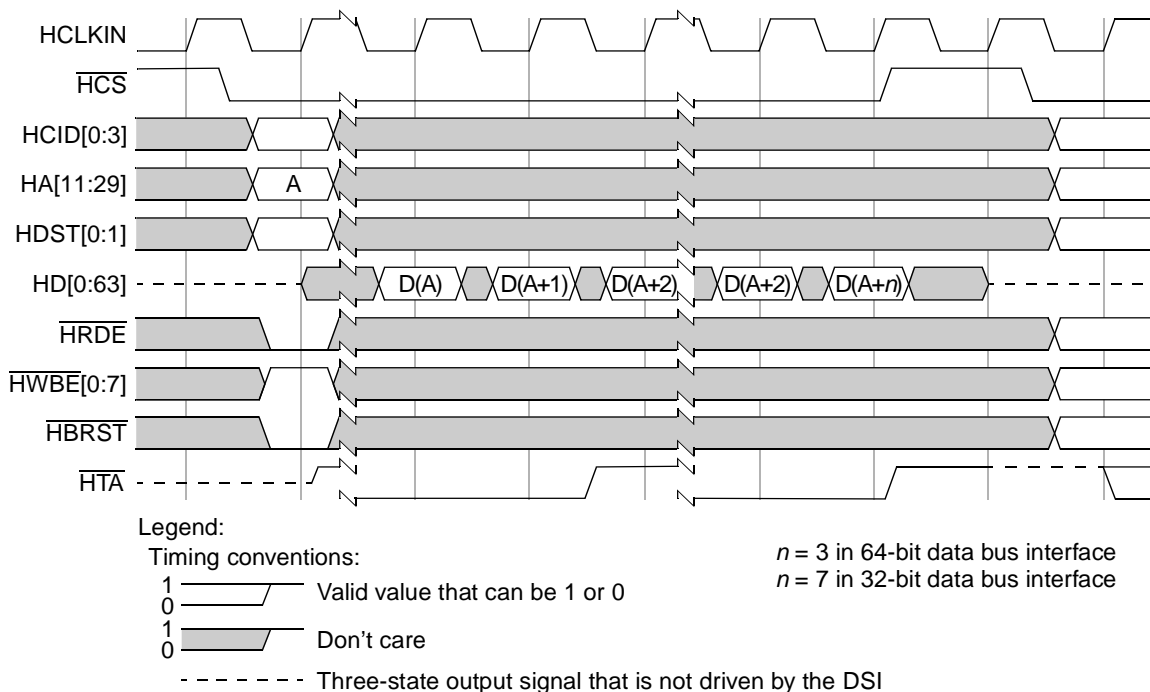


Figure 10-86. Synchronous Burst Read from MSC8122 DSI

$\overline{\text{HRDE}}$ and $\overline{\text{HBRST}}$ are asserted and $\overline{\text{HWBE}}$ are negated. When DCR[RPE] (see the MSC8122 documentation) is set, a burst read access initiates data prefetching from consecutive addresses in the internal memory space. Assertion of $\overline{\text{HTA}}$ indicates that data is valid for the current beat of the access and the host must proceed to the next beat of this access. Because $\overline{\text{HTA}}$ is connected to the LUPWAIT signal of the UPM, all local bus signals are frozen until $\overline{\text{HTA}}$ goes to 0 and then the UPM continues in its pattern. When the host reaches the last beat of the access, it must terminate the burst access. The $\overline{\text{HTA}}$ is asserted earlier when the data for this access is already prefetched to the read buffer. Typically, after the first beat of the burst access, $\overline{\text{HTA}}$ remains asserted until the end of the access. After the last beat of the access, $\overline{\text{HTA}}$ is driven to 1 and stops being driven in the next rising edge of HCLKIN. The host can start its next access to the same MSC8122 immediately in the next HCLKIN rising edge without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8122, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving $\overline{\text{HTA}}$. The easiest way to achieve this is insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ is inactive.

10.5.5.4 Broadcast Accesses

By using $\overline{\text{HBCS}}$, a host can share one chip-select signal between multiple MSC8122 devices for broadcasting write accesses. In broadcast mode, the DSI does not drive its $\overline{\text{HTA}}$ signal to prevent contention between multiple devices driving different values to the same signal. Also, the DSI does not decode HCID[0:3].

Note that broadcasting is allowed only for write accesses.

The DSI sets the overflow bit, DER[OVF], if there is an overflow during broadcast accesses. This bit can be cleared by writing a value of 1 to it. To avoid overflow when accessing DSI registers during broadcast accesses, wait at least 10 host clock cycles in synchronous mode or 8 internal clock cycles in asynchronous mode between each DSI register access.

To avoid data corruption, if DER[OVF] is set, no broadcast access is written until the bit is reset. Therefore, after the last broadcast access, and before any regular write access, DER[OVF] must first be read and reset if it is set.

NOTE

In asynchronous mode, write data from a previous access (even a normal write access) may be lost due to overflow during broadcast accesses. To prevent such a loss, ensure that previous access data has propagated to the FIFO or DSI registers, depending on the type of previous access. This can be achieved by performing a read access before the first broadcast access.

In broadcast accesses, the host must comply with the following rules:

- In asynchronous mode, $\overline{HWBS}[0:3]/\overline{HDBS}[0:3]$ assertion time should be at least the minimum, which is defined in the AC characteristics section of the *MSC8122 Technical Data* sheet.
- In synchronous mode single access, the host must wait 1 cycle before terminating the access. Access signals must be in the same valid state during two positive edges of the host clock cycles. Access duration is two clock cycles (the DSI may translate accesses lasting longer than two clock cycles as two or more back-to-back accesses).
- In synchronous mode burst accesses, broadcast accesses are not allowed.



Chapter 11 Sequencer

11.1 Sequencer Overview

The I/O sequencer switches transactions among its ports, using a buffer pool to minimize blocking. [Figure 11-1](#) is a block diagram of the I/O sequencer (IOS).

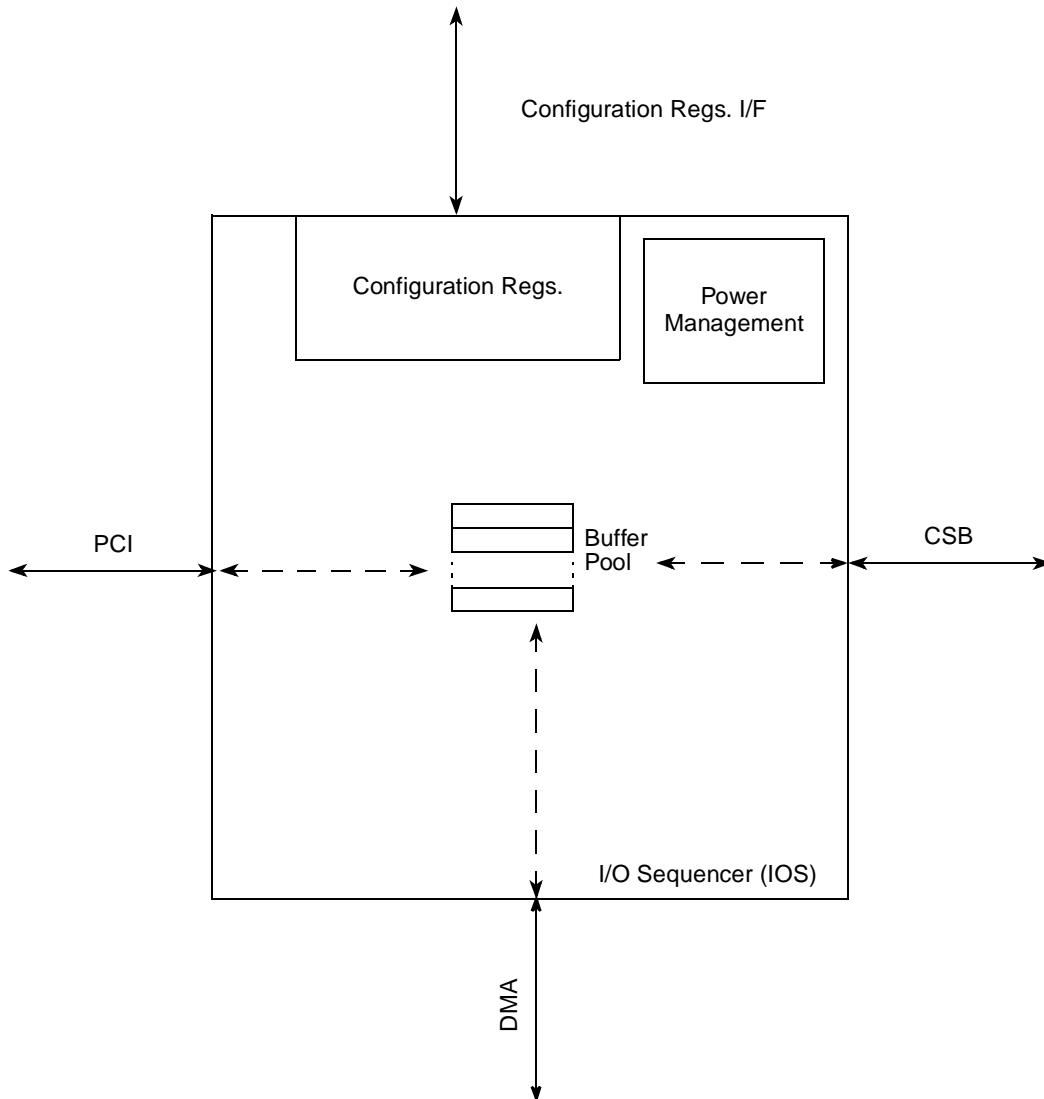


Figure 11-1. I/O Sequencer Block Diagram

11.1.1 Sequencer Features

The I/O sequencer includes the following features:

- Switches transactions among its ports
- Contains 8 cache-line (32-byte) buffers to allow streaming of PCI transactions
- Performs address translation on outbound PCI transactions

Note that the number of CSB masters allowed to access to PCI outbound windows is restricted to no more than four non-CPU masters or no more than two non-CPU plus the CPU. If this number is exceeded, deadlock can occur on CSB arbitration.

11.2 Sequencer External Signal Description

The I/O sequencer has no external signals.

11.3 Sequencer Memory Map/Register Definition

Table 11-1 shows the I/O sequencer memory map.

Table 11-1. Sequencer Memory Map

Offset	Register	Access	Reset	Section/Page
0x00	POTAR0—PCI outbound translation address register 0	R/W	All zeros	11.4.1/11-3
0x08	POBAR0—PCI outbound base address register 0	R/W	All zeros	11.4.2/11-3
0x10	POCMR0—PCI outbound comparison mask register 0	R/W	All zeros	11.4.3/11-4
0x18	POTAR1—PCI outbound translation address register 1	R/W	All zeros	11.4.1/11-3
0x20	POBAR1—PCI outbound base address register 1	R/W	All zeros	11.4.2/11-3
0x28	POCMR1—PCI outbound comparison mask register 1	R/W	All zeros	11.4.3/11-4
0x30	POTAR2—PCI outbound translation address register 2	R/W	All zeros	11.4.1/11-3
0x38	POBAR2—PCI outbound base address register 2	R/W	All zeros	11.4.2/11-3
0x40	POCMR2—PCI outbound comparison mask register 2	R/W	All zeros	11.4.3/11-4
0x48	POTAR3—PCI outbound translation address register 3	R/W	All zeros	11.4.1/11-3
0x50	POBAR3—PCI outbound base address register 3	R/W	All zeros	11.4.2/11-3
0x58	POCMR3—PCI outbound comparison mask register 3	R/W	All zeros	11.4.3/11-4
0x60	POTAR4—PCI outbound translation address register 4	R/W	All zeros	11.4.1/11-3
0x68	POBAR4—PCI outbound base address register 4	R/W	All zeros	11.4.2/11-3
0x70	POCMR4—PCI outbound comparison mask register 4	R/W	All zeros	11.4.3/11-4
0x78	POTAR5—PCI outbound translation address register 5	R/W	All zeros	11.4.1/11-3
0x80	POBAR5—PCI outbound base address register 5	R/W	All zeros	11.4.2/11-3
0x88	POCMR5—PCI outbound comparison mask register 5	R/W	All zeros	11.4.3/11-4

Table 11-1. Sequencer Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xF0	PMCR—Power management control register	R/W	All zeros	11.4.4/11-5
0xF8	DTCR—Discard timer control register	R/W	All zeros	11.4.5/11-6

11.4 Sequencer Register Descriptions

11.4.1 PCI Outbound Translation Address Registers (POTAR_n)

The PCI outbound translation address register defines the location of the outbound translation window in the PCI (translated) address space. [Figure 11-2](#) shows the POTAR_n register fields.


Figure 11-2. PCI Outbound Translation Address Registers (POTAR_n)

[Table 11-2](#) describes POTAR_n fields.

Table 11-2. POTAR_n Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	TA	Translation address. Contains the starting address of the outbound translated address. It also corresponds to the most-significant 20 bits of a 32-bit address. The translation address must be aligned based on the window's size.

11.4.2 PCI Outbound Base Address Registers (POBAR_n)

The PCI outbound base address register (POBAR_n) defines the location of the outbound translation window in the local (source) memory space. [Figure 11-3](#) shows the POBAR_n register fields.

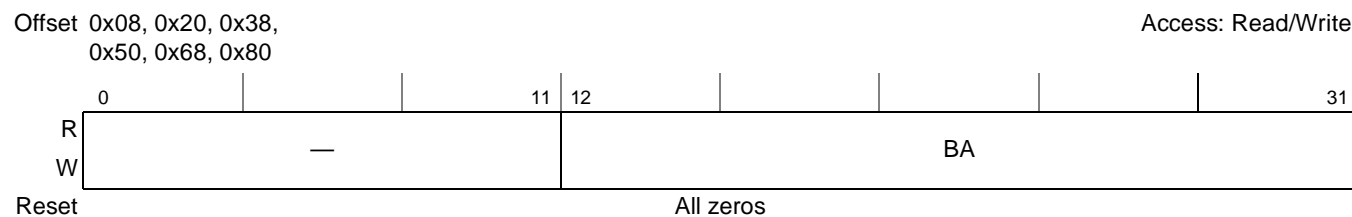

Figure 11-3. PCI Outbound Base Address Registers (POBAR_n)

Table 11-3 describes POBAR_n fields.

Table 11-3. POBAR_n Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	BA	Base address. This field contains the starting address of the outbound translated window. This field corresponds to the most-significant 20 bits of a 32-bit address.

11.4.3 PCI Outbound Comparison Mask Registers (POCMR_n)

The PCI outbound comparison mask register (POCMR_n) defines the size and destination of the outbound translation window. It also defines some properties of the window in the PCI address space. See Section 11.5.1, “Transaction Forwarding,” for more information. Figure 11-4 shows the POCMR_n register fields.

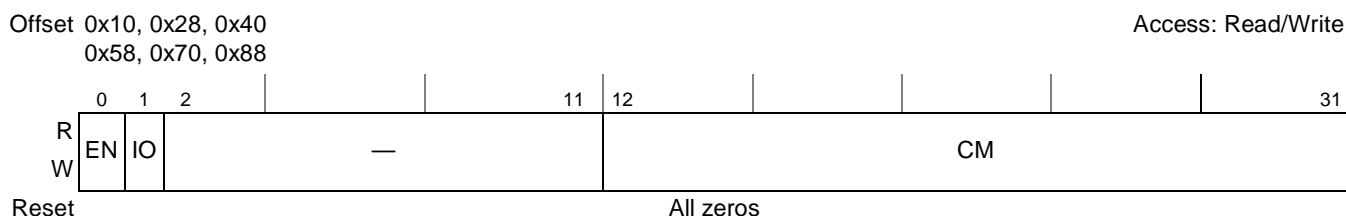


Figure 11-4. PCI Outbound Comparison Mask Registers (POCMR_n)

Table 11-4 describes the bit settings of the POCMR_n register.

Table 11-4. POCMR_n Field Descriptions

Bits	Name	Description
0	EN	Enable. Enables the address translation window. 0 Address translation is disabled for this window. 1 Address translation is enabled for this window. Local addresses that match the definition of the window will be recognized by the device and translated to the PCI memory space.
1	IO	I/O space. Determines whether the window is mapped to the PCI memory space or PCI I/O space. 0 Memory space 1 I/O space
2–11	—	Reserved, should be cleared.

11.5.1.1 Transactions from the Coherency System Bus (CSB) Port

Transactions from the CSB port are forwarded as follows:

- If the address matches the 12-byte PCI controller software configuration memory space of the PCI controller, the transaction is forwarded to the PCI port. These address values are configuration options of the I/O sequencer. See [Table 13-3](#) for more information on PCI controller software configuration memory space.
- If the address matches the DMA register memory space, the transaction is forwarded to the DMA port.
- If the address hits any of the outbound translation windows, the transaction is forwarded to the PCI port, with the address translated. See [Section 11.5.2, “PCI Outbound Address Translation,”](#) for more information.

11.5.1.2 Transactions from the PCI Port

Transactions from the PCI port are forwarded as follows:

- If the address matches the DMA register memory space, the transaction is forwarded to the DMA port.
- All other transactions are forwarded to the CSB port.

11.5.1.3 Transactions from the DMA Port

Transactions from the DMA port are forwarded as follows:

- If the address hits any of the outbound translation windows, the transaction is forwarded to the PCI port, with the address translated. See [Section 11.5.2, “PCI Outbound Address Translation,”](#) for more information.
- All other transactions are forwarded to the CSB port.

11.5.2 PCI Outbound Address Translation

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 [Section 11.4.2, “PCI Outbound Base Address Registers \(POBARn\),”](#) for more information. 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 (POTARs).

Figure 11-7 shows an example translation window for outbound memory accesses.

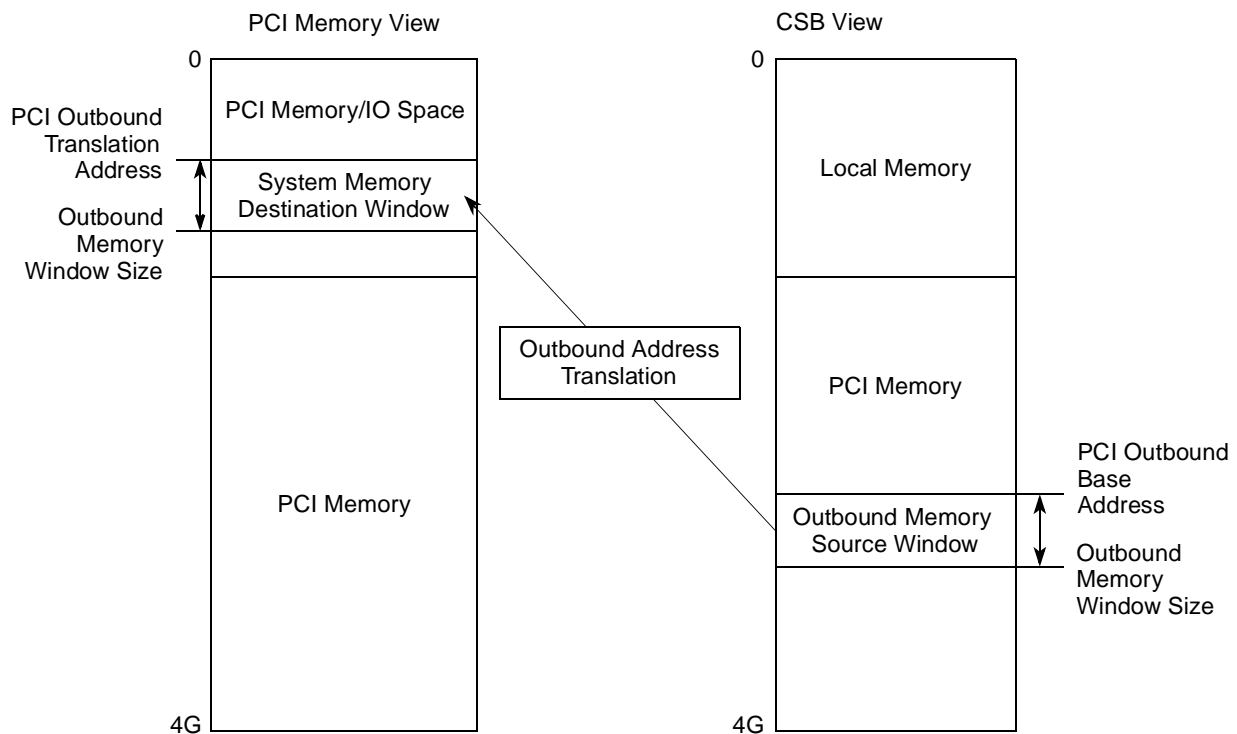


Figure 11-7. Outbound PCI Memory Address Translation

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.

11.5.3 Transaction Ordering

The following rules are applied to maintain proper ordering of transactions:

- The transactions arriving from each port are dispatched to the destination port in the order of arrival. The dispatch order of transactions arriving on different ports is not necessarily maintained.
- A read transaction that originates at the CSB port and reads from the PCI port pulls out of the IOS any posted writes that originated on the PCI port and were posted before the read data arrives from the PCI.
- The IOS can always accept a write from the PCI port without forcing the PCI port to first accept a read.

Chapter 12

DMA/Messaging Unit

The DMA/messaging unit supports communication between two processors on different buses, for example, a local processor and a processor on a PCI bus. This unit operates with generic messages and doorbell registers. [Figure 12-1](#) is a block diagram of the DMA/messaging unit.

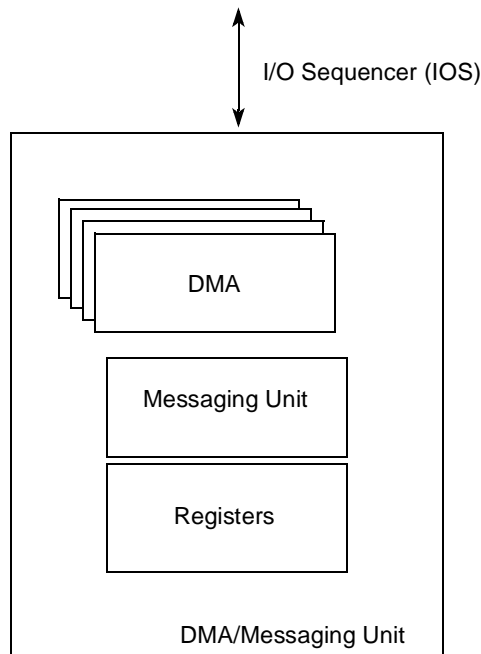


Figure 12-1. DMA/Messaging Unit Block Diagram

This block also provides a DMA controller, which transfers blocks of data independent of the local processor or PCI hosts. The DMA module has four high-speed DMA channels, which share buffer space in the I/O sequencer (IOS) to facilitate the gathering and sending of data.

12.1 DMA Features

The DMA/messaging unit includes the following features:

- Message and doorbell registers for inter-processor communication
- DMA controller
 - Four DMA channels
 - Concurrent execution across multiple channels with programmable bandwidth control
 - Misaligned transfer capability

- Data chaining and direct mode
- Interrupt on completed segment, chain, and error
- Optional external control signals (REQ/ACK/DONE) per channel

12.2 DMA External Signal Description

This section describes the DMA signals.

12.2.1 DMA Detailed Signal Descriptions

Table 12-1 contains the detailed descriptions of the DMA interface signals.

Table 12-1. DMA Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description
$\overline{\text{DREQ}}[0:3]$	I	DMA request signals, one per channel. The DMA request signal indicates the start or continuation of a DMA transfer. The falling edge of $\overline{\text{DREQ}}_n$ causes $\text{DMAMR}_n[\text{CS}]$ to be set, thereby activating the DMA channel.
		State Meaning Asserted—Assertion of $\overline{\text{DREQ}}_n$ starts or resumes a DMA transfer if $\text{DMAMR}_n[\text{EMSEN}]$ is 1. Negated—Negation of $\overline{\text{DREQ}}_n$ has no effect.
		Timing Assertion—Can be asserted asynchronously. Negation—Should remain asserted until DACK_n is asserted or the requested transaction to the peripheral occurs.
$\overline{\text{DACK}}[0:3]$	O	DMA acknowledge signals, one per channel. The DMA acknowledge signal reflects the value of $\text{DMAMR}_n[\text{CS}]$.
		State Meaning Asserted—A DMA transfer is active. Negated—The DMA transfer is halted or complete.
		Timing Assertion—Asserted asynchronously when a DMA transfer is started or resumed in the internal control logic. Negation—Negated asynchronously when a DMA transfer is halted or completed in the internal control logic. Note that there may still be outstanding write transactions in the bus pipeline after the negation of $\overline{\text{DACK}}_n$.
$\overline{\text{DDONE}}[0:3]$	O	DMA done signals, one per channel. The DMA done signal indicates that the DMA transfer has completed.
		State Meaning Asserted—A DMA transfer is complete. Negated—A DMA transfer is active or halted.
		Timing Assertion—Asserted asynchronously when a DMA transfer is completed in the internal control logic. Note that there may still be outstanding write transactions in the bus pipeline after the assertion of $\overline{\text{DDONE}}_n$. Negation—Negated asynchronously when a DMA transfer begins in the internal control logic.

12.3 DMA Memory Map/Register Definition

Table 12-2 lists the address and access of the memory map module.

Table 12-2. Module Memory Map

Offset	Register	Access	Reset	Section/Page
0x00_8030	OMISR—Outbound message interrupt status register	Mixed	All zeros	12.4.1/12-4
0x00_8034	OMIMR—Outbound message interrupt mask register	R/W	All zeros	12.4.2/12-5
0x00_8050	IMR0—Inbound message register 0	R/W	All zeros	12.4.3/12-6
0x00_8054	IMR1—Inbound message register 1	R/W	All zeros	12.4.3/12-6
0x00_8058	OMR0—Outbound message register 0	R/W	All zeros	12.4.4/12-6
0x00_805C	OMR1—Outbound message register 1	R/W	All zeros	12.4.4/12-6
0x00_8060	ODR—Outbound doorbell register	R/W	All zeros	12.4.5/12-7
0x00_8068	IDR—Inbound doorbell register	R/W	All zeros	12.4.5/12-7
0x00_8080	IMISR—Inbound message interrupt status register	Mixed	All zeros	12.4.6/12-8
0x00_8084	IMIMR—Inbound message interrupt mask register	R/W	All zeros	12.4.7/12-9
0x00_8100	DMAMR0—DMA 0 mode register	R/W	All zeros	12.4.8.1/12-10
0x00_8104	DMASR0—DMA 0 status register	R/W	All zeros	12.4.8.2/12-12
0x00_8108	DMACDAR0—DMA 0 current descriptor address register	R/W	All zeros	12.4.8.3/12-13
0x00_8110	DMASAR0—DMA 0 source address register	R/W	All zeros	12.4.8.4/12-14
0x00_8118	DMADAR0—DMA 0 destination address register	R/W	All zeros	12.4.8.5/12-14
0x00_8120	DMABCR0—DMA 0 byte count register	R/W	All zeros	12.4.8.6/12-15
0x00_8124	DMANDAR0—DMA 0 next descriptor address register	R/W	All zeros	12.4.8.7/12-15
0x00_8180	DMAMR1—DMA 1 mode register	R/W	All zeros	12.4.8.1/12-10
0x00_8184	DMASR1—DMA 1 status register	R/W	All zeros	12.4.8.2/12-12
0x00_8188	DMACDAR1—DMA 1 current descriptor address register	R/W	All zeros	12.4.8.3/12-13
0x00_8190	DMASAR1—DMA 1 source address register	R/W	All zeros	12.4.8.4/12-14
0x00_8198	DMADAR1—DMA 1 destination address register	R/W	All zeros	12.4.8.5/12-14
0x00_81A0	DMABCR1—DMA 1 byte count register	R/W	All zeros	12.4.8.6/12-15
0x00_81A4	DMANDAR1—DMA 1 next descriptor address register	R/W	All zeros	12.4.8.7/12-15
0x00_8200	DMAMR2—DMA 2 mode register	R/W	All zeros	12.4.8.1/12-10
0x00_8204	DMASR2—DMA 2 status register	R/W	All zeros	12.4.8.2/12-12
0x00_8208	DMACDAR2—DMA 2 current descriptor address register	R/W	All zeros	12.4.8.3/12-13
0x00_8210	DMASAR2—DMA 2 source address register	R/W	All zeros	12.4.8.4/12-14
0x00_8218	DMADAR2—DMA 2 destination address register	R/W	All zeros	12.4.8.5/12-14
0x00_8220	DMABCR2—DMA 2 byte count register	R/W	All zeros	12.4.8.6/12-15

Table 12-2. Module Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x00_8224	DMANDAR2—DMA 2 next descriptor address register	R/W	All zeros	12.4.8.7/12-15
0x00_8280	DMAMR3—DMA 3 mode register	R/W	All zeros	12.4.8.1/12-10
0x00_8284	DMASR3—DMA 3 status register	R/W	All zeros	12.4.8.2/12-12
0x00_8288	DMACDAR3—DMA 3 current descriptor address register	R/W	All zeros	12.4.8.3/12-13
0x00_8290	DMASAR3—DMA 3 source address register	R/W	All zeros	12.4.8.4/12-14
0x00_8298	DMADAR3—DMA 3 destination address register	R/W	All zeros	12.4.8.5/12-14
0x00_82A0	DMABCR3—DMA 3 byte count register	R/W	All zeros	12.4.8.6/12-15
0x00_82A4	DMANDAR3—DMA 3 next descriptor address register	R/W	All zeros	12.4.8.7/12-15
0x00_82A8	DMAGSR—DMA general status register	R	All zeros	12.4.8.8/12-16
0x00_82B0– 0x00_82FF	Reserved	—	—	—

12.4 DMA Register Descriptions

The following sections describe the DMA/messaging unit configuration, control, and status registers.

NOTE

The registers described in this section use little-endian byte ordering. Software running on the local processor in big-endian mode must byte-swap the data. No byte swapping occurs when the registers are accessed from the PCI bus.

12.4.1 Outbound Message Interrupt Status Register (OMISR)

OMISR contains the interrupt status of the doorbell and outbound message registers. A PCI device acknowledges the outbound message interrupt by writing a 1 to the appropriate status bit: OMISR[OM1I] or OMISR[OM0I]. Setting one of these bits clears both the interrupt and the corresponding status bit. The local processor provokes an outbound message interrupt by writing to either of the two outbound message registers: OMR0 or OMR1. OMISR can be accessed from the CSB or the PCI bus, but it is normally accessed only from the PCI bus. [Figure 12-2](#) shows the OMISR fields.

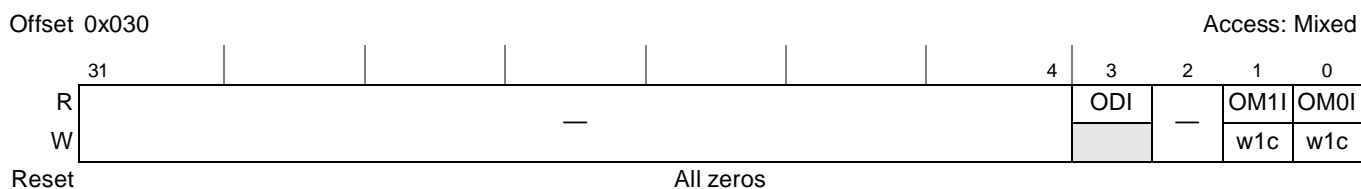


Figure 12-2. Outbound Message Interrupt Status Register (OMISR)

Table 12-4. OMIMR Field Descriptions (continued)

Bits	Name	Description
1	OM1IM	Outbound message 1 interrupt mask. 0 Outbound message 1 interrupt is allowed 1 Outbound message 1 interrupt is masked
0	OM0IM	Outbound message 0 interrupt mask. 0 Outbound message 0 interrupt is allowed 1 Outbound message 0 interrupt is masked

12.4.3 Inbound Message Registers (IMR0–IMR1)

The inbound message registers can be read from the PCI bus and the CSB in both host and agent modes. They can be written only from the PCI bus. [Figure 12-4](#) shows the IMR0 and IMR1 fields.



Figure 12-4. Inbound Message Registers (IMR0, IMR1)

[Table 12-5](#) describes the IMR_n register.

Table 12-5. IMR0 and IMR1 Field Descriptions

Bits	Name	Description
31–0	IMSG _n	Inbound message <i>n</i> . Contains generic data to be passed between the local processor and external hosts.

12.4.4 Outbound Message Registers (OMR0–OMR1)

The outbound message registers can be read from the PCI bus and the CSB in both host and agent modes. They can be written only from the CSB.

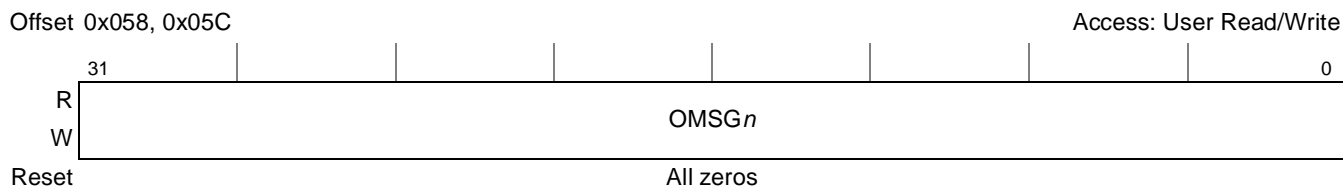


Figure 12-5. Outbound Message Registers (OMR0–OMR1)

[Table 12-6](#) describes the OMR_n registers.

Table 12-6. OMR0 and OMR1 Field Descriptions

Bits	Name	Description
31–0	OMSG _n	Outbound message <i>n</i> . Contains generic data to be passed between the local processor and external hosts.

12.4.5 Doorbell Registers

The following sections describe the outbound and inbound doorbell registers.

12.4.5.1 Outbound Doorbell Register (ODR)

ODR is accessible from the PCI bus and the CSB in both host and agent modes. [Figure 12-6](#) shows the ODR n fields.

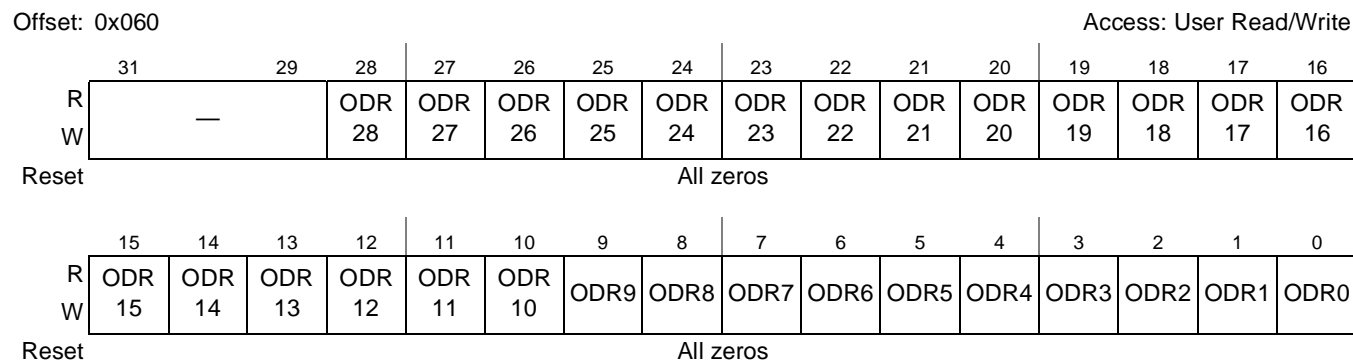


Figure 12-6. Outbound Doorbell Register (ODR)

[Table 12-7](#) describes the ODR registers.

Table 12-7. ODR Field Descriptions

Bits	Name	Description
31–29	—	Reserved
28–0	ODR n	Outbound doorbell n . Write 1 from the CSB to set. Write 1 from the PCI bus to clear. Writing 0 has no effect. (Writing a bit in this register from the CSB causes an interrupt ($\overline{\text{PCI_INTA}}$) to be generated.)

12.4.5.2 Inbound Doorbell Register (IDR)

IDR is accessible from the PCI bus and the CSB in both host and agent modes. [Figure 12-7](#) shows the IDR fields.

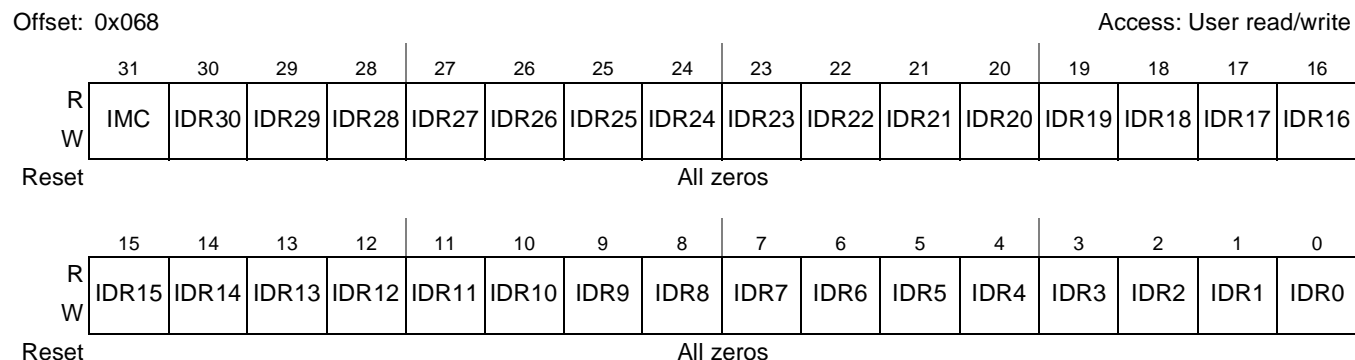


Figure 12-7. Inbound Doorbell Register (IDR)

[Table 12-8](#) describes the IDR registers.

Table 12-8. IDR Field Descriptions

Bits	Name	Descriptions
31	IMC	Inbound machine check. Write 1 from the PCI bus to set. Write 1 from the CSB to clear. Writing 0 has no effect. Writing this bit from the PCI bus causes a machine check interrupt to be generated to the local processor.
30–0	IDR n	Inbound doorbell n . Write 1 from the PCI bus to set. Write 1 from the CSB to clear. Writing 0 has no effect. Writing a bit in this register from the PCI bus causes an interrupt to be generated to the local processor.

12.4.6 Inbound Message Interrupt Status Register (IMISR)

The IMISR contains the interrupt status of the doorbell and message register events. Writing a 1 to IM1I clears the bit. The events are generated by the PCI masters.

[Figure 12-8](#) shows the IMISR fields.

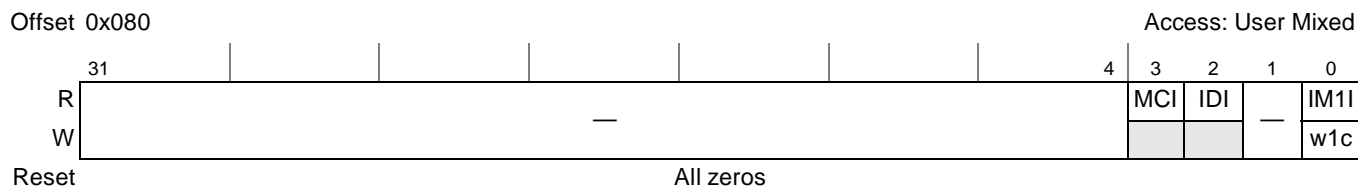


Figure 12-8. Inbound Message Interrupt Status Register (IMISR)

Table 12-9 describes the IMISR register.

Table 12-9. IMISR Field Descriptions

Bits	Name	Descriptions
31–5	—	Reserved
4	MCI	Machine check interrupt. Indicates whether a machine check interrupt condition was generated by setting the IDR[31]. The interrupt is cleared by writing a 1 to IDR[IMC] from the CSB. 0 No machine check interrupt 1 There is a machine check interrupt
3	IDI	Inbound doorbell interrupt. Indicates whether an inbound doorbell interrupt occurred. 0 No inbound doorbell interrupt 1 There is an inbound doorbell interrupt
2	—	Reserved
1	IM1I	Inbound message 1 interrupt. Indicates whether an inbound message 1 interrupt occurred. Write 1 to this position to clear this bit. 0 No inbound message 1 interrupt. 1 There is an inbound message 1 interrupt.
0	IM0I	Inbound message 0 interrupt. Indicates whether an inbound message 0 interrupt occurred. Write 1 to this position to clear this bit. 0 No inbound message 0 interrupt. 1 There is an inbound message 0 interrupt.

12.4.7 Inbound Message Interrupt Mask Register (IMIMR)

This register contains the interrupt mask of the doorbell and message register events generated by the PCI master. Figure 12-9 shows the IMIMR fields.

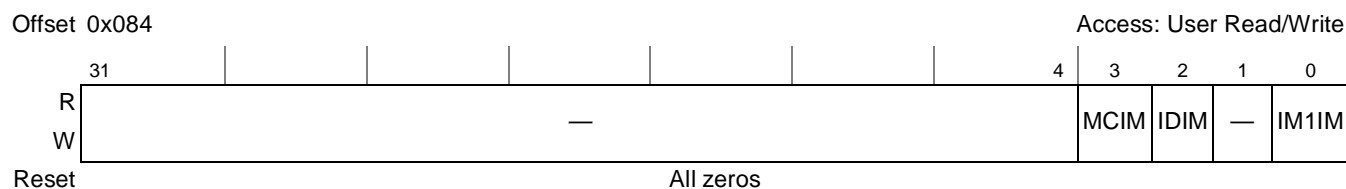


Figure 12-9. Inbound Message Interrupt Mask Register (IMIMR)

Table 12-10 describes the IMISR register.

Table 12-10. IMIMR Field Descriptions

Bits	Name	Description
31–5	—	Reserved
4	MCIM	Machine check interrupt mask. 0 Machine check interrupt from the IDR is allowed 1 Machine check interrupt is masked. IMISR[MC1] is cleared
3	IDIM	Inbound doorbell interrupt mask. 0 Inbound doorbell interrupt is allowed 1 Inbound doorbell interrupt is masked. IMISR[IDI] is cleared.
2	—	Reserved

Table 12-10. IMIMR Field Descriptions (continued)

Bits	Name	Description
1	IM1IM	Inbound message 1 interrupt mask. 0 Inbound message 1 interrupt is allowed 1 Inbound message 1 interrupt is masked. IMISR[IM1] is cleared
0	IM0IM	Inbound message 0 interrupt mask. 0 Inbound message 0 interrupt is allowed 1 Inbound message 0 interrupt is masked. IMISR[IM0] is cleared

12.4.8 DMA Registers

Each DMA channel has a set of seven 32-bit registers (mode, status, current descriptor address, next descriptor address, source address, destination address, and byte count) to support transactions. The following sections describe the format of the DMA support registers.

12.4.8.1 DMA Mode Register (DMAMR_n)

This section describes the DMA mode register. The mode register allows software to start the DMA transfer and to control various DMA transfer characteristics. [Figure 12-10](#) shows the DMAMR_n fields.

Offset: 0x100, 0x180, 0x200, 0x280

Access: User read/write

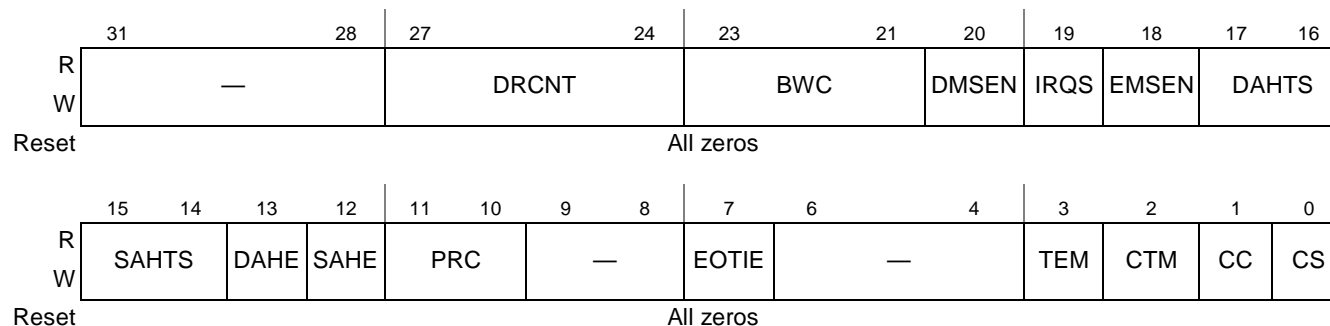


Figure 12-10. DMA Mode Register (DMAMR_n)

[Table 12-11](#) describes the DMAMR_n register.

Table 12-11. DMAMR_n Field Descriptions

Bits	Name	Description
31–28	—	Reserved
27–24	DRCNT	DMA request count. This field specifies the number of cache lines transferred per DMA request assertion if EMSEN is 1. This field is not used if EMSEN is 0. 0101 1 cache line 0110 2 cache lines 0111 4 cache lines 1000 8 cache lines 1001 16 cache lines 1010 32 cache lines Others Reserved

Table 12-11. DMAMR_n Field Descriptions (continued)

Bits	Name	Description
23–21	BWC	Bandwidth control. Only applies when multiple channels are executing transfers concurrently. The field determines how many cache lines a given channel is allowed to transfer after it is granted access to the IOS interface and before it releases the interface to the next channel. This allows the user to prioritize the DMA channels. The BWC values are listed as follows: 000 1 cache line 001 2 cache lines 010 4 cache lines 011 8 cache lines 100 16 cache lines Others Reserved
20	DMSEN	Direct mode snoop enable. This bit controls snooping of direct mode DMA transactions. 0 Snooping is disabled 1 Snooping is enabled
19	IRQS	Interrupt steer. This bit determines the destination of the DMA interrupts. 0 All DMA interrupts are routed to the on-chip interrupt controller 1 All DMA interrupts are routed to the PCI bus through <u>PCI_INTA</u>
18	EMSEN	External master start enable. This bit is cleared when the DMA transfer has completed, so it must be set again for each transfer. 0 The channel is started by software setting the CS bit 1 The channel is started by hardware asserting the <u>DREQ</u> pin
17–16	DAHTS	Destination address hold transfer size. This field indicates the transfer size used for each transaction when DAHE is 1. The byte count register must be in multiples of the size, and the destination address register must be aligned based on the size. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
15–14	SAHTS	Source address hold transfer size. This field indicates the transfer size used for each transaction when SAHE is 1. The byte count register must be in multiples of the size, and the source address register must be aligned based on the size. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
13	DAHE	Destination address hold enable. This bit allows the DMA controller to hold the destination address constant for every transfer. The size used for transfer is indicated by DAHTS. Note that hardware supports only aligned transfers for this feature. 0 Do not hold the destination address constant 1 Hold the destination address constant Note: The DMA does not support address hold when the external trigger mode is selected (EMSEN = 1). Note: The DMA does not support address hold for both the source and the destination at the same transfer.
12	SAHE	Source address hold enable. This bit allows the DMA controller to hold the source address constant for every transfer. The size used for transfer is indicated by SAHTS. Note that hardware supports only aligned transfers for this feature. 0 Do not hold the source address constant 1 Hold the source address constant Note: The DMA does not support address hold when the external trigger mode is selected (EMSEN = 1). Note: The DMA does not support address hold for both the source and the destination at the same transfer.

Table 12-11. DMAMR_n Field Descriptions (continued)

Bits	Name	Description
11–10	PRC	PCI read command. This field indicates the type of PCI read command to use. 00 Reserved 01 PCI read line 10 PCI read multiple 11 Reserved
9–8	—	Reserved
7	EOTIE	End-of-transfer interrupt enable. This bit determines whether an interrupt is generated at the completion of a DMA transfer. End-of-transfer is defined as the end of a direct mode transfer or in chaining mode, as the end of the transfer of the last segment of a chain. 0 No EOT interrupt is generated 1 EOT interrupt is generated
6–4	—	Reserved
3	TEM	Transfer error mask. This bit determines the DMA response in the event of a transfer error. 0 The DMA will halt when a transfer error occurs. 1 The DMA will complete the transfer regardless of whether a transfer error occurs. Note: Regardless of the setting of TEM, if an error condition was detected during the DMA transfer, it will cause DMASR _n [TE] to be set.
2	CTM	Channel transfer mode. 0 Chaining mode 1 Direct mode
1	CC	Channel continue. This bit applies only to chaining mode. Setting this bit indicates that the current descriptor segment should be repeated. CC is cleared by the DMA once the repeat takes effect, so it only causes a single repeat. 0 Normal chaining 1 DMACDAR is not loaded from DMANDAR, causing a repeat of the current descriptor segment
0	CS	Channel start. A 0-to-1 transition occurring on this bit when the channel is not busy (SR[CB] bit is 0) will start the DMA process. If the channel is busy and a 0-to-1 transition occurs, the DMA channel will restart from a previous halt condition. A 1-to-0 transition when the channel is busy (CB bit is 1) will halt the DMA process. Nothing happens if the channel is not busy and a 1-to-0 transition occurs. This bit is cleared by the DMA at the end of a transfer.

12.4.8.2 DMA Status Register (DMASR_n)

This section describes the DMA status register. The status register reports various DMA conditions during and after the DMA transfer. Writing a 1 to a specific set bit clears the bit. [Figure 12-11](#) shows the DMASR_n fields.

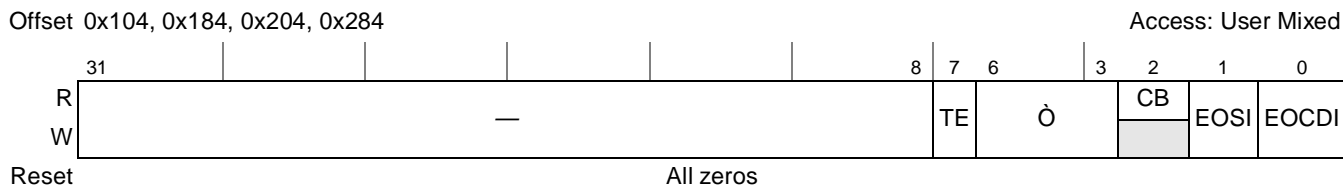


Figure 12-11. DMA Status Register (DMASR_n)

Table 12-12 describes the $DMASR_n$ register.

Table 12-12. $DMASR_n$ Field Descriptions

Bits	Name	Description
31–8	—	Reserved
7	TE	Transfer error. Set when there is an error condition during the DMA transfer.
6–3	—	Reserved
2	CB	Channel busy. This bit indicates whether the channel is busy. It is cleared as a result of any of the following conditions: an error or completion of the DMA transfer. 0 No DMA transfer is currently in progress 1 A DMA transfer is currently in progress
1	EOSI	End-of-segment interrupt. After transferring a segment of data, if the $DMACDAR_n$ [EOSIE] bit in the current descriptor address register is set, this bit is set and an interrupt is generated.
0	EOCDI	End-of-chain/direct interrupt. When the last DMA transfer is finished, either in chaining or direct mode, if $DMAMR$ [EOTIE] is set, this bit is set and an interrupt is generated.

12.4.8.3 DMA Current Descriptor Address Register ($DMACDAR_n$)

$DMACDAR_n$ contains the address of the current segment descriptor being transferred. In chaining mode, software must initialize this register to point to the first descriptor in the chain. After processing the first descriptor, the DMA controller moves the contents of the next descriptor address register into $DMACDAR$, loads the following descriptor into $DMANDAR$, and executes the current transfer.

Figure 12-12 shows the $DMACDAR_n$ fields.

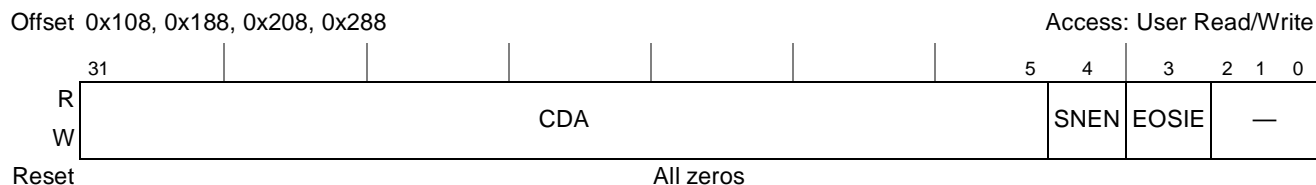


Figure 12-12. DMA Current Descriptor Address Register ($DMACDAR_n$)

Table 12-13 describes the $DMACDAR_n$ register.

Table 12-13. $DMACDAR_n$ Field Descriptions

Bits	Name	Description
31–5	CDA	Current descriptor address. This field contains the current descriptor address of the segment descriptor in memory. It must be aligned on an 8-word boundary.
4	SNEN	Snoop enable. 0 Snooping is disabled on DMA transactions of the current segment. 1 Snooping is enabled on DMA transactions of the current segment.
3	EOSIE	End-of-segment interrupt enable 0 No end-of-segment interrupt is generated. 1 An interrupt is generated when the current DMA transfer for the current descriptor is finished.
2–0	—	Reserved

12.4.8.4 DMA Source Address Register (DMASAR_n)

DMASAR_n indicates the address from which the DMA controller will be reading data. The software must ensure that this is a valid memory address. [Figure 12-13](#) shows the DMASAR_n.



Figure 12-13. DMA Source Address Register (DMASAR_n)

[Table 12-14](#) describes the DMASAR_n register.

Table 12-14. DMASAR_n Field Descriptions

Bits	Name	Description
31–0	SA	Source address of DMA transfer. The content of this field is updated after each DMA read operation.

12.4.8.5 DMA Destination Address Register (DMADAR_n)

DMADAR_n indicates the address to which the DMA controller will be writing data. The software must ensure that this is a valid memory address. [Figure 12-14](#) shows the DMADAR_n fields.

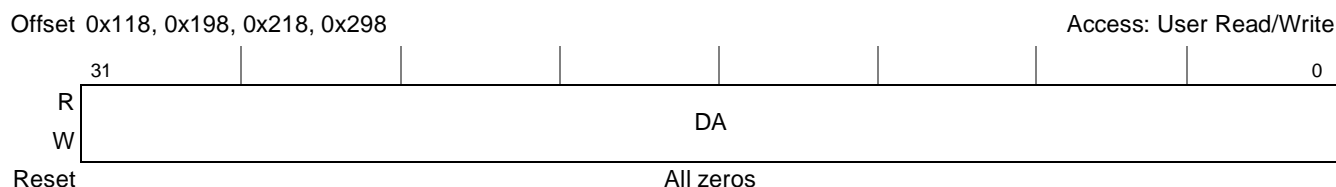


Figure 12-14. DMA Destination Address Register (DMADAR_n)

[Table 12-15](#) describes the DMADAR_n register.

Table 12-15. DMASAR_n Field Descriptions

Bits	Name	Description
31–0	DA	Destination address of DMA transfer. Updated after each DMA write operation.

Table 12-17. DMANDAR_n Field Descriptions (continued)

Bits	Name	Descriptions
2–1	—	Reserved
0	EOTD	End-of-transfer descriptor. 0 This descriptor contains a link to another descriptor. 1 This descriptor is the last to be executed.

12.4.8.8 DMA General Status Register (DMAGSR)

DMAGSR provides faster access to the status bits by combining the status bits of all of the DMA channels into one register. Each byte of this register provides the value of bits 7–0 of a channel’s DMA status register. These bits are cleared by writing to the individual DMA status registers. [Figure 12-17](#) shows the DMAGSR fields.

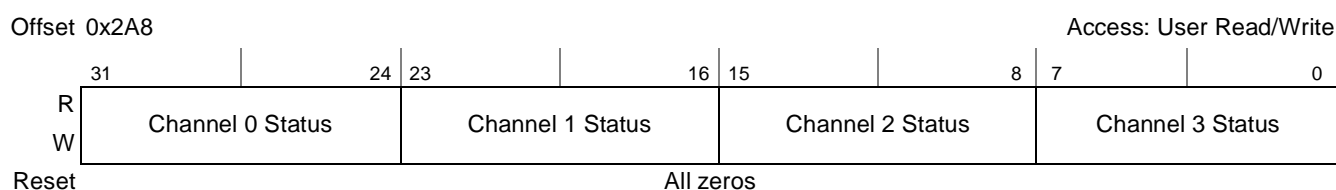


Figure 12-17. DMA General Status Register (DMAGSR)

12.5 Functional Description

12.5.1 Message Unit

An embedded processor is often part of a larger system containing many processors and distributed memory. These processors tend to work on tasks independent of the host and other peripheral processors in the system. Because of the independent nature of the tasks, it is necessary to provide a communication mechanism between the peripheral processors and the rest of the system. One such method is the use of messages. This block provides a messaging unit to further facilitate communications between host and peripheral. The message unit uses generic messages and doorbell registers.

12.5.1.1 Messaging Registers (IMR0–IMR1, OMR0–OMR1)

There are two 32-bit inbound message registers (IMR0–IMR1) and two 32-bit outbound message registers (OMR0–OMR1). IMR0 and IMR1 allow a remote host or PCI master to write a 32-bit value that, in turn, causes an interrupt request to the on-chip interrupt controller that drives an interrupt line to the local processor. OMR0 and OMR1 allow the local processor to write an outbound message which, in turn, causes the outbound interrupt signal $\overline{\text{PCI_INTA}}$ to assert.

The interrupt to the local processor is cleared by writing 1 to the appropriate IMISR bit. The interrupt to PCI ($\overline{\text{PCI_INTA}}$) is cleared by writing 1 to the appropriate OMISR bit.

12.5.1.2 Doorbell Registers (IDR and ODR)

This block contains the inbound doorbell register (IDR) and the outbound doorbell register (ODR). The inbound doorbell allows a remote processor to set a bit in the register from the PCI bus. This, in turn, generates an interrupt request to the on-chip interrupt controller that drives an interrupt line to the local processor. The local processor can write to the ODR, which causes the outbound interrupt signal `PCI_INTA` to assert, thus interrupting the remote processor on the PCI bus.

The interrupt to the local processor is cleared by writing 1 to the appropriate IDR bit. The interrupt to PCI (`PCI_INTA`) is cleared by writing 1 to the appropriate ODR bit.

12.5.2 DMA Controller

The DMA controller transfers blocks of data independent of the local processor or PCI hosts. Data movement occurs on the PCI bus and/or CSB. The DMA module has four high-speed DMA channels, which share buffer space in the IOS to facilitate the gathering and sending of data. Both the local processor and PCI masters can initiate a DMA transfer.

Features of the DMA controller include the following:

- Four channels
- Concurrent execution across multiple channels with programmable bandwidth control
- All channels are accessible by local processor and remote PCI masters
- Unaligned transfer capability
- Data chaining and direct mode
- Interrupt on completed segment, chain, and error

Figure 12-18 shows a diagram of the DMA controller in the integrated device.

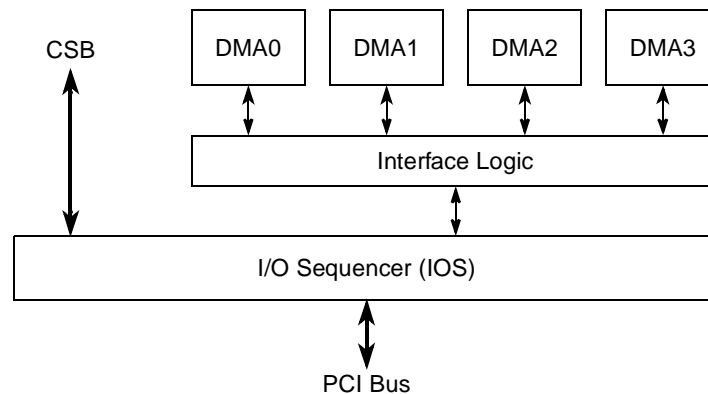


Figure 12-18. DMA Controller Block Diagram

12.5.3 DMA Operation

The DMA controller operates in the following two modes:

- Direct mode, in direct mode, the DMA controller does not read a chain of descriptors from memory but instead uses the current parameters in the DMA registers to start a DMA transfer. The DMA

transfer finishes after all the bytes specified in the byte count register have been transferred. See [Section 12.6.1, “Initialization Steps in Direct Mode,”](#) for more details on initialization steps.

- Chaining mode, in chaining mode, the DMA controller loads descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the descriptor information loaded for each segment. Once the current segment is finished, the DMA controller reads the next descriptor from memory and begins another DMA transfer. The process is finished if the current descriptor is the last one in the chain. See [Section 12.6.2, “Initialization Steps in Chaining Mode,”](#) for more details on initialization steps.

In both modes, setting the start bit in the DMA mode register begins the DMA transfer.

The DMA controller supports misaligned transfers for both the source and destination addresses. It gathers data beginning at the source address and aligns the data accordingly before sending it to the destination address. The DMA controller assumes that the source and destination addresses are valid PCI or CSB memory addresses.

Accesses to CSB memory depend on the alignment of the source and destination addresses and the size of the transfer. The DMA controller transfers a full cache line whenever possible. On misaligned addresses, full cache line transfers (32 byte bursting) occur if the source and destination offsets end in the same byte offset. For example, if the destination address is 0x4000_1050 and the source address is 0x9000_2050, the transfer bursts because both end in 0x50. However, if the destination address is 0x4000_1050 and the source is 0x9000_2000, the transfer does not burst because the last byte offset is not the same. On misaligned destination addresses, subtransfers of less than a cache line occur on the initial and final beats of the transfer while full cache lines occur on intermediate beats.

Configuring a DMA channel for address hold mode $DMAMR_n$ precludes cache line transfers.

PCI memory read operations depend on the PRC (PCI read command) field in the mode register, the alignment of the source address, and the size of the transfer. The DMA controller attempts to read a full cache line whenever possible. Writing to PCI memory depends on the alignment of the destination address and the size of the transfer.

12.5.3.1 External Control

The DMA transfer of any channel, in either direct mode or chaining mode, can be controlled by the DMA request input signals. External control is enabled by setting the external master start enable (EMSEN) bit instead of the channel start (CS) bit in the DMA mode register ($DMAMR_n$).

When using external control, the following restrictions apply:

- Both the source and destination addresses must be aligned to 32-byte boundaries.
- In chaining mode all byte count values except the last must be multiples of 32 bytes.

A falling edge on \overline{DREQ}_n sets $DMAMR_n[CS]$ to start the transfer and asserts the corresponding \overline{DACK}_n output signal. The number of cache lines specified by the DMA request count (DRCNT) bit in the DMA mode register ($DMAMR_n$)—or the remaining byte count, if smaller— is transferred, and the CS bit and the \overline{DACK}_n output signal are then cleared and negated to halt the transfer until the next \overline{DREQ}_n assertion.

When using the \overline{DACK}_n handshake signal, \overline{DREQ}_n should remain asserted until \overline{DACK}_n is asserted, at which point \overline{DREQ}_n may be negated. \overline{DREQ}_n may be asserted again to resume the transfer or start a new transfer once \overline{DACK}_n has been negated.

If the \overline{DACK}_n handshake signal is not used, \overline{DREQ}_n should remain asserted until the first transaction of the DMA transfer appears on the external interface, at which point \overline{DREQ}_n may be negated. \overline{DREQ}_n may be asserted again to resume the transfer or start a new transfer as early as one clock cycle after its negation, even if the current transfer is still active. The DMA controller is able to record a new assertion of \overline{DREQ}_n while a transfer is in progress, with the effect of setting $DMAMR_n[CS]$ again once the transfer has been halted.

Once a transfer has been completed, $DMAMR_n[EMSEN]$ is cleared by the DMA controller, and $DMAMR_n[CS]$ will not be set again until $DMAMR_n[EMSEN]$ has been set by the user. The assertion of \overline{DREQ}_n and the setting of $DMAMR_n[EMSEN]$ may occur in either order; whichever occurs later will trigger the DMA transfer.

The \overline{DDONE}_n output signal is asserted when the DMA transfer has completed, that is, all bytes specified in the byte count register or the descriptors have been transferred. This signal could be used as an indication that the number of cache lines transferred might be smaller than that specified by the $DRCNT$ field.

NOTE

The \overline{DACK}_n and \overline{DDONE}_n output signals are intended as handshake signals for \overline{DREQ}_n . They are asserted and negated according to the DMA controller internal logic. These signals are not synchronized to the transactions appearing on the external pins of the device. Specifically, the negation of \overline{DACK}_n or the assertion of \overline{DDONE}_n does not mean that all transactions of the DMA transfer have been completed as seen on the external pins.

See [Section 12.6.3, “Initialization Steps in Direct Mode with External Control,”](#) and [Section 12.6.4, “Initialization Steps in Chaining Mode with External Control,”](#) for more details on initialization steps.

12.5.3.2 DMA Coherency

The four DMA channels use up to four cache lines (128 bytes) of buffer space in the IOS in addition to 16 bytes of local buffer space. Because no address snooping occurs in these internal queues, data posted in these queues is not visible to the rest of the system while a DMA transfer is in progress. It is the responsibility of application software to ensure the coherency of the region being transferred during the DMA process.

Snooping of the CPU or processor data cache is selectable during DMA transactions. A snoop bit is provided in the DMA current descriptor address register ($DMACDAR_n$) and the DMA next descriptor address register ($DMANDAR_n$) that allows software to control when the cache is snooped on a per segment basis.

12.5.3.3 Halt and Error Conditions

DMA transfers are halted either by clearing the CS (channel start) bit in the DMA mode register (DMAMR n) or when encountering an error condition. In either case, the application software can do one of the following:

- Continue the DMA transfer
- Reconfigure the DMA for a new transfer
- Leave the channel in the halted state

When a DMA channel is halted, its programming model is completely accessible. If the DMA is halted due to an error condition, the TE (transfer error) bit in the DMA status register (DMASR n) must be cleared before the transfer can be resumed or a new transfer initiated. Note that the TE bit is not cleared automatically by hardware.

12.5.4 DMA Segment Descriptors

DMA segment descriptors contain the source and destination addresses of the data segment, the segment byte count, and a link to the next descriptor. Segment descriptors are built on cache-line (32-byte) boundaries in either CSB or PCI memory and are linked together into chains using the next-descriptor-address field.

Table 12-18. DMA Segment Descriptor Fields

Descriptor Field	Description
Source address	Contains the source address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field will be loaded into the DMA source address register (DMASAR n).
Destination address	Contains the destination address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field will be loaded into the DMA destination address register (DMADAR n).
Next descriptor address	Points to the next descriptor in memory. After the DMA controller reads the descriptor from memory, this field will be loaded into the DMA next descriptor address register (DMANDAR n).
Byte count	Contains the number of bytes to transfer. After the DMA controller reads the descriptor from memory, this field will be loaded into the DMA byte count register (DMABCR n).

Application software initializes the current DMA current descriptor address register (DMACDAR n) to point to the first descriptor in the chain. For each descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by the descriptor. The DMA controller traverses the descriptor chain until reaching the last descriptor (with its EOTD bit set).

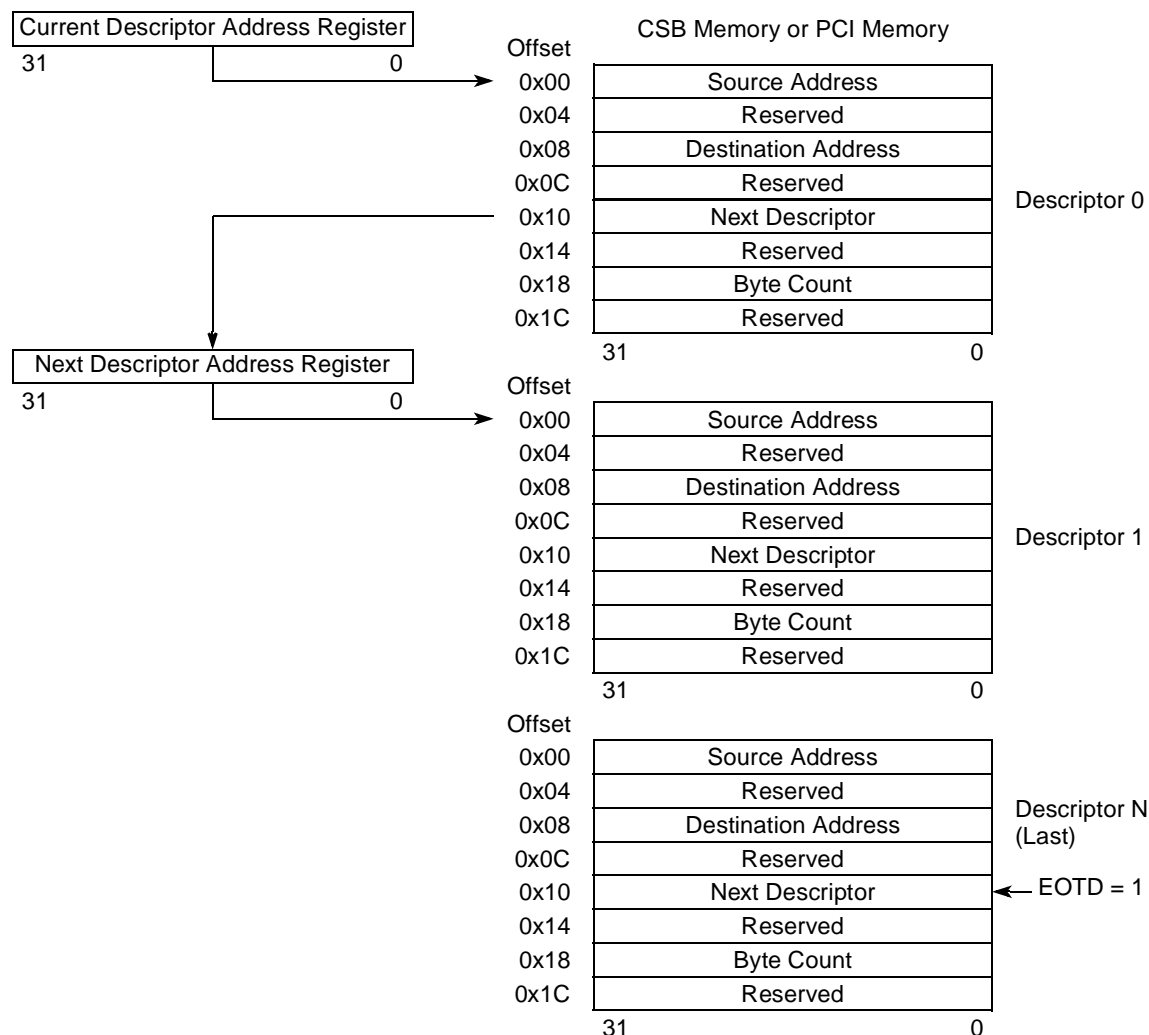


Figure 12-19. DMA Chain of Segment Descriptors

12.5.4.1 Descriptor in Big-Endian Mode

In big-endian mode, the descriptor in CSB memory should be programmed such that data appears in ascending significant-byte order. If segment descriptors are written to memory located in the CSB, they should be treated like they are translated from big-endian to little-endian mode.

Example: Big-endian mode descriptor's data structure. Note that the descriptor structure must be aligned on an 8-word boundary.

```
struct {
    double a;           /* 0x1122334455667788 double word*/
    double b;           /* 0x55667788aabbccdd double word*/
    double c;           /* 0x8765432101234567 double word */
    double d;           /* 0x0123456789abcdef double word */
} Descriptor;
Results: Source Address = 0x44332211 <MSB..LSB>
        Destination Address = 0x88776655 <MSB..LSB>
        Next Descriptor Address = 0x21436587 <MSB..LSB>
        Byte Count = 0x67452301 <MSB..LSB>
```

12.5.4.2 Descriptor in Little-Endian Mode

In little-endian mode, each segment descriptor should be programmed in descending significant-byte order.

Example: Little-endian mode descriptor's data structure. Note that the descriptor structure must be aligned on an 8-word boundary.

```
struct {
    double a;           /* 0x8877665544332211 double word*/
    double b;           /* 0x1122334488776655 double word*/
    double c;           /* 0x7654321012345678 double word */
    double d;           /* 0x0123456776543210 double word */
} Descriptor;
Results: Source Address = 0x44332211 <MSB..LSB>
        Destination Address = 0x88776655 <MSB..LSB>
        Next Descriptor Address = 0x12345678 <MSB..LSB>
        Byte Count = 0x76543210 <MSB..LSB>
```

12.6 Initialization/Application Information

12.6.1 Initialization Steps in Direct Mode

The initialization steps of a DMA transfer in direct mode are described as follows:

1. Poll the CB (channel busy) bit in the DMA status register (DMASR n) to make sure the DMA channel is idle.
2. Initialize the DMASAR n , DMADAR n , and the DMABCR n .
3. Initialize DMAMR n [CTM]) to indicate direct mode. Other control parameters in the mode register can also be initialized here if necessary.
4. First clear then set the DMAMR n [CS] to start the DMA transfer.

12.6.2 Initialization Steps in Chaining Mode

The initialization steps of a DMA transfer in chaining mode are described as follows:

1. Build a chain of descriptor segments in memory. Refer to [Section 12.5.4, “DMA Segment Descriptors.”](#)

2. Poll the $DMASR_n[CB]$ to make sure the DMA channel is idle.
3. Initialize the $DMACDAR_n$ to point to the first descriptor in the chain.
4. Initialize the $DMAMR_n[CTM]$ to indicate chaining mode. Other control parameters in the mode register can also be initialized here if necessary.
5. First clear then set the $DMAMR_n[CS]$ to start the DMA transfer.

12.6.3 Initialization Steps in Direct Mode with External Control

The initialization steps of a DMA transfer in direct mode with external control are described as follows:

1. Poll the CB (channel busy) bit in the DMA status register ($DMASR_n$) to make sure the DMA channel is idle.
2. Initialize the DMA source address register ($DMASAR_n$), the DMA destination address register ($DMADAR_n$), and the DMA byte count register ($DMABCR_n$).
3. Set $DMAMR_n[CTM]$ to indicate direct mode, program the DRCNT field, and set the EMSEN bit. Other control parameters in the mode register can also be initialized here if necessary.

12.6.4 Initialization Steps in Chaining Mode with External Control

The initialization steps of a DMA transfer in chaining mode with external control are described as follows:

1. Build a chain of descriptor segments in memory. Refer to [Section 12.5.4, “DMA Segment Descriptors,”](#) for more information.
2. Poll the CB (channel busy) bit in the DMA status register ($DMASR_n$) to make sure the DMA channel is idle.
3. Initialize the DMA current descriptor address register ($DMACDAR_n$) to point to the first descriptor in the chain.
4. Clear the $DMAMR_n[CTM]$ to indicate chaining mode, program the DRCNT field, and set the EMSEN bit. Other control parameters in the mode register can also be initialized here if necessary.



Chapter 13

PCI Bus Interface

The PCI interface is compatible with the *PCI Local Bus Specification*, Rev. 2.3. It is beyond the scope of this manual to document the intricacies of PCI. 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. Designers of systems incorporating PCI devices should refer to the respective specifications 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 in this chapter. Instead, the number of bits or bytes indicates the exact quantity.

13.1 PCI Introduction

The PCI controller acts as a bridge between the PCI interface and the CSB. The I/O sequencer buffers the data. [Figure 13-1](#) is a high-level block diagram of the PCI controller.

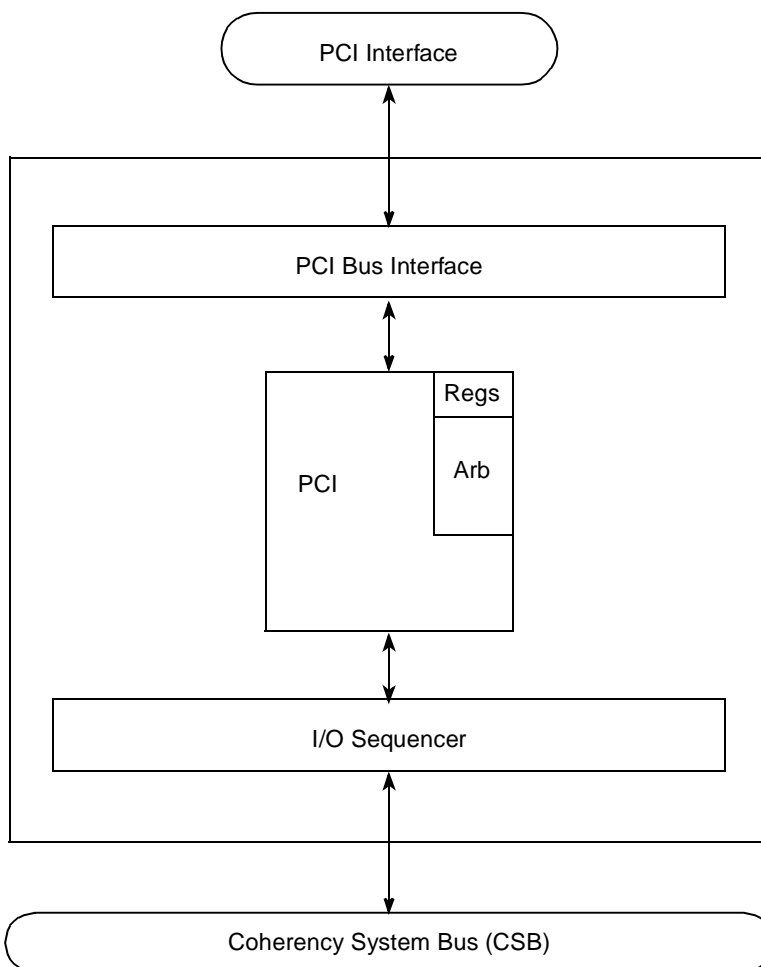


Figure 13-1. PCI Controller Block Diagram

The PCI controller connects the processor and memory system to the I/O components through the PCI system bus. This interface acts as both initiator (master) and target (slave) 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.

Note that PCI supports up to three external masters.

The PCI interface can function as either a PCI host bridge referred to as host mode or a peripheral device on the PCI bus referred to as agent mode. See [Section 13.4.4.4, “Host Mode Configuration Access,”](#) for more information. Note that the PCI controller can be configured from the PCI bus while in agent mode. An address translation mechanism is provided to map PCI memory windows between the PCI bus and the internal bus.

The PCI interface does not flush pending outbound writes as a result of an inbound read command. Systems must not rely on inbound reads to ensure all pending outbound writes have completed. For example, consider the case where a core writes data to a PCI device and then updates a flag in the local DDR memory indicating the write to PCI has completed. An external PCI master may misread the flag ahead of the actual write transaction's completion on the PCI bus.

13.1.1 PCI Features

The PCI controller includes the following features:

- PCI specification revision 2.3 compliant
- 32-bit PCI interface support
- Host and 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
- On-chip arbitration supporting three masters on PCI
- Arbiter supports two-level priority request/grant signal pairs
- Supports 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
- Supports selectable snooping for inbound transactions
- Address translation units for address mapping between host and peripheral
- Supports parity
- PCI 3.3-V compatible

13.1.2 PCI Modes of Operation

PCI controller modes of operation are determined at reset by the reset configuration word high (RCWH) as described in [Section 4.3.2, “Reset Configuration Words.”](#) [Table 13-1](#) summarizes these modes.

Table 13-1. PCI Controller Modes

Parameter	Description	Section/Page
Host/agent configuration	Selects between host and agent mode for the PCI interface.	4.3.2.2.1/4-18
PCI clock output enable	Enables the PCI clock output signals.	/4-32
PCI arbiter enable	Enables the on-chip PCI bus arbiter	4.3.2.2/4-17

13.1.2.1 PCI Enable Configuration

The $\overline{\text{PCI_MODE}}$ signal enables PCI mode for QUICC Engine parallel I/O ports, which can function as PCI signals. See [Section 5.4.1.1, “PCI_MODE Signal,”](#) for more information.

13.1.2.2 Host/Agent Mode Configuration

The PCI controller can function as either a PCI host bridge (referred to as host mode) or a peripheral device on the PCI bus (referred to as agent mode). Note that host/agent mode selection is determined at power-up as summarized in [Section 4.3.2.2.1, “PCI Host/Agent Configuration.”](#)

When the device powers up in host mode, all inbound configuration accesses are ignored (and thus master aborted). When the device powers up in agent mode, it acknowledges inbound configuration accesses. Note that in PCI agent mode, the PCI controller ignores all PCI memory accesses except those to the memory-mapped registers until inbound address translation is enabled. In agent mode, configuration cycles are acknowledged if CFG_LOCK is 0 (see [Section 13.3.3.24, “PCI Function Configuration Register”](#)), either from reset configuration or after being cleared by software.

13.1.2.3 PCI Clock Output Enable Configuration

Normally when the device is in host mode, PCI clock outputs are provided for connection to other agents in the system. However it is possible to disable the PCI clock outputs of the device. When PCI clock outputs are disabled, PCI clock distribution should be done on the board by an external device. The benefit of disabling the PCI clock outputs is that the PCI clock device pins can be used for alternate functions.

13.1.2.4 PCI Arbiter Configuration

The interface can be configured to use an on-chip or off-chip PCI arbiter. Arbitration for PCI is determined by the value in RCWH[PCIARB]. See [Section 4.3.2.2, “Reset Configuration Word High Register \(RCWHR\),”](#) for more information.

13.2 PCI External Signal Description

[Table 13-2](#) shows the properties of the PCI signals.

Table 13-2. Signal Properties

Name	Function	Reset State	Pull Up
CPCI_HS_ENUM	CompactPCI hot swap enumerator	High impedance	Required
CPCI_HS_ES	CompactPCI hot swap ejector switch	—	—
CPCI_HS_LED	CompactPCI hot swap LED	Asserted	—
M66EN	66-MHz enable	—	—
PCI_AD[31:0]	PCIn address / data	High impedance	—
PCI_C/BE[3:0]	PCI bus command / byte enable	High impedance	—
PCI_DEVSEL	PCI device select	High impedance	Required
PCI_FRAME	PCI cycle frame	High impedance	Required
PCI_REQ[0:2]	PCI arbiter requests	Configuration-dependent	Required on inputs
PCI_GNT[0:2]	PCI arbiter grants	Configuration-dependent	—
PCI_IDSEL	PCI initialization device select	—	—
PCI_INTA	PCI interrupt A	High impedance	Required

Table 13-2. Signal Properties (continued)

Name	Function	Reset State	Pull Up
PCI_IRDY	PCI initiator ready	High impedance	Required
PCI_PAR	PCI parity	High impedance	—
PCI_PERR	PCI parity error	High impedance	Required
PCI_RESET_OUT	PCI reset output	Asserted	
PCI_SERR	PCI system error	High impedance	Required
PCI_STOP	PCI stop	High impedance	Required
PCI_TRDY	PCI target ready	High impedance	Required

Figure 13-2 shows the external PCI signals.

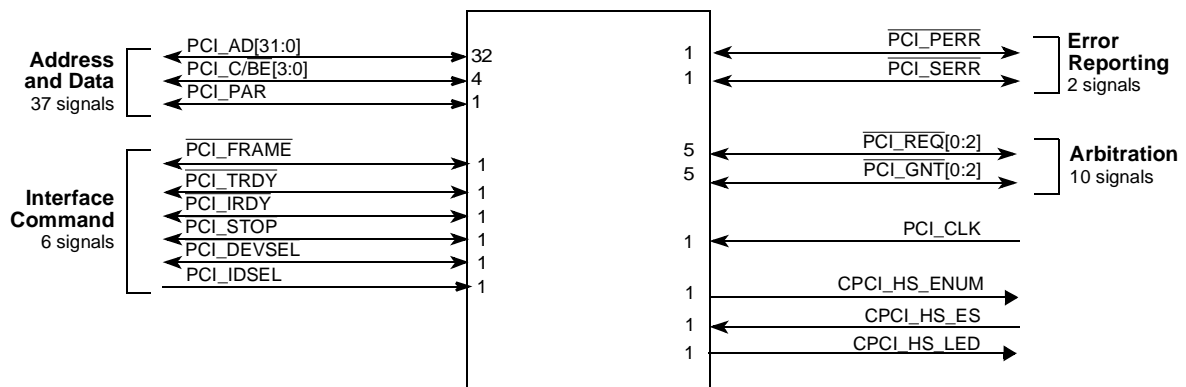


Figure 13-2. PCI Interface External Signals

Table 13-3 contains detailed descriptions of the external PCI interface signals.

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description				
CPCI_HS_ENUM	O	CompactPCI hot swap enumerator. Used for the hot swap interface to connect to the host as the enumeration request in a compact PCI system. This signal is used for agent mode only.				
		<table border="1"> <tr> <td>State Meaning</td> <td>Asserted—This card was inserted and needs to be configured, or this card is about to be extracted and needs to be removed from the system resources list. Negated—No action is needed.</td> </tr> <tr> <td>Timing</td> <td>Assertion/Negation—No timing is specified</td> </tr> </table>	State Meaning	Asserted—This card was inserted and needs to be configured, or this card is about to be extracted and needs to be removed from the system resources list. Negated—No action is needed.	Timing	Assertion/Negation—No timing is specified
		State Meaning	Asserted—This card was inserted and needs to be configured, or this card is about to be extracted and needs to be removed from the system resources list. Negated—No action is needed.			
Timing	Assertion/Negation—No timing is specified					
CPCI_HS_ES	I	CompactPCI hot swap ejector switch. Used for agent mode only. In a compact PCI system this input signal is used for the hot swap interface to connect to the ejector switch logic.				
		<table border="1"> <tr> <td>State Meaning</td> <td>Asserted—The switch is open. Negated—The switch is closed.</td> </tr> <tr> <td>Timing</td> <td>Assertion/Negation—No timing is specified</td> </tr> </table>	State Meaning	Asserted—The switch is open. Negated—The switch is closed.	Timing	Assertion/Negation—No timing is specified
		State Meaning	Asserted—The switch is open. Negated—The switch is closed.			
Timing	Assertion/Negation—No timing is specified					

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description				
CPCI_HS_LED	O	CompactPCI hot swap LED. Used for the hot swap interface to connect to the hot swap LED in a CompactPCI system. This signal is used for agent mode only.				
		<table border="1"> <tr> <td>State Meaning</td> <td>Asserted—Output is driving logic 1 to illuminate the hot swap LED. Negated—Output is driving logic 0 to turn off the hot swap LED.</td> </tr> <tr> <td>Timing</td> <td>Assertion/Negation—No timing is specified</td> </tr> </table>	State Meaning	Asserted—Output is driving logic 1 to illuminate the hot swap LED. Negated—Output is driving logic 0 to turn off the hot swap LED.	Timing	Assertion/Negation—No timing is specified
		State Meaning	Asserted—Output is driving logic 1 to illuminate the hot swap LED. Negated—Output is driving logic 0 to turn off the hot swap LED.			
Timing	Assertion/Negation—No timing is specified					
M66EN	I	66-MHz enable. Determines the AC timing of the PCI interface.				
		<table border="1"> <tr> <td>State Meaning</td> <td>Asserted—The PCI interface signals use the 66-MHz PCI AC timing parameters. Negated—The PCI interface signals use the 33-MHz PCI AC timing parameters.</td> </tr> <tr> <td>Timing</td> <td>Assertion/Negation—Constant</td> </tr> </table>	State Meaning	Asserted—The PCI interface signals use the 66-MHz PCI AC timing parameters. Negated—The PCI interface signals use the 33-MHz PCI AC timing parameters.	Timing	Assertion/Negation—Constant
		State Meaning	Asserted—The PCI interface signals use the 66-MHz PCI AC timing parameters. Negated—The PCI interface signals use the 33-MHz PCI AC timing parameters.			
Timing	Assertion/Negation—Constant					
PCI_AD[31:0]	I/O	PCI address/data bus. During an address phase, these signals contain a physical address. During a data phase, these signals contain the data bytes.				
		O	Outputs for the bi-directional PCI address/data bus.			
		<table border="1"> <tr> <td>State Meaning</td> <td>Asserted/Negated—Represents the physical address during the address phase of a PCI transaction. During the data phase(s) of a PCI transaction, the PCI address/data bus contain the data being written. PCI_AD[7:0] define the LSB and, PCI_AD[31:24] define the MSB.</td> </tr> <tr> <td>Timing</td> <td>Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i></td> </tr> </table>	State Meaning	Asserted/Negated—Represents the physical address during the address phase of a PCI transaction. During the data phase(s) of a PCI transaction, the PCI address/data bus contain the data being written. PCI_AD[7:0] define the LSB and, PCI_AD[31:24] define the MSB.	Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
	State Meaning	Asserted/Negated—Represents the physical address during the address phase of a PCI transaction. During the data phase(s) of a PCI transaction, the PCI address/data bus contain the data being written. PCI_AD[7:0] define the LSB and, PCI_AD[31:24] define the MSB.				
	Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>				
	I	Inputs for the bi-directional PCI address/data bus.				
<table border="1"> <tr> <td>State Meaning</td> <td>Asserted/Negated—Represents the address to be decoded as a check for device select during the address phase of a PCI transaction or the data being received during the data phase(s) of a PCI transaction. PCI_AD[7:0] define the LSB and, PCI_AD[31:24] define the MSB.</td> </tr> <tr> <td>Timing</td> <td>Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i></td> </tr> </table>		State Meaning	Asserted/Negated—Represents the address to be decoded as a check for device select during the address phase of a PCI transaction or the data being received during the data phase(s) of a PCI transaction. PCI_AD[7:0] define the LSB and, PCI_AD[31:24] define the MSB.	Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>	
State Meaning		Asserted/Negated—Represents the address to be decoded as a check for device select during the address phase of a PCI transaction or the data being received during the data phase(s) of a PCI transaction. PCI_AD[7:0] define the LSB and, PCI_AD[31:24] define the MSB.				
Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>					
PCI_C/BE[3:0]	I/O	PCI bus command/byte enable.				
		O	Outputs for the bi-directional command/byte enable.			
		<table border="1"> <tr> <td>State Meaning</td> <td>Asserted/Negated—During the address phase, PCI_CBE[3:0], define the bus command. Byte enables determine which byte lanes carry meaningful data for PCI bus data phases. The PCI_CBE[0] signal applies to the LSB.</td> </tr> <tr> <td>Timing</td> <td>Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i></td> </tr> </table>	State Meaning	Asserted/Negated—During the address phase, PCI_CBE[3:0], define the bus command. Byte enables determine which byte lanes carry meaningful data for PCI bus data phases. The PCI_CBE[0] signal applies to the LSB.	Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
	State Meaning	Asserted/Negated—During the address phase, PCI_CBE[3:0], define the bus command. Byte enables determine which byte lanes carry meaningful data for PCI bus data phases. The PCI_CBE[0] signal applies to the LSB.				
	Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>				
	I	Inputs for the bi-directional command/byte enable.				
<table border="1"> <tr> <td>State Meaning</td> <td>Asserted/Negated—During the address phase, PCI_CBE[3:0], indicate the command that another master is sending. During the PCI bus data phase, PCI_CBE[3:0], indicate which byte lanes are valid.</td> </tr> <tr> <td>Timing</td> <td>Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i></td> </tr> </table>		State Meaning	Asserted/Negated—During the address phase, PCI_CBE[3:0], indicate the command that another master is sending. During the PCI bus data phase, PCI_CBE[3:0], indicate which byte lanes are valid.	Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>	
State Meaning		Asserted/Negated—During the address phase, PCI_CBE[3:0], indicate the command that another master is sending. During the PCI bus data phase, PCI_CBE[3:0], indicate which byte lanes are valid.				
Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>					

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{PCI_DEVSEL}}$	I/O	PCI device select.	
	O	Outputs for the bi-directional device select.	
		State Meaning	Asserted—The PCI controller has decoded the address and is the target of the current access. Negated—The PCI controller has decoded the address and is not the target of the current access.
		Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
	I	Inputs for the bi-directional device select.	
		State Meaning	Asserted—Some PCI agents (other than this PCI controller) have decoded its address as the target of the current access. Negated—No PCI agent has been selected.
Timing		Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>	
$\overline{\text{PCI_FRAME}}$	I/O	PCI cycle frame. Used by the current PCI master to indicate the beginning and duration of an access.	
	O	Outputs for the bi-directional frame.	
		State Meaning	Asserted—The PCI controller acting as a PCI master which is initiating a bus transaction. While $\overline{\text{PCI_FRAME}}$ is asserted, data transfers may continue. Negated—If $\overline{\text{PCI_IRDY}}$ is asserted, indicates that the PCI transaction is in the final data phase; if $\overline{\text{PCI_IRDY}}$ is negated, indicates that the PCI bus is idle.
		Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
	I	Inputs for the bi-directional frame.	
		State Meaning	Asserted—Another PCI master is initiating a bus transaction. Negated—The transaction is in the final data phase or that the bus is idle.
Timing		Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>	
$\overline{\text{PCI_GNT0}}$	I/O	PCI arbiter grants. Output signal on this PCI controller when the arbiter is enabled. Input signal when the arbiter is disabled. Note: $\overline{\text{PCI_GNT}}[0]$ is a point-to-point signal. Every master has its own bus grant signal.	
	O	Outputs for the bi-directional arbiter grants.	
		State Meaning	Asserted—The PCI controller granted control of the PCI bus to agent 0. Negated—The PCI controller did not grant control of the PCI bus to agent 0.
		Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
	I	Inputs for the bi-directional arbiter grants.	
		State Meaning	Asserted—The PCI controller has been granted control of the PCI bus by an external arbiter. Negated—The PCI controller has not been granted control of the PCI bus by an external arbiter.
Timing		Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>	

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
$\overline{\text{PCI_GNT}}[1:2]$	O	PCI arbiter grants. Output signals on this PCI controller when the arbiter is enabled. Note that $\overline{\text{PCI_GNT}}_n$ is a point-to-point signal. Every master has its own bus grant signal.
		State Meaning Asserted—The PCI controller granted control of the PCI bus to agent <i>n</i> . Negated—The PCI controller did not grant control of the PCI bus to agent <i>n</i> .
		Timing Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
PCI_IDSEL	I	PCI initialization device select. Used as a chip select during a PCI configuration cycle in agent mode. This signal should be tied low in host mode.
		State Meaning Asserted—The PCI controller is being selected as a target of a configuration read or write transactions. Negated—The PCI controller is not being selected as a target of configuration read or write transactions.
		Timing Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
$\overline{\text{PCI_INTA}}$	O	PCI interrupt A.
		State Meaning Asserted—The PCI controller signals an interrupt to the PCI host. Negated—The PCI controller is not currently signalling an interrupt.
		Timing Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
$\overline{\text{PCI_IRDY}}$	I/O	PCI initiator ready. This signal is driven by the PCI controller when it is the initiator of a PCI transfer.
		O
		State Meaning Asserted—The PCI controller, acting as a PCI master, can complete the current data phase of a PCI transaction. During a write, this PCI controller asserts $\overline{\text{PCI_IRDY}}$ to indicate that valid data is present on $\text{PCI_AD}[31:0]$. During a read, this PCI controller asserts $\overline{\text{PCI_IRDY}}$ to indicate that it is prepared to accept data. Negated—The PCI target needs to wait before this PCI controller, acting as a PCI master, can complete the current data phase. During a write, this PCI controller negates $\overline{\text{PCI_IRDY}}$ to insert a wait cycle when it cannot provide valid data to the target. During a read, this PCI controller negates $\overline{\text{PCI_IRDY}}$ to insert a wait cycle when it cannot accept data from the target.
		Timing Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
	I	Inputs for the bi-directional initiator ready.
State Meaning Asserted—Another PCI master can complete the current data phase of a transaction. Negated—If $\overline{\text{PCI_FRAME}}$ is asserted, indicates a wait cycle from another master. If $\overline{\text{PCI_FRAME}}$ is negated, indicates that the PCI bus is idle.		

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_PAR	I/O	PCI parity.	
	O	Outputs for the bi-directional parity.	
		State Meaning	Asserted—Odd parity across PCI_AD[31:0] and PCI_CBE[3:0] during address and data phases. Negated—Even parity across PCI_AD[31:0] and PCI_AD[31:0] during address and data phases.
		Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
	I	Inputs for the bi-directional parity.	
		State Meaning	Asserted—Odd parity driven by another PCI master or the PCI target during address and data phases. Negated—Even parity driven by another PCI master or the PCI target during address and data phases.
Timing		Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>	
$\overline{\text{PCI_PERR}}$	I/O	PCI parity error	
	O	Outputs for the bi-directional parity error.	
		State Meaning	Asserted—The PCI controller, acting as a PCI agent, detected a data parity error. (driven by the PCI initiator on reads; driven by the PCI target on writes.) Negated—No error.
		Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
	I	Inputs for the bi-directional parity error.	
		State Meaning	Asserted—Another PCI agent detects a data parity error while this PCI controller is sourcing data (this PCI controller was acting as the PCI initiator during a write, or is acting as the PCI target during a read). Negated—No error.
Timing		Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>	
PCI_REQ0	I/O	PCI bus request. Input signal on this PCI controller when the arbiter is enabled. Output signal when the arbiter is disabled. Note that $\overline{\text{PCI_REQ}}_n$ is a point-to-point signal. Every master has its own bus request signal.	
	O	Outputs for the bi-directional bus request.	
		State Meaning	Asserted—The PCI controller is requesting control of the PCI bus to perform a transaction. Negated—The PCI controller does not require use of the PCI bus.
		Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
	I	Input for the bi-directional bus request.	
		State Meaning	Asserted—Agent 0 is requesting control of the PCI bus to perform a transaction. Negated—Agent 0 does not require use of the PCI bus.
Timing		Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>	

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
$\overline{\text{PCI_REQ}}[1:2]$	I	PCI bus request. Input signals on this PCI controller when the arbiter is enabled. Note that $\overline{\text{PCI_REQ}}[n]$ is a point-to-point signal. Every master has its own bus request signal. Following is the state meaning for the $\overline{\text{PCI_REQ}}[n]$ input.
	State Meaning	Asserted—An agent n is requesting control of the PCI bus to perform a transaction. Negated—An agent n does not require use of the PCI bus.
	Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
$\overline{\text{PCI_RESET_OUT}}$	O	PCI reset. This signal is used only in host mode. It should be left unconnected in agent mode.
	State Meaning	Asserted—Devices on the PCI bus are in reset. Negated—Devices on the PCI bus operate normally.
	Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
$\overline{\text{PCI_SERR}}$	I/O	PCI system error
	O	Outputs for the bi-directional system error.
	State Meaning	Asserted—An address parity error, a target-abort (when this PCI controller is acting as the initiator), or some other system error (where the result is a catastrophic error) was detected. Negated—No error.
	Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
	I	Inputs for the bi-directional system error.
	State Meaning	Asserted—A device (other than this PCI controller) has detected a catastrophic error. Negated—No error.
$\overline{\text{PCI_STOP}}$	I/O	PCI stop.
	O	Outputs for the bi-directional stop.
	State Meaning	Asserted—The PCI controller, acting as a PCI target, is requesting that the initiator stop the current transaction. Negated—The current transaction can continue.
	Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
	I	Inputs for the bi-directional stop.
	State Meaning	Asserted—A target is requesting that this PCI controller, as the initiator, stop the current transaction. Negated—The current transaction can continue.
Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>	

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{PCI_TRDY}}$	I/O	PCI target ready.	
	O	Outputs for the bi-directional target ready.	
		State Meaning	Asserted—The PCI controller, acting as a PCI target, can complete the current data phase of a PCI transaction. During a read, this PCI controller asserts $\overline{\text{PCI_TRDY}}$ to indicate that valid data is present on PCI_AD[31:0]. During a write, this PCI controller asserts $\overline{\text{PCI_TRDY}}$ to indicate that it is prepared to accept data. Negated—The PCI initiator needs to wait before this PCI controller, acting as a PCI target, can complete the current data phase. During a read, this PCI controller negates $\overline{\text{PCI_TRDY}}$ to insert a wait cycle when it cannot provide valid data to the initiator. During a write, this PCI controller negates $\overline{\text{PCI_TRDY}}$ to insert a wait cycle when it cannot accept data from the initiator.
		Timing	Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>
	I	Inputs for the bi-directional target ready.	
		State Meaning	Asserted—Another PCI target is able to complete the current data phase of a transaction. Negated—A wait cycle from another target.
Timing		Assertion/Negation—As specified by <i>PCI Local Bus Specification Rev 2.3</i>	

13.3 PCI Memory Map/Register Definitions

The PCI controller has the following types of registers:

- The PCI configuration access registers. Used for generating PCI configuration accesses from the CSB. These registers, listed in [Table 13-4](#), are memory-mapped on the CSB and accessed through the IMMR window.
- The PCI memory-mapped registers. Used to manage error functions, general control and status, and address translation control for the inbound path. These registers are shown in [Table 13-5](#). They can be accessed by PCI masters via the PCI controller to the CSB through the PIMMR inbound window. Note that [Table 13-5](#) does not list outbound address translation registers; these are contained in the I/O sequencer (IOS) memory-mapped registers. See [Chapter 12](#), “DMA/Messaging Unit,” for more information.
- The PCI configuration space registers. Defined by the PCI specification. These registers are accessed by PCI masters using configuration accesses and are described in [Section 13.3.3](#), “PCI Configuration Space Registers.”

Table 13-4. PCI Configuration Access Registers

Offset	Register	Access	Reset	Section/Page
PCI Configuration Access Registers—Block Base Address 0x0_8300				
0x00	PCI_CONFIG_ADDRESS	W	All zeros	13.3.1.1/13-13
0x04	PCI_CONFIG_DATA	R/W	All zeros	13.3.1.2/13-14
0x08	PCI_INT_ACK	R	N/A	13.3.1.3/13-15

Table 13-5. PCI Memory-Mapped Registers

Offset	Register	Access	Reset	Section/Page
PCI Controller—Block Base Address 0x0_8500				
PCI Error Management Registers				
0x00	PCI error status register (PCI_ESR)	w1c	All zeros	13.3.2.1/13-15
0x04	PCI error capture disable register (PCI_ECDR)	R/W	All zeros	13.3.2.2/13-16
0x08	PCI error enable register (PCI_EER)	R/W	All zeros	13.3.2.3/13-17
0x0C	PCI error attributes capture register (PCI_EATCR)	R/W	All zeros	13.3.2.4/13-18
0x10	PCI error address capture register (PCI_EACR)	R	All zeros	13.3.2.5/13-19
0x14	PCI error extended address capture register (PCI_EEACR)	R	All zeros	13.3.2.6/13-20
0x18	PCI error data capture register (PCI_EDCR)	R/W	All zeros	13.3.2.7/13-20
0x1C	Reserved	—	—	—
PCI Control and Status Registers				
0x20	PCI general control register (PCI_GCR)	R/W	All zeros	13.3.2.8/13-20
0x24	PCI error control register (PCI_ECR)	R/W	All zeros	13.3.2.9/13-21
0x28	PCI general status register (PCI_GSR)	R	All zeros	13.3.2.10/13-22
PCI Inbound ATU Registers				
0x38	PCI inbound translation address register 2 (PITAR2)	R/W	All zeros	13.3.2.11/13-22
0x3C	Reserved	—	—	—
0x40	PCI inbound base address register 2 (PIBAR2)	R/W	All zeros	13.3.2.12/13-23
0x44	PCI inbound extended base address register 2 (PIEBAR2)	R/W	All zeros	13.3.2.13/13-24
0x48	PCI inbound window attributes register 2 (PIWAR2)	R/W	All zeros	13.3.2.14/13-24
0x50	PCI inbound translation address register 1 (PITAR1)	R/W	All zeros	13.3.2.11/13-22
0x54	Reserved	—	—	—
0x58	PCI inbound base address register 1 (PIBAR1)	R/W	All zeros	13.3.2.12/13-23
0x5C	PCI inbound extended base address register 1 (PIEBAR1)	R/W	All zeros	13.3.2.13/13-24
0x60	PCI inbound window attributes register 1 (PIWAR1)	R/W	All zeros	13.3.2.14/13-24
0x68	PCI inbound translation address register 0 (PITAR0)	R/W	All zeros	13.3.2.11/13-22
0x6C	Reserved	—	—	—
0x70	PCI inbound base address register 0 (PIBAR0)	R/W	All zeros	13.3.2.12/13-23
0x78	PCI inbound window attributes register 0 (PIWAR0)	R/W	All zeros	13.3.2.13/13-24
0x7C–0xFF	Reserved	—	—	—

13.3.1 PCI Configuration Access Registers

This section describes the registers used to allow a local bus master to access the PCI configuration space, and generate special cycle or interrupt acknowledge transactions on the PCI bus. A special case provides access to the PCI controller’s internal PCI configuration registers. The PCI registers, PCI_CONFIG_ADDRESS, PCI_CONFIG_DATA, and PCI_INT_ACK, are little endian registers.

13.3.1.1 PCI_CONFIG_ADDRESS

Figure 13-3 shows the PCI_CONFIG_ADDRESS register fields.

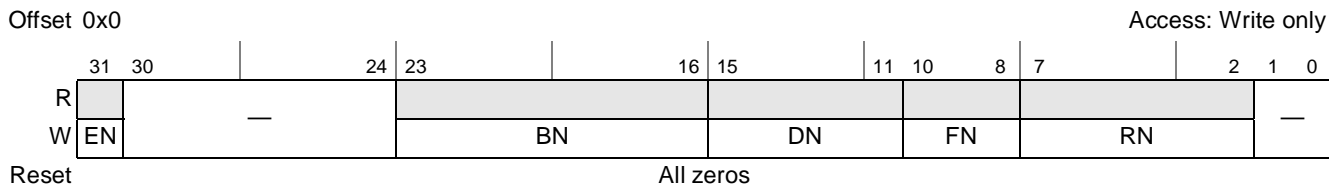


Figure 13-3. PCI_CONFIG_ADDRESS Register

The PCI_CONFIG_ADDRESS register holds the address for an access to the PCI configuration space from the local bus. This register must be programmed before accessing PCI_CONFIG_DATA to perform the transaction. Only 32-bit accesses are permitted.

If EN=1, BN=0, and DN=0, the access is to the internal PCI configuration registers, so no transaction is generated on the PCI bus.

If EN=1, BN=0, DN=31, FN=7, and RN=0, writing to PCI_CONFIG_DATA generates a special cycle transaction and reading from PCI_CONFIG_DATA generates an interrupt acknowledge transaction.

Table 13-6 shows the bit settings of the PCI_CONFIG_ADDRESS register.

Table 13-6. PCI_CONFIG_ADDRESS Field Descriptions

Bits	Name	Description
31	EN	Enable configuration transaction. Determines the type of transaction to be generated. 0 No configuration transaction will be generated by accessing the CONFIG_DATA register. Such an access will be passed through to the PCI bus as an I/O transaction. Since this is generally not desirable, the user should not access CONFIG_DATA when the EN bit is 0. 1 A configuration transaction will be generated by accessing the CONFIG_DATA register if BN and DN are not both zero.
30–24	—	Reserved
23–16	BN	Bus number. Specifies the bus segment to which a configuration transaction is directed. If this field is 0, a Type 0 configuration transaction is generated. Otherwise, a Type 1 configuration transaction is generated.

Table 13-6. PCI_CONFIG_ADDRESS Field Descriptions (continued)

Bits	Name	Description																																																		
15–11	DN	Device number. Specifies the device to which a configuration transaction is directed. For a Type 0 configuration transaction, this field is decoded to individual PCI1_IDSEL signals for the address phase according to the following values. For a Type 1 configuration transaction, this field is used directly for the address phase.																																																		
		<table border="0"> <thead> <tr> <th>Value</th> <th>AD Signal that is Driving High</th> </tr> </thead> <tbody> <tr><td>01010</td><td>31</td></tr> <tr><td>01011</td><td>11</td></tr> <tr><td>01100</td><td>12</td></tr> <tr><td>01101</td><td>13</td></tr> <tr><td>01110</td><td>14</td></tr> <tr><td>01111</td><td>15</td></tr> <tr><td>10000</td><td>16</td></tr> <tr><td>10001</td><td>17</td></tr> <tr><td>10010</td><td>18</td></tr> <tr><td>10011</td><td>19</td></tr> <tr><td>10100</td><td>20</td></tr> <tr><td>10101</td><td>21</td></tr> <tr><td>10110</td><td>22</td></tr> <tr><td>10111</td><td>23</td></tr> <tr><td>11000</td><td>24</td></tr> <tr><td>11001</td><td>25</td></tr> <tr><td>11010</td><td>26</td></tr> <tr><td>11011</td><td>27</td></tr> <tr><td>11100</td><td>28</td></tr> <tr><td>11101</td><td>29</td></tr> <tr><td>11110</td><td>30</td></tr> <tr><td>11111</td><td>Special cycle / interrupt acknowledge</td></tr> <tr><td>00000</td><td>Internal access</td></tr> <tr><td>Others</td><td>Reserved</td></tr> </tbody> </table>	Value	AD Signal that is Driving High	01010	31	01011	11	01100	12	01101	13	01110	14	01111	15	10000	16	10001	17	10010	18	10011	19	10100	20	10101	21	10110	22	10111	23	11000	24	11001	25	11010	26	11011	27	11100	28	11101	29	11110	30	11111	Special cycle / interrupt acknowledge	00000	Internal access	Others	Reserved
		Value	AD Signal that is Driving High																																																	
		01010	31																																																	
		01011	11																																																	
		01100	12																																																	
		01101	13																																																	
		01110	14																																																	
		01111	15																																																	
		10000	16																																																	
		10001	17																																																	
		10010	18																																																	
		10011	19																																																	
		10100	20																																																	
		10101	21																																																	
		10110	22																																																	
		10111	23																																																	
		11000	24																																																	
		11001	25																																																	
		11010	26																																																	
11011	27																																																			
11100	28																																																			
11101	29																																																			
11110	30																																																			
11111	Special cycle / interrupt acknowledge																																																			
00000	Internal access																																																			
Others	Reserved																																																			
10–8	FN	Function number. Specifies the function to which the configuration transaction is directed on a multi-function device. It is used directly in the address phase of the configuration transaction.																																																		
7–2	RN	Register number. Specifies the register being accessed in the PCI configuration space.																																																		
1–0	—	Reserved																																																		

13.3.1.2 PCI_CONFIG_DATA

An access to PCI_CONFIG_DATA usually generates a PCI configuration transaction if PCI_CONFIG_ADDRESS[EN] is set. There are some exceptions contained in the description of PCI_CONFIG_ADDRESS[EN].

This register may be accessed with an 8-, 16-, or 32-bit access, depending on the width of the register targeted by the configuration transaction.

Figure 13-4 shows the PCI_CONFIG_DATA register fields.

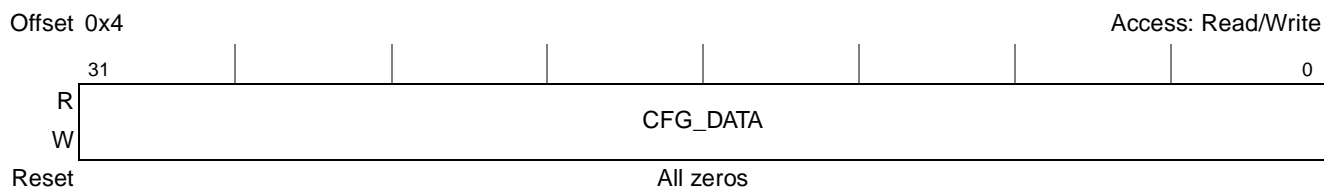


Figure 13-4. PCI_CONFIG_DATA

Table 13-7 shows the bit settings of the PCI_CONFIG_DATA register.

Table 13-7. PCI_CONFIG_DATA Field Descriptions

Bits	Name	Description
31–0	CFG_DATA	Configuration data. This field contains the data transferred on a PCI configuration transaction.

13.3.1.3 PCI Interrupt Acknowledge Register (PCI_INT_ACK)

Reading this register generates an interrupt acknowledge transaction on the PCI bus. The value that is read is undefined.

13.3.2 PCI Memory-Mapped Control and Status Registers

This section describes the control and status registers.

13.3.2.1 PCI Error Status Register (PCI_ESR)

The PCI error status register (PCI_ESR) contains status bits for various types of error conditions captured by the PCI controller. 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. Figure 13-5 shows the PCI_ESR fields.

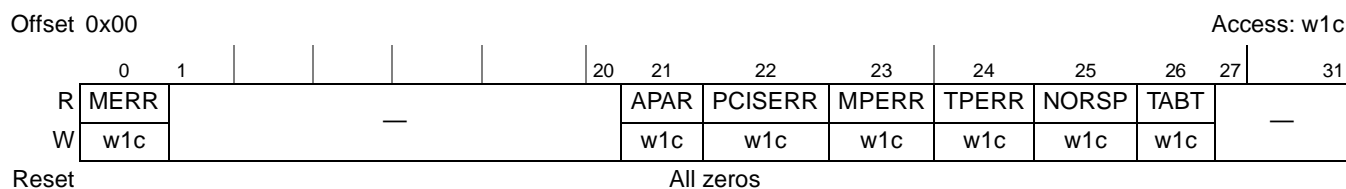


Figure 13-5. PCI Error Status Register (PCI_ESR)

Table 13-8 describes the bit settings of the PCI_ESR register.

Table 13-8. PCI_ESR Field Descriptions

Bits	Name	Description
0	MERR	Multiple errors. Set if any other bit of this register is 1 and the same error type occurs again.
1–20	—	Reserved
21	APAR	Address parity error. Set when there is an address parity error on a PCI access initiated by a device other than this PCI controller.
22	PCISERR	PCI system error. Set when the $\overline{\text{PCI_SERR}}$ input signal is asserted. See Table 13-3 for more information on $\overline{\text{PCI_SERR}}$.
23	MPERR	Master parity error. Set when the $\overline{\text{PCI_PERR}}$ input signal is asserted on a write access initiated by this PCI controller or when a data parity error is detected by this PCI controller on a read access that it initiated.
24	TPERR	Target parity error. Set when this PCI controller is the target of a transaction and the $\overline{\text{PCI_PERR}}$ input signal is asserted on a read access or a data parity error is detected by this PCI controller on a write access.
25	NORSP	No response. Set when there is no response to a transaction initiated by this PCI controller on the PCI bus (no $\overline{\text{PCI_DEVSEL}}$ assertion).
26	TABT	Target abort. Set when a PCI target abort occurs on a transaction initiated by this PCI controller.
27–31	—	Reserved

13.3.2.2 PCI Error Capture Disable Register (PCI_ECDR)

PCI_ECDR contains fields for controlling the capture of the transaction that caused an error. Each bit corresponds to the error condition reported in the PCI error status register (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.

- 1 = Do not capture the transaction that caused this error.
- 0 = Capture the transaction that caused this error.

Figure 13-6 shows the PCI_ECDR fields.

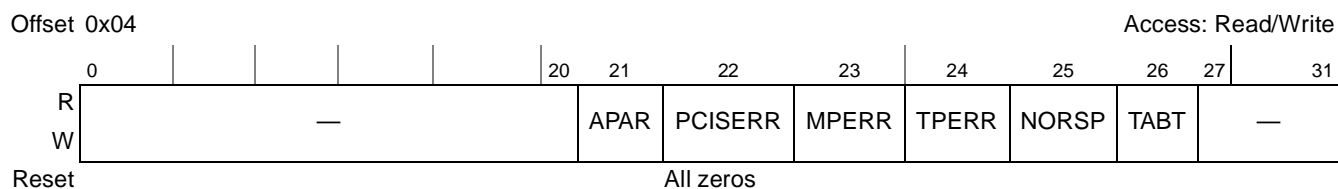


Figure 13-6. PCI Error Capture Disable Register (PCI_ECDR)

Table 13-9 describes the bit settings of the PCI_ECDR register.

Table 13-9. PCI_ECDR Field Descriptions

Bits	Name	Description
0–20	—	Reserved
21	APAR	Address parity error. Disable capture for address parity errors
22	PCISERR	PCI system error. Disable capture for received $\overline{\text{PCI_SERR}}$ errors
23	MPERR	Master parity error. Disable capture for master $\overline{\text{PCI_PERR}}$ errors
24	TPERR	Target parity error. Disable capture for target $\overline{\text{PCI_PERR}}$ errors
25	NORSP	No response. Disable capture for master-abort errors
26	TABT	Target abort. Disable capture for target abort errors
27–31	—	Reserved

13.3.2.3 PCI Error Enable Register (PCI_EER)

PCI_EER contains fields for enabling the assertion of an interrupt for the error conditions reported in the PCI error status register (PCI_ESR).

- 1 = The interrupt is enabled.
- 0 = The interrupt is disabled.

Figure 13-7 shows the PCI_EER fields.

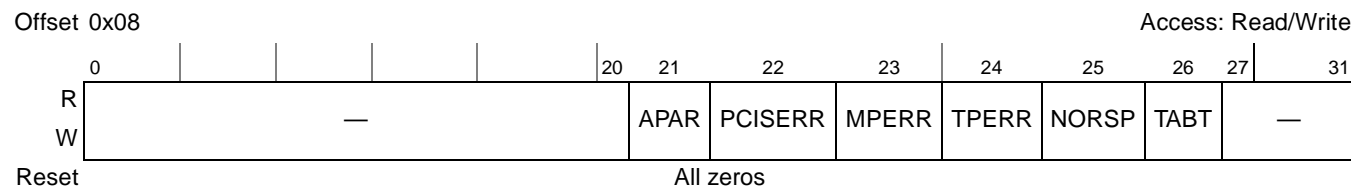


Figure 13-7. PCI Error Enable Register (PCI_EER)

Table 13-10 describes the bit settings of the PCI_EER register.

Table 13-10. PCI_EER Field Descriptions

Bits	Name	Description
0–20	—	Reserved
21	APAR	Address parity error. Generate an interrupt when the corresponding bit of the PCI_ESR is 1.
22	PCISERR	PCI system error. Generate an interrupt when the corresponding bit of the PCI_ESR is 1.
23	MPERR	Master parity error. Generate an interrupt when the corresponding bit of the PCI_ESR is 1.
24	TPERR	Target parity error. Generate an interrupt when the corresponding bit of the PCI_ESR is 1.
25	NORSP	No response. Generate an interrupt when the corresponding bit of the PCI_ESR is 1.

Table 13-10. PCI_EER Field Descriptions (continued)

Bits	Name	Description
26	TABT	Target abort. Generate an interrupt when the corresponding bit of the PCI_ESR is 1.
27–31	—	Reserved

13.3.2.4 PCI Error Attributes Capture Register (PCI_EATCR)

PCI_EATCR contains fields for storing information associated with the first PCI error captured. Figure 13-8 shows the PCI_EATCR fields.

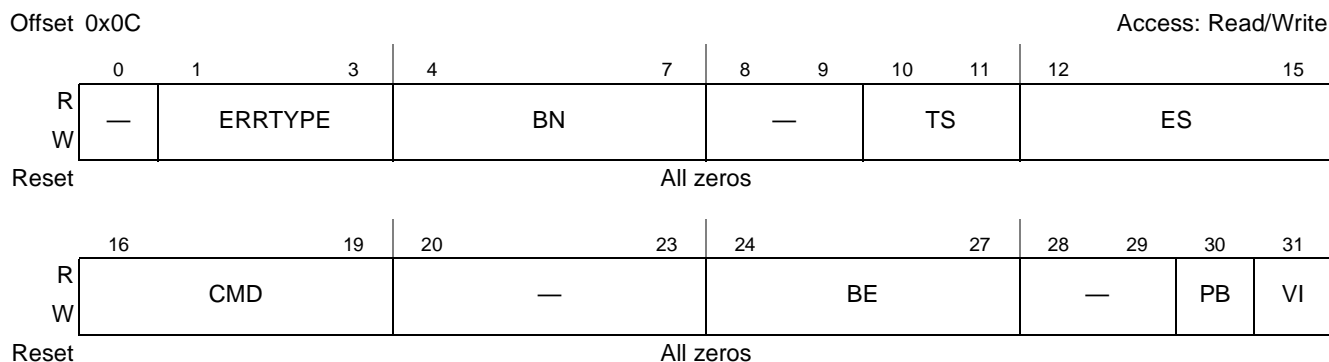


Figure 13-8. PCI Error Attributes Capture Register (PCI_EATCR)

Table 13-11 describes the bit settings of the PCI_EATCR register.

Table 13-11. PCI_EATCR Field Descriptions

Bits	Name	Description
0	—	Reserved
1–3	ERRTYPE	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 Master 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
4–7	BN	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. The beat values are described as follows: 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

Table 13-11. PCI_EATCR Field Descriptions (continued)

Bits	Name	Description
8–9	—	Reserved
10–11	TS	Transaction size. Indicates 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 double words in the cache line on which the error occurred. This field is valid only if the PCI controller was the master of the transaction. 00 4 double words 01 1 double word 10 2 double words 11 3 double words
12–15	ES	Error source. This field indicates the source of the PCI transaction. 0000 External master 0101 DMA Others reserved
16–19	CMD	PCI command. Contains the PCI command PCI_CBE[3:0] of the transaction.
20–23	—	Reserved
24–27	BE	PCI byte enables. Contains the PCI byte enables PCI_CBE[3:0] for the data word.
28–29	—	Reserved
30	PB	Parity bit. Contains the PCI parity bit for the captured data word.
31	VI	Error information valid. This 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

13.3.2.5 PCI Error Address Capture Register (PCI_EACR)

PCI_EACR contains fields for storing the low portion of the address associated with the first PCI error captured. [Figure 13-9](#) shows the PCI_EACR fields.

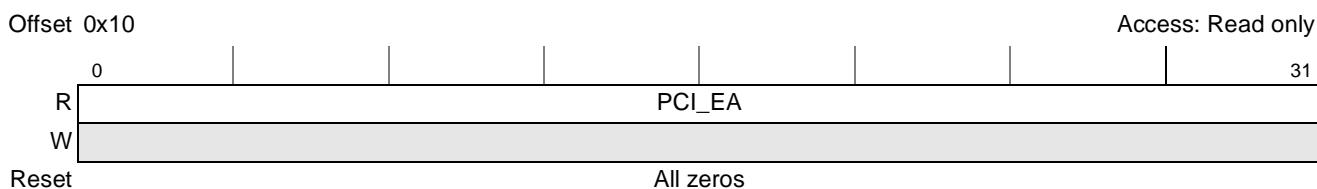


Figure 13-9. PCI Error Address Capture Register (PCI_EACR)

[Table 13-12](#) describes the bit settings of the PCI_EACR register.

Table 13-12. PCI_EACR Field Description

Bits	Name	Description
0–31	PCI_EA	PCI error address. Contains the low portion of the address associated with the first detected error. Read only.

13.3.2.6 PCI Error Extended Address Capture Register (PCI_EEACR)

PCI_EEACR contains fields for storing the high portion of the address associated with the first PCI error captured. [Figure 13-10](#) shows the PCI_EEACR fields.

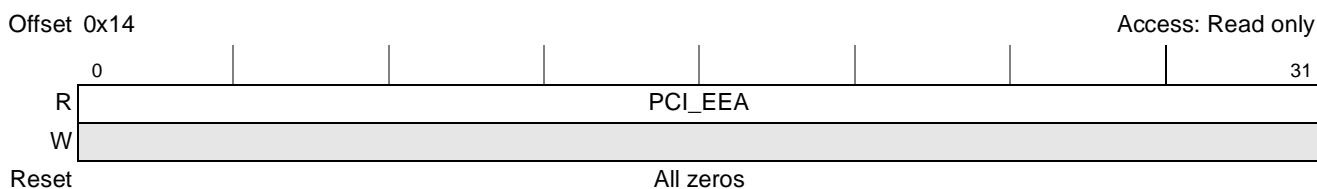


Figure 13-10. PCI Error Extended Address Capture Register (PCI_EEACR)

[Table 13-13](#) describes the bit settings of the PCI_EEACR register.

Table 13-13. PCI_EEACR Field Description

Bits	Name	Description
0–31	PCI_EEA	PCI error extended address. Contains the high portion of the address associated with the first detected error.

13.3.2.7 PCI Error Data Low Capture Register (PCI_EDLCR)

PCI_EDLCR contains fields for storing the data associated with the first PCI error captured. [Figure 13-11](#) shows the PCI_EDLCR fields.



Figure 13-11. PCI Error Data Low Capture Register (PCI_EDLCR)

[Table 13-14](#) describes the bit settings of the PCI_EDLCR register.

Table 13-14. PCI_EDLCR Field Description

Bits	Name	Description
0–31	PCI_EDR	PCI error data. Contains the data associated with the first detected error.

13.3.2.8 PCI General Control Register (PCI_GCR)

PCI_GCR contains fields for controlling the behavior of the internal arbiter, the state of the bus signals, and the PCI reset signal for host mode. [Figure 13-12](#) shows the PCI_GCR fields.

Table 13-16 describes the bit settings of the PCI_ESR register.

Table 13-16. PCI_ESR Field Descriptions

Bits	Name	Description
0–20	—	Reserved
21	APAR	0 An interrupt is generated if the corresponding bit of the PCI_ESR is 1. 1 A machine check is generated if the corresponding bit of the PCI_ESR is 1.
22	PCISERR	0 An interrupt is generated if the corresponding bit of the PCI_ESR is 1. 1 A machine check is generated if the corresponding bit of the PCI_ESR is 1.
23	MPERR	0 An interrupt is generated if the corresponding bit of the PCI_ESR is 1. 1 A machine check is generated if the corresponding bit of the PCI_ESR is 1.
24	TPERR	0 An interrupt is generated if the corresponding bit of the PCI_ESR is 1. 1 A machine check is generated if the corresponding bit of the PCI_ESR is 1.
25	NORSP	0 An interrupt is generated if the corresponding bit of the PCI_ESR is 1. 1 A machine check is generated if the corresponding bit of the PCI_ESR is 1.
26	TABT	0 An interrupt is generated if the corresponding bit of the PCI_ESR is 1. 1 A machine check is generated if the corresponding bit of the PCI_ESR is 1.
27–31	—	Reserved

13.3.2.10 PCI General Status Register (PCI_GSR)

PCI_GSR contains fields for providing status information, shown in Figure 13-14.

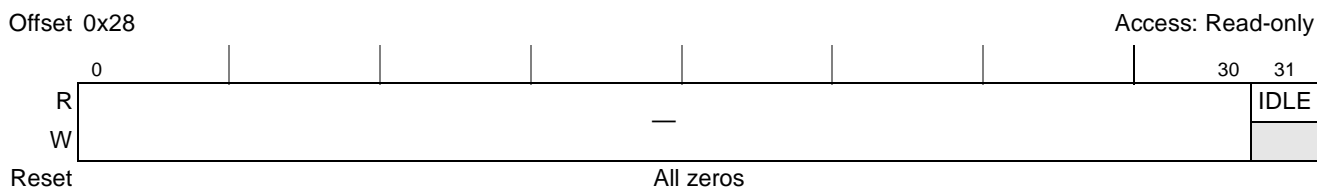


Figure 13-14. PCI General Status Register (PCI_GSR)

Table 13-17 shows the bit settings of the PCI_GSR register. All bits are read-only.

Table 13-17. PCI_GSR Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	IDLE	PCI controller is idle. Indicates when the PCI bus is totally idle before setting PCI_GCR[PPL]. 0 The PCI controller is active. 1 The PCI controller is idle.

13.3.2.11 PCI Inbound Translation Address Registers (PITAR_n)

PITAR_n contains fields for defining the starting point of the inbound translation windows in the local memory space (see Section 13.4.6, “PCI Inbound Address Translation”; for more information on outbound address translation registers, see Chapter 11, “Sequencer”). Inbound and outbound windows for the same bus should not overlap. Therefore, situations where an inbound window translation points back into an

outbound window, or where an outbound translation window points back into an inbound window, are not allowed.

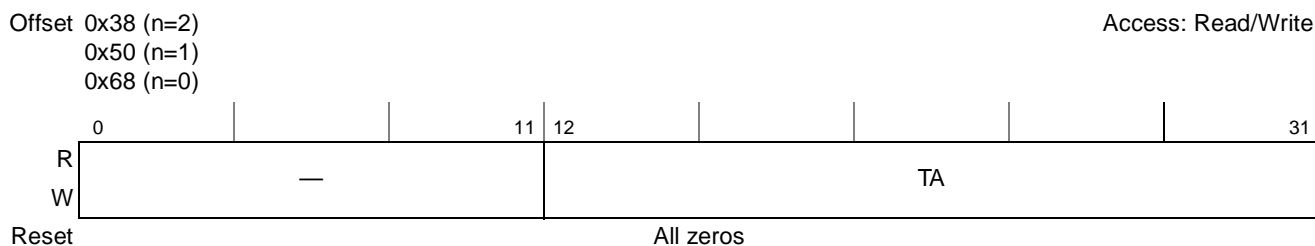


Figure 13-15. PCI Inbound Translation Address Registers (PITAR n)

Table 13-18 shows the bit settings of PITAR n .

Table 13-18. PITAR n Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	TA	Translation address. Contains the starting address of the inbound translated address. TA corresponds to the 20 highest-order bits of a 32-bit local address. The specified address must be aligned to the window size, as defined by PIWAR n [IWS].

13.3.2.12 PCI Inbound Base Address Registers (PIBAR n)

PIBAR n contains fields for defining the starting point of the inbound windows in the PCI memory space. A write to a PIBAR n register also causes a change in the base address bits in the corresponding GPL base address register in the PCI configuration space. Figure 13-16 shows the PIBAR x fields.



Figure 13-16. PCI Inbound Base Address Registers (PIBAR n)

Table 13-19 shows the bit settings of PIBAR n .

Table 13-19. PIBAR n Field Descriptions

Bits	Name	Description
0–31	BA	Base address. Contains the starting address in the PCI memory space of the inbound window. This field corresponds to bits 43–12 of a 64-bit address. In PIBAR0, the upper 12 bits are reserved because only a 32-bit address is supported. The specified address must be aligned to the window size, as defined by PIWAR n [IWS].

13.3.2.13 PCI Inbound Extended Base Address Registers (PIEBAR_n)

PIEBAR_n contains fields for defining the high portion of the starting point of the inbound windows in the PCI memory space. Figure 13-17 shows the PIEBAR_n fields.



Figure 13-17. PCI Inbound Extended Base Address Registers (PIEBAR_n)

Table 13-20 shows the bit settings of PIEBAR_n.

Table 13-20. PIEBAR_n Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	EBA	Extended base address. Contains the high portion of the starting address in the PCI memory space of the inbound base address. This 20-bit field corresponds to bits 63–44 of a 64-bit address.

13.3.2.14 PCI Inbound Window Attribute Registers (PIWAR_n)

PIWAR_n contains fields for defining the size of an inbound translation window. It also defines some properties of the window. Figure 13-18 shows the PIWAR_n fields. See Section 4.3.1.1, “Reset Configuration Word Source,” for more information on reset configuration.

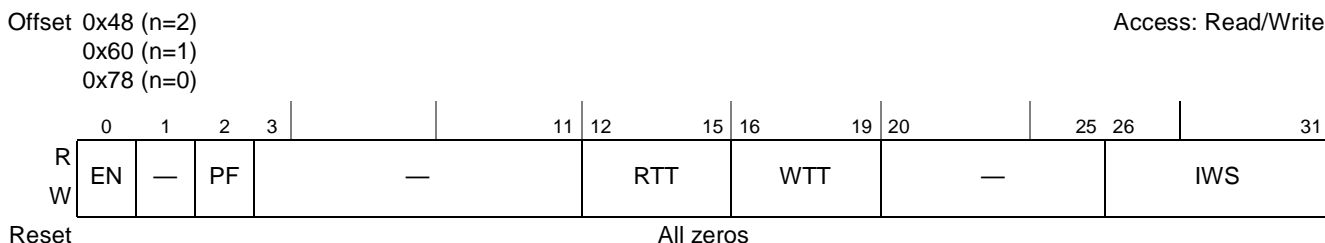


Figure 13-18. PCI Inbound Window Attribute Registers (PIWAR_n)

Table 13-21 shows the bit settings of PIWAR_n.

Table 13-21. PIWAR_n Field Descriptions

Bits	Name	Description
0	EN	Enable. Used to enable the address translation window. 0 Address translation is disabled for this window. 1 Address translation is enabled for this window. PCI addresses that match the definition of the window will be recognized by the PCI controller and translated to the local memory space.
1	—	Reserved

Table 13-21. PIWAR_n Field Descriptions (continued)

Bits	Name	Description
2	PF	Prefetchable. Defines whether the transactions that are translated through this window are prefetchable on the local bus. Streaming the transactions requires the memory space to be prefetchable. 0 Not prefetchable 1 Prefetchable
3–11	—	Reserved
12–15	RTT	Read transaction type. Determines the type of transaction performed on the local bus when the PCI transaction is a read. The RTT values are described as follows: 0100 Read without snoop on system bus 0101 Read with snoop on system bus Others reserved
16–19	WTT	Write transaction type. Determines the type of transaction performed on the local bus when the PCI transaction is a write. The WTT values are described as follows: 0100 Write without snoop of local processor 0101 Write with snoop of local processor Others reserved
20–25	—	Reserved
26–31	IWS	Inbound window size. Indicates 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) 000000–001010 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011110 2-Gbyte window size 011111–111111 Reserved

13.3.3 PCI Configuration Space Registers

This section describes the PCI configuration space registers. These registers are shown with descending bit numbering to correspond to the PCI standard.

NOTE

The registers described in this section use little-endian byte ordering. Software running on the local processor in big-endian mode must byte-swap the data. No byte swapping occurs when the registers are accessed from the PCI bus.

Table 13-22 shows the PCI configuration registers that are mapped in PCI configuration space. Some fields are common to registers in both spaces to ensure consistency. These fields are discussed in the register definitions.

Table 13-22. PCI Configuration Space Registers

Address	Use	Access
00	Vendor ID configuration register	R
02	Device ID configuration register	R
04	PCI command configuration register	R/W
06	PCI status configuration register	Read/bit-reset
08	Revision ID configuration register	R
09	Standard programming interface	R
0A	Subclass code configuration register	R
0B	Base class code configuration register	R
0C	Cache line size configuration register	R/W
0D	Latency timer configuration register	R/W
0E	Header type configuration register	R
0F	BIST control configuration register	R
10	PIMMR base address register	R/W
14	GPL base address register 0	R/W
18	GPL base address register 1	R/W
1C	GPL extended base address register 1	R/W
20	GPL base address register 2	R/W
24	GPL extended base address register 2	R/W
2C	Subsystem vendor ID configuration register	R
2E	Subsystem device ID configuration register	R
34	Capabilities pointer configuration register	R
3C	Interrupt line configuration register	R/W
3D	Interrupt pin configuration register	R
3E	Minimum grant configuration register	R
3F	Maximum latency configuration register	R
44	PCI function configuration register	R/W
46	PCI arbiter control register (PCIACR)	R/W
48	Hot swap register block	R/W

13.3.3.1 Vendor ID Configuration Register

Figure 13-19 shows the vendor ID fields. This is a read-only register.

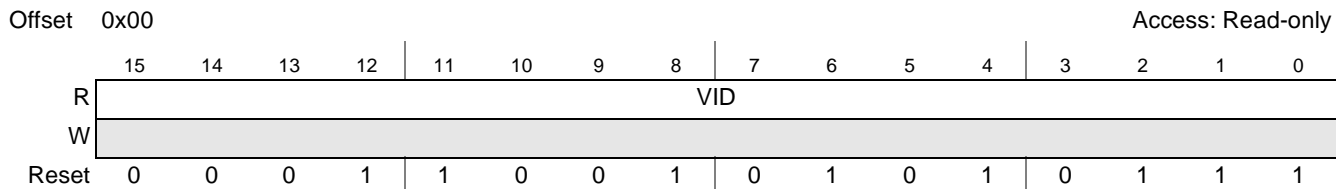


Figure 13-19. Vendor ID Configuration Register

Table 13-23 shows the bit settings of the vendor ID register.

Table 13-23. Vendor ID Configuration Register Field Descriptions

Bits	Name	Description
15-0	VID	Vendor ID. The read-only value 0x1957 specifies Freescale Semiconductor as the manufacturer of the device.

13.3.3.2 Device ID Configuration Register

Figure 13-20 shows the device ID fields. This is a read only register.

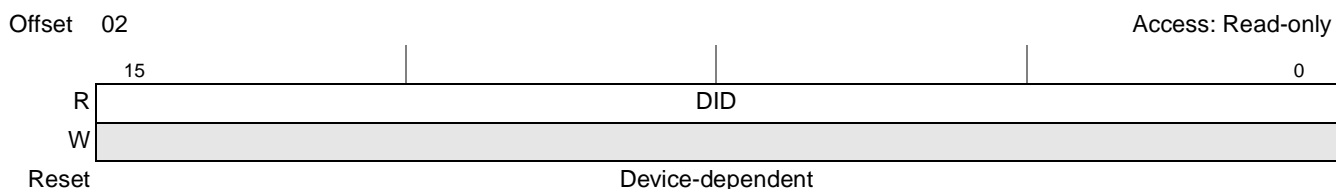


Figure 13-20. Device ID Configuration Register

Table 13-24 shows the bit settings of the device ID register.

Table 13-24. Device ID Configuration Register Field Descriptions

Bits	Name	Description
15-0	DID	Device ID. This field identifies the device. 0091 MPC8360 TBGA 0090 MPC8360E TBGA 0093 MPC8358 TBGA 0092 MPC8358E TBGA 0097 MPC8358 PBGA 0096 MPC8358E PBGA

13.3.3.3 PCI Command Configuration Register

Figure 13-21 shows the PCI command fields.

Offset 04

Access: Mixed

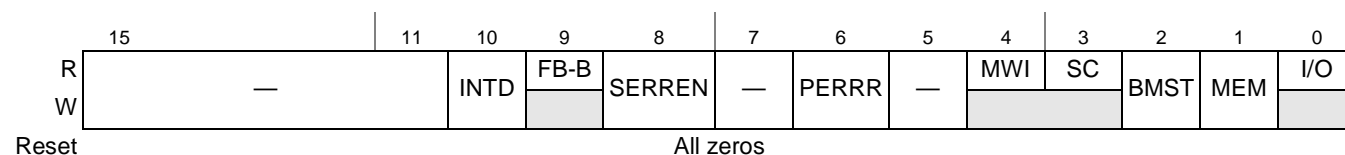


Figure 13-21. PCI Command Configuration Register

Table 13-25 shows the bit settings of the PCI command register.

Table 13-25. PCI Command Configuration Register Field Descriptions

Bits	Name	Description
15–11	—	Reserved
10	INTD	Interrupt Disable. Setting this bit masks the $\overline{\text{PCI_INTA}}$ output. 0 $\overline{\text{PCI_INTA}}$ provides the device interrupt status. 1 $\overline{\text{PCI_INTA}}$ is always negated.
9	FB-B	Fast back-to-back. Hard-wired to 0.
8	SERREN	SERR enable. This bit is an enable bit for the SERR driver. Address parity errors are reported only if this bit and bit 6 are 1. 0 $\overline{\text{PCI_SERR}}$ is never asserted. 1 $\overline{\text{PCI_SERR}}$ may be asserted to indicate error conditions.
7	—	Reserved
6	PERRR	Parity error response. Controls the PCI controller's response to a parity error. 0 Parity errors are ignored and normal operation continues. 1 Standard parity error treatment.
5	—	Reserved
4	MWI	Memory-write-and-invalidate. Hard-wired to 0.
3	SC	Special cycles. Hard-wired to 0.
2	BMST	Bus master. Controls the PCI controller's ability to be a master on the PCI bus. At reset, this bit is cleared in Agent Mode and set in Host Mode. 0 The PCI controller does not generate PCI accesses. 1 The PCI controller behaves as a bus master.
1	MEM	Memory space. Controls the response to memory space accesses. 0 The PCI controller does not respond to Memory Space accesses. 1 The PCI controller as a target responds to Memory Space accesses.
0	I/O	I/O space. Hard-wired to 0.

13.3.3.4 PCI Status Configuration Register

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 PCI controller. Other bits can be cleared by writing 1 to the bit location. Figure 13-22 shows the PCI status fields.

Offset 06											Access: Mixed				
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	0
R	DPERR	SSERR	RMA	RTA	STA	DEVSEL_T		DPD	FB-BC	—	66M	CL	INTS	—	
W	w1c	w1c	w1c	w1c	w1c			w1c					w1c		
Reset	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0

Figure 13-22. PCI Status Configuration Register

Table 13-26 shows the bit settings of the PCI status register.

Table 13-26. PCI Status Configuration Register Field Descriptions

Bits	Name	Description
15	DPERR	Detected parity error. Set whenever the PCI controller 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).
14	SSERR	Signaled system error. Set whenever $\overline{\text{PCI_SERR}}$ is asserted.
13	RMA	Received master abort. Set whenever the PCI controller, acting as the PCI master on the PCI bus, terminates a transaction (except for a special-cycle) using master-abort.
12	RTA	Received target abort. Set whenever a transaction initiated by this PCI controller on the PCI bus is terminated by a target-abort.
11	STA	Signaled target abort. Set whenever the PCI controller, acting as the PCI target on the PCI bus, issues a target-abort to a PCI master.
10–9	DEVSEL_T	DEVSEL timing. Hard-wired to 00.
8	DPD	Master data parity error. Set when a data parity error is detected on the PCI bus, if the PCI controller is the master that initiated the transaction and bit 6 in the PCI command register is set.
7	FB-BC	Fast back-to-back capable. Hard-wired to 1.
6	—	Reserved
5	66M	66-MHz capable. Hard-wired to 1.
4	CL	Capabilities list. Hard-wired to 1.
3	INTS	Interrupt status. Contains the status of the device interrupt. The value of this bit is not affected by the INTD bit of the PCI command configuration register.
2–0	—	Reserved

13.3.3.5 Revision ID Configuration Register

Figure 13-23 shows the revision ID fields.

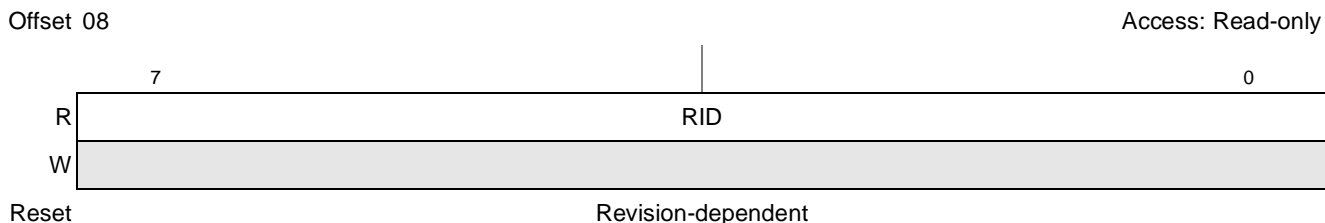


Figure 13-23. Revision ID Configuration Register

Table 13-27 shows the bit settings of the revision ID register.

Table 13-27. Revision ID Configuration Register Field Descriptions

Bits	Name	Description
7–0	RID	Revision ID. Specifies a revision code of the PCI controller.

13.3.3.6 Standard Programming Interface Configuration Register

Figure 13-24 shows the standard programming interface fields. This is the lower byte of the class code.

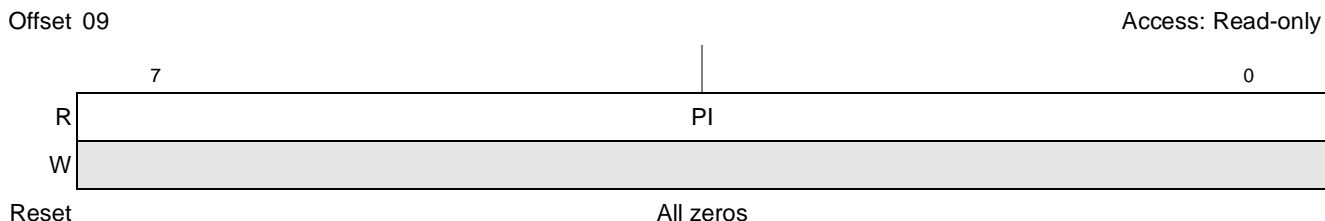


Figure 13-24. Standard Programming Interface Configuration Register

Table 13-28 shows the bit settings of the standard programming interface register.

Table 13-28. Standard Programming Interface Configuration Register Field Descriptions

Bits	Name	Description
7–0	PI	Programming interface. This field is hard-wired to 0x00.

13.3.3.7 Subclass Code Configuration Register

Figure 13-25 shows the subclass code fields. This is the middle byte of the class code.

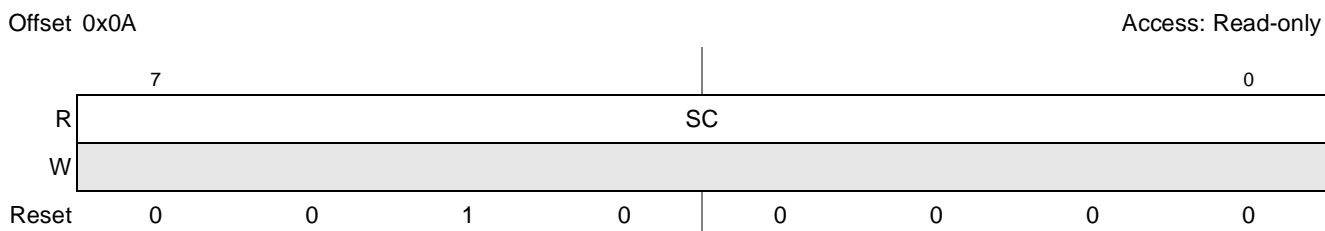


Figure 13-25. Subclass Code Configuration Register

Table 13-29 shows the bit settings of the subclass code register.

Table 13-29. Subclass Code Configuration Register Field Descriptions

Bits	Name	Description
7-0	SC	Sub-class code. This field is hard-wired to 0x20, indicating a Power PC processor.

13.3.3.8 Base Class Code Configuration Register

Figure 13-26 shows the base class code fields. This is the upper byte of the class code.

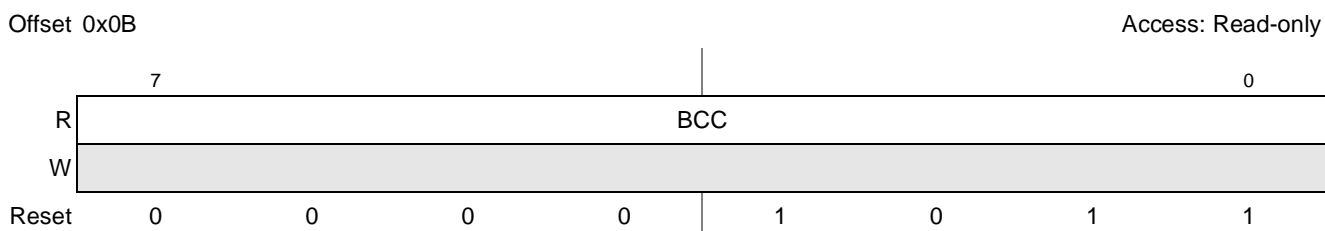


Figure 13-26. Base Class Code Configuration Register

Table 13-30 shows the bit settings of the class code register.

Table 13-30. Class Code Configuration Register Field Descriptions

Bits	Name	Description
7-0	BCC	Base class code. This field is hard-wired to 0x0B, indicating a processor.

13.3.3.9 Cache Line Size Configuration Register

Figure 13-27 shows the cache line size fields.

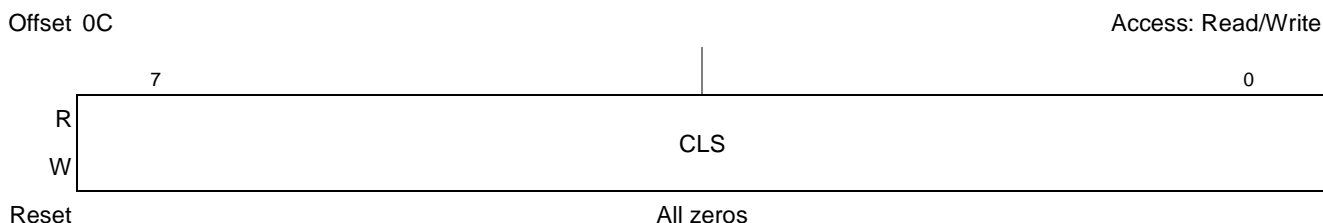


Figure 13-27. Cache Line Size Configuration Register

Table 13-31 shows the bit settings of the cache line size register.

Table 13-31. Cache Line Size Configuration Register Field Descriptions

Bits	Name	Description
7–0	CLS	Cache line size. Cache-line in terms of 32-bit words. Although the register is writable, only the value 0x08 is legal.

13.3.3.10 Latency Timer Configuration Register

Figure 13-28 shows the latency timer fields.

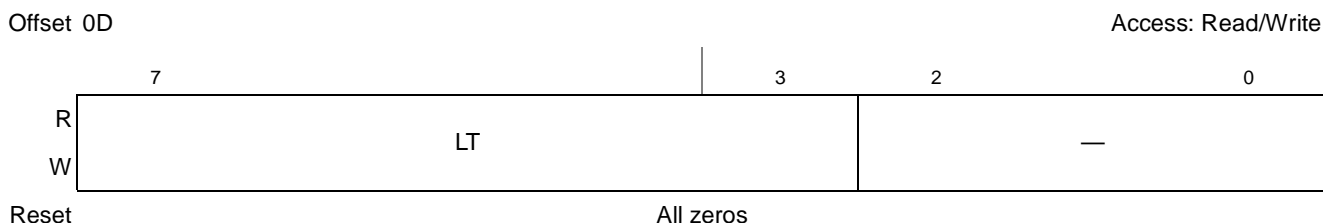


Figure 13-28. Latency Timer Configuration Register

Table 13-32 shows the bit settings of the latency timer register.

Table 13-32. Latency Timer Configuration Register Field Descriptions

Bits	Name	Description
7–3	LT	Latency timer. Specifies a granularity of 8 PCI clocks, the length of time that the PCI controller, when mastering a transaction, may hold the bus as the result of a bus grant. Refer to the PCI 2.3 specification for the rules by which the PCI controller completes transactions when the timer has expired.
2–0	—	Reserved

13.3.3.11 Header Type Configuration Register

Figure 13-29 shows the read-only header type register, which is hard-wired to 0x00.

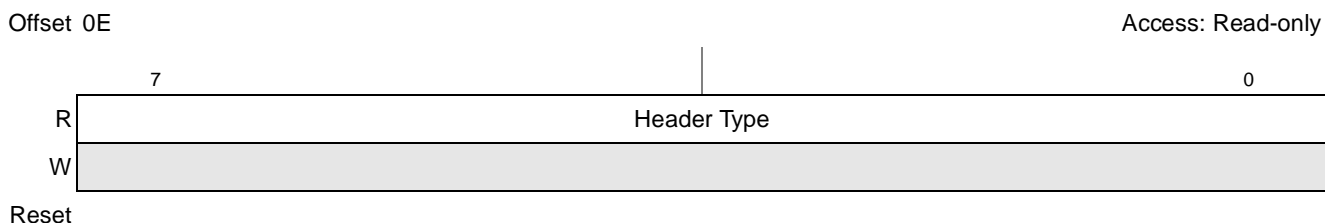


Figure 13-29. Header Type Configuration Register

13.3.3.12 BIST Control Configuration Register

Figure 13-30 shows the read-only BIST control register, which is hard-wired to 0x00.

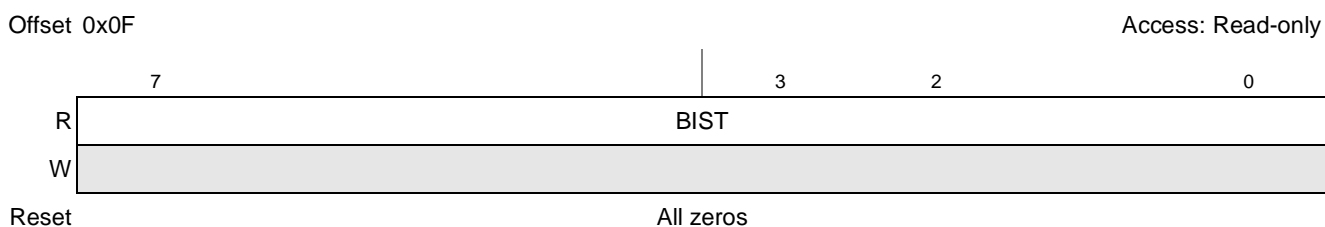


Figure 13-30. BIST Control Configuration Register

13.3.3.13 PIMMR Base Address Configuration Register

Figure 13-31 shows the PIMMR base address register fields.

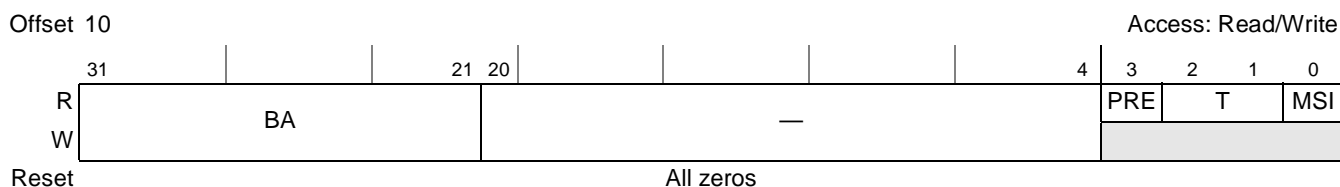


Figure 13-31. PIMMR Base Address Configuration Register

Table 13-33 shows the bit settings of the PIMMR base address register.

Table 13-33. PIMMR Base Address Configuration Register Field Descriptions

Bits	Name	Description
31–21	BA	Base address. Defines the base address for the internal (on-chip) memory-mapped register space. The size of this space is 2 Mbytes.
20–4	—	Reserved
3	PRE	Prefetchable. Hard-wired to 0.

Table 13-33. PIMMR Base Address Configuration Register Field Descriptions (continued)

Bits	Name	Description
2–1	T	Type. Hard-wired to 00.
0	MSI	Memory space indicator. Hard-wired to 0

13.3.3.14 GPL Base Address Register 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 will not change the bits that are masked by the IWS field. For read operation these masked bits will always return zeros.

Figure 13-32 shows the GPL base address register 0 fields.



Figure 13-32. GPL Base Address Register 0

Table 13-34 shows the bit settings of the GPL base address register 0.

Table 13-34. GPL Base Address Register 0 Field Descriptions

Bits	Name	Description
31–12	BA	Base address. Defines the base address for the inbound window. Bits 11–4 are hard-wired to 0 since the minimum window size is 4 Kbytes.
3	PRE	Prefetchable. This bit is read-only and contains the value of the PF bit in PIWAR0.
2–1	T	Type. Hard-wired to 00.
0	MSI	Memory space indicator. Hard-wired to 0

13.3.3.15 GPL Base Address Registers 1–2

The general purpose local access base address registers are provided to allow access to local memory space. These registers are closely tied to PIBAR_n and PIWAR_n 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_n in the corresponding PIBAR_n. Note that this write operation will not change the bits that are masked by the IWS field. For read operations, these masked bits always return zeros.

Figure 13-33 shows the GPL base address register 1–2 fields.

13.3.3.17 Subsystem Vendor ID Configuration Register

Figure 13-35 shows the subsystem vendor ID fields. The subsystem vendor ID configuration register is read-only from the PCI bus, but it can be programmed from the CSB.

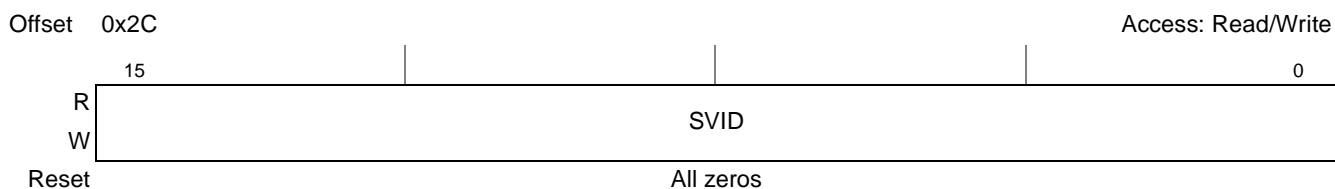


Figure 13-35. Subsystem Vendor ID Configuration Register

Table 13-37 shows the bit settings of the subsystem vendor ID configuration register.

Table 13-37. Subsystem Vendor ID Configuration Register Field Descriptions

Bits	Name	Description
15–0	SVID	Subsystem vendor ID. Identifies the manufacturer of the board or subsystem that contains this device.

13.3.3.18 Subsystem Device ID Configuration Register

Figure 13-36 shows the subsystem device configuration register ID fields. The subsystem device ID configuration register is read-only from the PCI bus, but it can be programmed from the CSB.

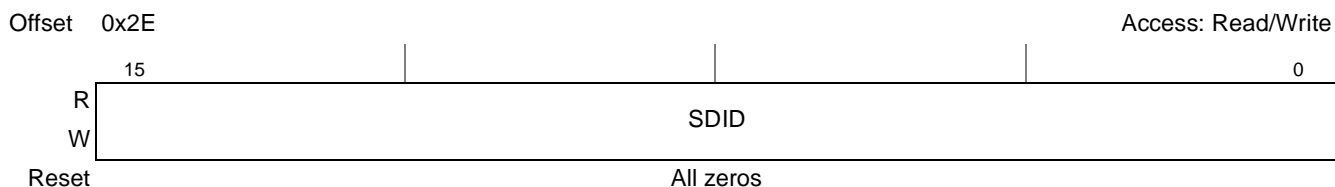


Figure 13-36. Subsystem Device ID Configuration Register

Table 13-38 shows the bit settings of the subsystem device ID configuration register.

Table 13-38. Subsystem Device ID Configuration Register Field Descriptions

Bits	Name	Description
15–0	SDID	Subsystem device ID. This field identifies the board or subsystem that contains this device.

13.3.3.19 Capabilities Pointer Configuration Register

The capabilities pointer register specifies the byte offset in the PCI configuration space that contains the first item in the capabilities list. Figure 13-37 shows the capabilities pointer configuration register fields.

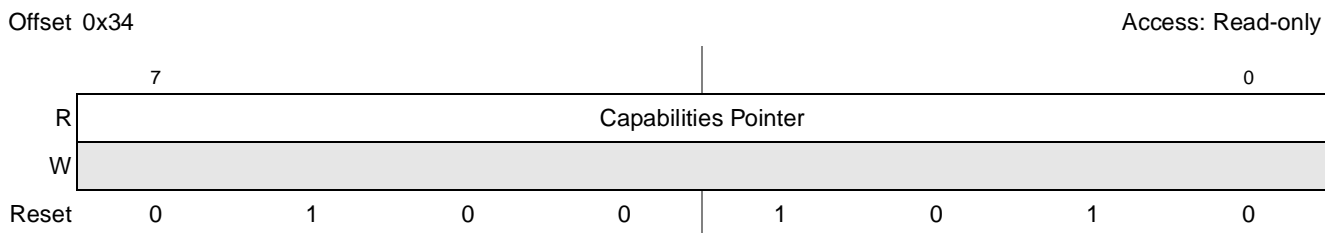


Figure 13-37. Capabilities Pointer Configuration Register

13.3.3.20 Interrupt Line Configuration Register

Figure 13-38 shows the interrupt line configuration register fields.

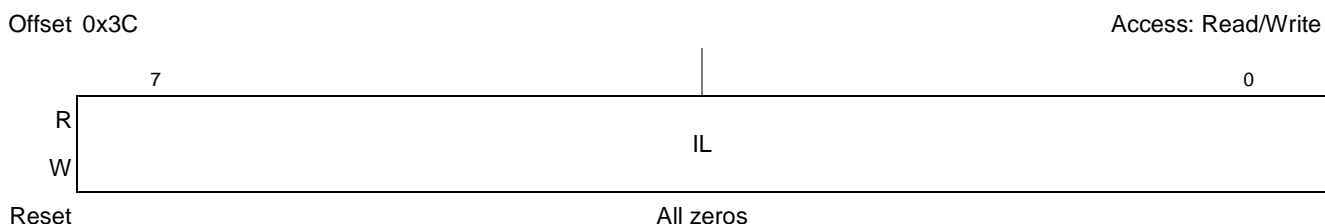


Figure 13-38. Interrupt Line Configuration Register

Table 13-39 shows the bit settings of the interrupt line configuration register.

Table 13-39. Interrupt Line Configuration Register Field Descriptions

Bits	Name	Description
7-0	IL	Interrupt line. Used to communicate interrupt line routing information. The value has no effect on the operation of the PCI controller.

13.3.3.21 Interrupt Pin Configuration Register

The interrupt pin configuration register tells which interrupt pin is used (0x01 means PCI_INTA).

Figure 13-39 shows the interrupt pin configuration register fields.

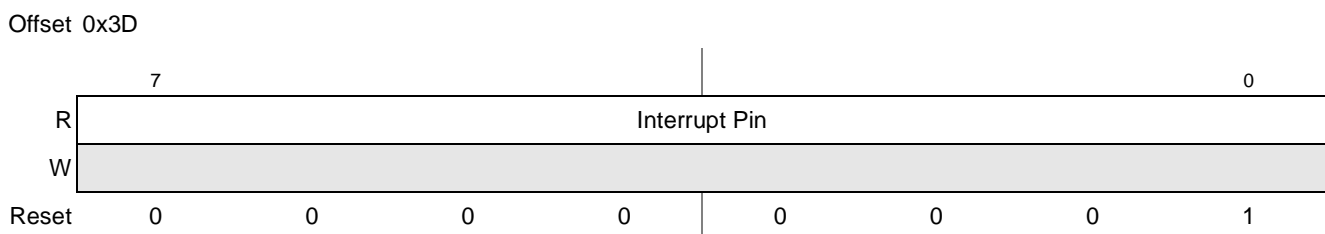


Figure 13-39. Interrupt Pin Register

13.3.3.22 Minimum Grant Configuration Register

Figure 13-40 shows the minimum grant configuration register fields.

PCI Bus Interface

Offset 0x3E

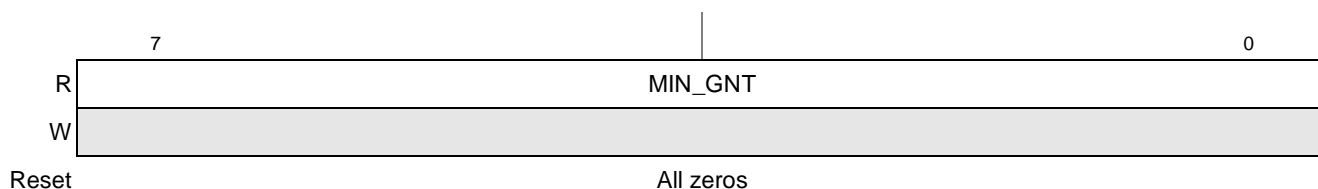


Figure 13-40. Minimum Grant Configuration Register

13.3.3.23 Maximum Latency Configuration Register

Figure 13-41 shows the maximum latency configuration register fields.

Offset 0x3F

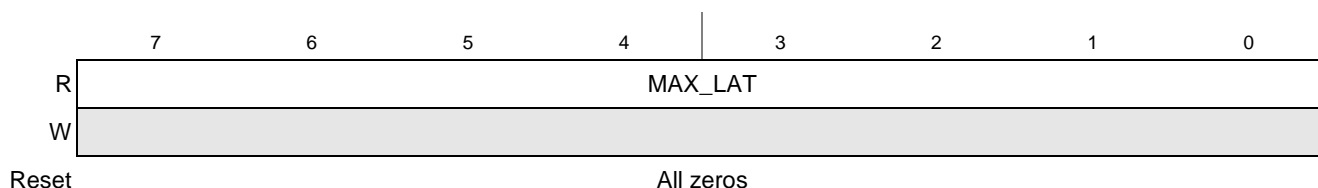


Figure 13-41. Maximum Latency Configuration Register

13.3.3.24 PCI Function Configuration Register

Figure 13-42 shows the PCI function configuration register fields.

Offset 0x44

Access: Read/Write

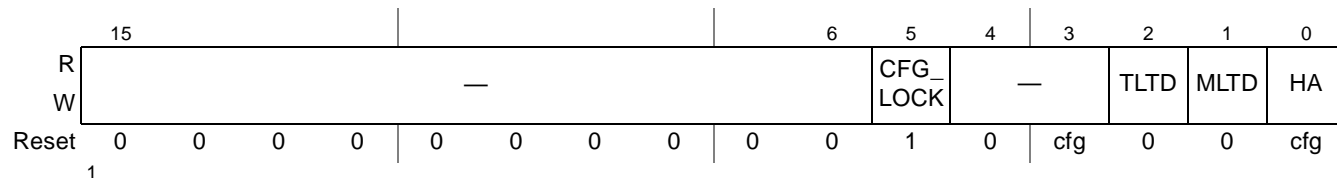


Figure 13-42. PCI Function Configuration Register

Table 13-40 shows the bit settings of the PCI function configuration register.

Table 13-40. PCI Function Configuration Register Field Descriptions

Bits	Name	Description
15–6	—	Reserved
5	CFG_LOCK	Configuration lock. Controls access to the PCI configuration space from the PCI port. In host mode the PCI configuration space is always inaccessible, so this bit is not used. Normally, this bit will be cleared in agent mode once the configuration of the PCI controller is complete to allow an external host to access the PCI configuration space. 0 Access to the configuration spaces is permitted. 1 Any inbound PCI access to the PCI configuration space is retried. See Section 4.3.1.1, “Reset Configuration Word Source,” for more information on reset configuration.
3–4	—	Reserved

Table 13-40. PCI Function Configuration Register Field Descriptions (continued)

Bits	Name	Description
2	TLTD	Target latency timeout disable. Determines whether the PCI controller, 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.
1	MLTD	Master latency timer disable. Determines whether the PCI controller, while acting as a PCI master, terminates a transaction upon the expiration of the master latency timer. 0 Master latency timer enabled. 1 Master latency timer disabled.
0	HA	Host/Agent. Indicates whether the PCI controller is in host mode or agent mode. It provides the value of the <code>PCI_HOST</code> —PCI host configuration bit is sampled at the end of the reset sequence. 0 Host mode 1 Agent mode

13.3.3.25 PCI Arbiter Control Register (PCIACR)

Figure 13-43 shows the PCI arbiter control register (PCIACR) fields.

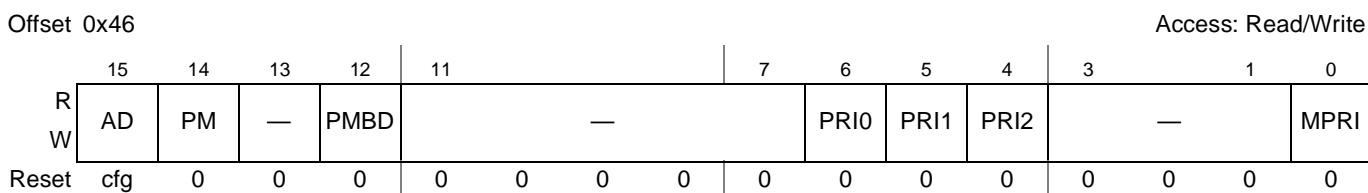


Figure 13-43. PCI Arbiter Control Register (PCIACR)

Table 13-41 shows the bit settings of the PCIACR.

Table 13-41. PCI Arbiter Control Register (PCIACR) Field Descriptions

Bits	Name	Description
15	AD	Arbiter disable. Indicates whether the PCI controller functions as the arbiter for the PCI bus. It provides the value of the PCI arbiter enable configuration bit as sampled at the end of the reset sequence. See Chapter 4, “Reset, Clocking, and Initialization,” for more information on reset configuration. 0 Arbiter enabled 1 Arbiter disabled
14	PM	Parking mode. Controls which device receives a bus grant when there are no outstanding bus requests and the bus is idle. 0 The bus is parked with the last device to use the bus. 1 The bus is parked with the PCI controller.
13	—	Reserved
12	PBMD	PCI broken master disable. Determines whether the PCI controller ignores the bus requests of an initiator that requests the bus for an excessive period without using it. 0 An initiator that requests the bus and receives the grant must begin using the bus within 16 PCI clock periods after the bus becomes idle or its request is subsequently ignored. 1 No requests are ignored.
11–7	—	Reserved

Table 13-41. PCI Arbiter Control Register (PCIACR) Field Descriptions (continued)

Bits	Name	Description
6–4	PRIn	Priority level for master <i>n</i> . When the PCI controller functions as the arbiter for the PCI bus, each PRIn bit determines the arbitration priority level for the PCI master connected to the REQn/GNTn pair. 0 Low priority 1 High priority
3–1	—	Reserved
0	MPRI	My priority. When the PCI controller functions as the arbiter for the PCI bus, this bit determines the arbitration priority level for the PCI controller when it acts as a PCI master. 0 Low priority 1 High priority

13.3.3.26 Hot Swap Register Block

Figure 13-44 shows the hot swap register block fields.

Offset 0x48

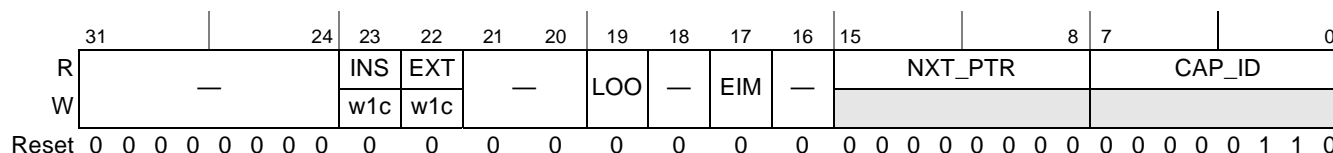


Figure 13-44. Hot Swap Register Block

Table 13-42 shows the bit settings of the Hot Swap register block.

Table 13-42. Hot Swap Register Block Field Descriptions

Bits	Name	Description
31–24	—	Reserved
23	INS	Insertion status. Indicates that a card has been inserted. Write 1 to clear this bit.
22	EXT	Extraction status. Indicates that a card has been extracted. Write 1 to clear this bit.
21–20	—	Reserved
19	LOO	LED On/Off. Controls the LED when the hardware is in state H2 0 LED off 1 LED on
18	—	Reserved
17	EIM	ENUM mask. This bit masks the CPCI_HS_ENUM input. 0 Enabled 1 Masked
16	—	Reserved
15–8	NXT_PTR	Next pointer—hardwired to 0x00 to indicate that this is the last item in the capabilities list.
7–0	CAP_ID	Capability ID for hot swap (hardwired to 0x06)

13.4 Functional Description

The following sections discuss the operation of the PCI controller.

13.4.1 PCI Bus Arbitration

The PCI bus arbitration approach is access-based. Bus masters must arbitrate for each access performed on the bus. PCI uses a central arbitration scheme where each master has its own unique request (\overline{REQn}) output and grant (\overline{GNTn}) input signal. A simple request-grant handshake is used to gain access to the bus. Arbitration for the bus occurs during the previous access so that no PCI bus cycles are consumed waiting for arbitration (except when the bus is idle).

The PCI internal arbiter supports three external masters (besides the PCI controller itself) by using the \overline{REQ} signals and generating the \overline{GNT} signals.

During reset, the PCI controller samples the reset configuration bit (and programs the PCI_ARB_DIS bit accordingly) to determine if the arbiter is enabled or disabled. See [Chapter 4, “Reset, Clocking, and Initialization,”](#) for more information. The arbiter can also be enabled or disabled by directly programming the PCI_ARB_DIS bit in the arbiter configuration register (see [Section 13.3.3.25, “PCI Arbiter Control Register \(PCIACR\),”](#) for more information). However, it is recommended to use the reset configuration bit to set the arbiter state because the arbiter state controls the direction of $\overline{REQ0}$ and $\overline{GNT0}$.

If the arbiter is disabled, the PCI controller uses $\overline{REQ0}$ to issue requests to an external arbiter, and uses $\overline{GNT0}$ to receive grants from the external arbiter.

13.4.1.1 Bus Parking

When no devices are requesting the bus, the bus is granted, or parked, for a specified device to prevent the AD, PCI_C/ \overline{BE} and PCI_PAR signals from floating. The PCI controller can be configured to either park on itself or park on the last master to use the bus (see [Section 13.3.3.25, “PCI Arbiter Control Register \(PCIACR\),”](#) for more information).

13.4.1.2 Arbitration Algorithm

The round-robin arbitration algorithm has two priority levels. Each of the external PCI bus masters, plus the PCI controller, are assigned either a high or a low priority level, as programmed in the arbiter configuration register (see [Section 13.3.3.25, “PCI Arbiter Control Register \(PCIACR\).”](#)) Within each priority group (high or low), the bus grant is given to the next requesting device in numerical order, with the PCI controller itself positioned before device 0. \overline{GNTn} is asserted for device n as soon as the previously granted device begins a transaction. Conceptually, the lowest priority device at any given time is the current bus master and the highest priority device is the next one to follow the current master. This is considered to be a fair algorithm because a given device cannot prevent other devices from having access to the bus—a given device automatically becomes the lowest priority device as soon as it begins to use the bus. If a master is not requesting the bus, the transaction slot is given to the next requesting device within the priority group.

The grant given to one device may be taken away and whenever a higher priority device asserts its request. If the bus is idle when a new device is to receive a grant, no device receives a grant for one clock; in the

next clock, the new winner of the arbitration receives a grant. This operation allows for a turnaround clock when a device is using address stepping or when the bus is parked.

The low priority group collectively receives one bus transaction request slot in the high priority group. Therefore, if there are N high-priority devices, each high-priority device is guaranteed to get at least one of (N+1) bus transactions, and the M low priority devices are guaranteed to each get at least one of (N+1) x M bus transactions, with one of the low-priority devices receiving the grant in one of (N+1) bus transactions. If all devices are programmed to the same priority level or if there is only one device at the low priority, the algorithm provides each device an equal number of bus grants in a round-robin sequence.

An arbitration example with three masters in the high priority group and two in the low priority group is shown in Figure 13-45. Noting that one position in the high priority group is actually a place-holder for the low priority group, it can be seen that each high priority initiator is guaranteed at least 1 out of 3 transaction slots, and each low priority initiator is guaranteed at least 1 out of 6 slots. Assuming all devices are requesting the bus, the grant sequence (with device 1 being the current master) is as follows: 0, 2, the PCI controller, 0, 2, 1, 0, 2, the PCI controller, and so on. If, for example, device 2 is not requesting the bus, the grant sequence becomes 0, the PCI controller, 0, 1, 0, the PCI controller, and so on. If device 2 now requests the bus at a point in the sequence when device 0 is conducting a transaction and the PCI controller is the next grant, then the PCI controller's grant is removed, and the higher-priority device 2 is awarded the next grant.

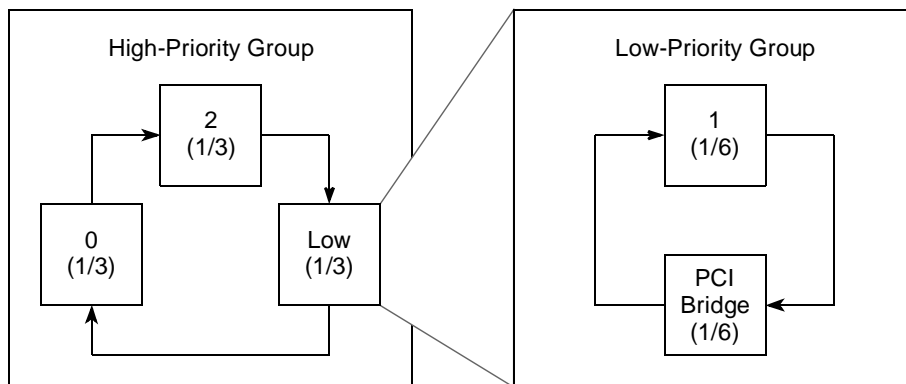


Figure 13-45. PCI Arbitration Example

13.4.1.3 Broken Master Lock-Out

The broken master feature allows the arbiter to lock out any masters that are broken or ill-behaved. This feature is controlled by programming the PCI arbiter control register. When the broken master feature is enabled, a granted device that does not assert $\overline{\text{PCI_FRAME}}$ within 16 PCI clock cycles after the bus is idle, has its grant removed and subsequent requests are ignored until its $\overline{\text{REQ}}$ is negated for at least one clock cycle. This prevents ill-behaved masters from monopolizing the bus. When the broken master feature is disabled, a device that requests the bus and receives a grant never loses its grant until and unless it begins a transaction or negates its $\overline{\text{REQ}}$ signal. Note that disabling the broken master feature is not recommended.

13.4.1.4 Master Latency Timer

The PCI controller implements the master latency timer register (see [Section 13.3.3.10, “Latency Timer Configuration Register”](#)) to prevent itself from monopolizing the bus. When the master latency timer expires, the PCI controller checks the state of its $\overline{\text{PCI_GNT}}$ signals. If the $\overline{\text{PCI_GNT}}$ signal is not asserted, the PCI controller completes one more data phase and relinquishes the bus. The master latency timer can be disabled if needed (see [Section 13.3.3.24, “PCI Function Configuration Register,”](#) for more information).

13.4.2 Bus Commands

PCI bus commands indicate the type of transaction occurring on the bus. These commands are encoded on $\text{PCI_C}/\overline{\text{BE}}[3:0]$ during the address phase of the transaction. PCI bus commands are described in [Table 13-43](#).

Table 13-43. PCI Command Definitions

PCI_C/ BE[3:0]	Command Type	Supported as:		Definition
		Initiator	Target	
0b0000	Interrupt acknowledge	Yes	No	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 phase.
0b0001	Special cycle	Yes	No	Provides a simple message broadcast mechanism. See Section 13.4.4.6, “Special Cycle Command,” for more information.
0b0010	I/O read	Yes	No	Accesses agents mapped in I/O address space.
0b0011	I/O write	Yes	No	Accesses agents mapped in I/O address space.
0b010x	—	—	—	Reserved. No response occurs.
0b0110	Memory read	Yes	Yes	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.
0b0111	Memory write	Yes	Yes	Accesses agents mapped in memory address space. Note that for inbound writes less than 4-bytes, the PCI controller splits the transaction into single byte writes to the target. Thus, the PCI interface cannot be used to perform single beat writes to 16-bit devices on the local peripheral interfaces.
0b100x	—	—	—	Reserved. No response occurs.
0b1010	Configuration read	Yes	Yes	Accesses the configuration space of each agent. An agent is selected when its IDSEL signal is asserted. See Section 13.4.4.4, “Host Mode Configuration Access,” for more information on configuration accesses. As a target, a configuration read is only accepted if the PCI controller is configured to be in agent mode.
0b1011	Configuration write	Yes	Yes	Accesses the configuration space of each agent. An agent is selected when its IDSEL signal is asserted. See Section 13.4.4.4, “Host Mode Configuration Access,” for more information. As a target, a configuration write is only accepted if the PCI controller is configured to be in agent mode.

Table 13-43. PCI Command Definitions (continued)

PCI_C/ BE[3:0]	Command Type	Supported as:		Definition
		Initiator	Target	
0b1100	Memory read multiple	Yes	Yes	Causes a prefetch of the next cache line.
0b1101	Dual address cycle	No	Yes	Transfers an 8-byte address to devices.
0b1110	Memory read line	Yes	Yes	Indicates that the initiator intends to transfer an entire cache line of data.
0b1111	Memory write and invalidate	No	Yes	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.

13.4.3 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.

13.4.3.1 Basic Transfer Control

PCI data transfers are controlled by the following 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 negated. 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 PCI controller, 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 PCI controller, 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 PCI controller (as a master) intends to complete only one more data transfer, $\overline{\text{PCI_FRAME}}$ is negated 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.

13.4.3.2 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 supports the 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. Once a target has claimed an I/O access, it first determines if it can complete the entire access as indicated by the byte enable signals. If all the selected bytes are not in the address range, the entire access should not be completed; that is, the target should not transfer any data and should terminate the transaction with a target-abort operation. See [Section 13.4.3.6, “Bus Transactions,”](#) for more information.

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. The PCI controller determines a configuration access is for a device on the PCI bus by decoding a configuration command. When in agent mode, the PCI controller responds to host-generated PCI configuration cycles when its IDSEL is asserted during a configuration cycle.

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 PCI controller 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 PCI controller 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.

13.4.3.3 Device Selection

As a target, the PCI controller drives $\overline{\text{PCI_DEVSEL}}$ one clock following the address phase as indicated in the configuration space status register; see [Section 13.3.3.4, “PCI Status Configuration Register,”](#) for more information. The PCI controller 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 PCI controller enables its $\overline{\text{PCI_TRDY}}$, $\overline{\text{PCI_STOP}}$, or data (for a read). The $\overline{\text{PCI_DEVSEL}}$ signal is not negated until $\overline{\text{PCI_FRAME}}$ is negated, 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 13.4.3.8, “Transaction Termination,”](#) for more information.

As an initiator, if the PCI controller does not see the assertion of $\overline{\text{PCI_DEVSEL}}$ within 4 clocks of $\overline{\text{PCI_FRAME}}$, it terminates the transaction with a master-abort as described in [Section 13.4.3.8, “Transaction Termination,”](#) for more information.

13.4.3.4 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 PCI controller, 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 PCI controller expects the data not to be changed, and on a write transaction, the data is not stored.

13.4.3.5 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. $\overline{\text{PCI_FRAME}}$, $\text{PCI_C}/\overline{\text{BE}}[3:0]$, and $\text{AD}[31:0]$ 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 negated).

Byte lanes not involved in the current data transfer are driven to a stable condition even though the data is not valid.

13.4.3.6 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 ‘negated’ refer to the globally visible state of the signal on the clock edge, and not to signal transitions.
- The symbol  represents a turnaround-cycle.

13.4.3.7 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 $\text{AD}[31:0]$ signals contain a byte address and the $\text{PCI_C}/\overline{\text{BE}}[3:0]$ 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 $\text{PCI_C}/\overline{\text{BE}}[3:0]$ signals indicate a read command. [Figure 13-46](#) shows an example of a single beat read transaction.

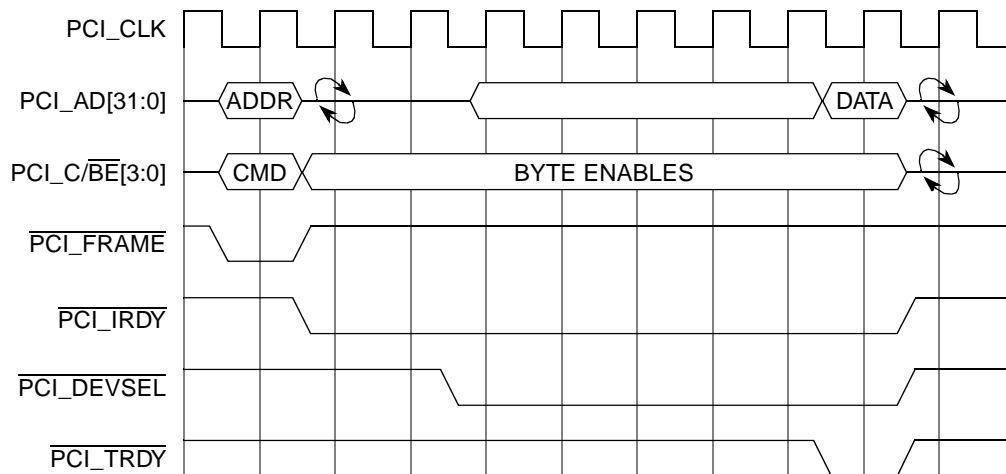
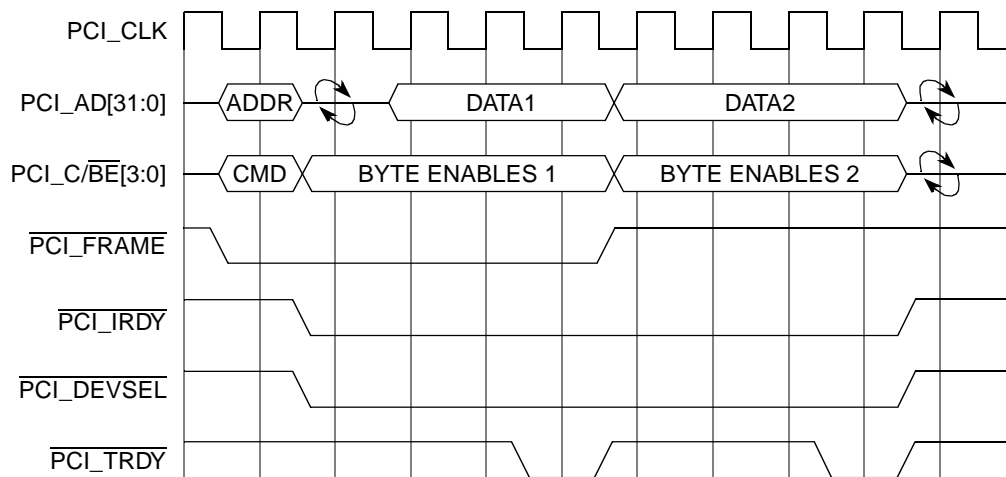

Figure 13-46. Single Beat Read Example

Figure 13-47 shows an example of a burst read transaction.


Figure 13-47. Burst Read Example

During the turnaround-cycle following the address phase, the $\overline{\text{PCI_C/BE}}[3:0]$ signals indicate which byte lanes are involved in the data phase. The turnaround-cycle must be enforced by the target with the $\overline{\text{PCI_TRDY}}$ signal if using fast $\overline{\text{PCI_DEVSEL}}$ 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 $\overline{\text{PCI_DEVSEL}}$ is asserted except during the turnaround cycle.

The data phase completes when data is transferred, which occurs when both $\overline{\text{PCI_IRDY}}$ and $\overline{\text{PCI_TRDY}}$ are asserted on the same clock edge. When either is negated, a wait cycle is inserted and no data is transferred. To indicate the last data phase $\overline{\text{PCI_IRDY}}$ must be asserted when $\overline{\text{PCI_FRAME}}$ is negated.

A write transaction starts when $\overline{\text{PCI_FRAME}}$ is asserted for the first time and the $\text{PCI_C}/\overline{\text{BE}}[3:0]$ signals indicate a write command. Figure 13-48 shows an example of a single-beat write transaction.

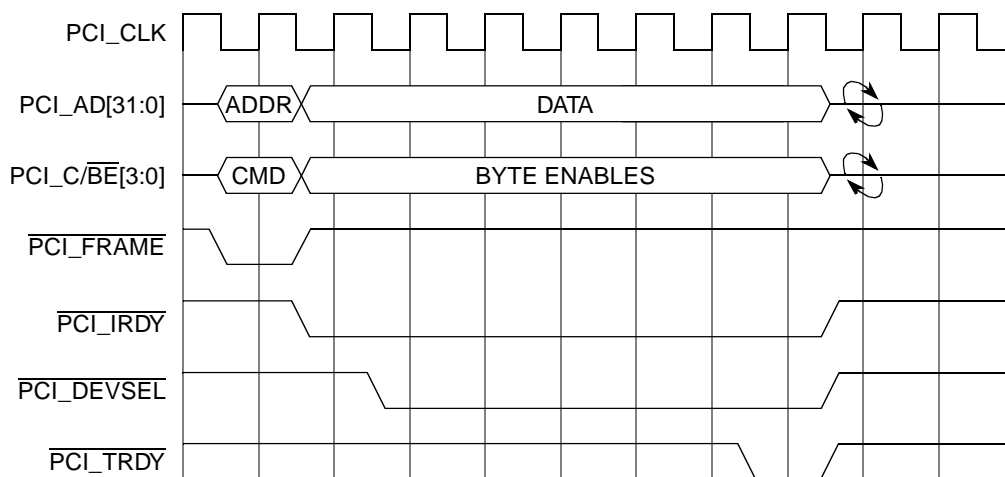


Figure 13-48. Single Beat Write Example

Figure 13-49 shows an example of a burst write transaction.

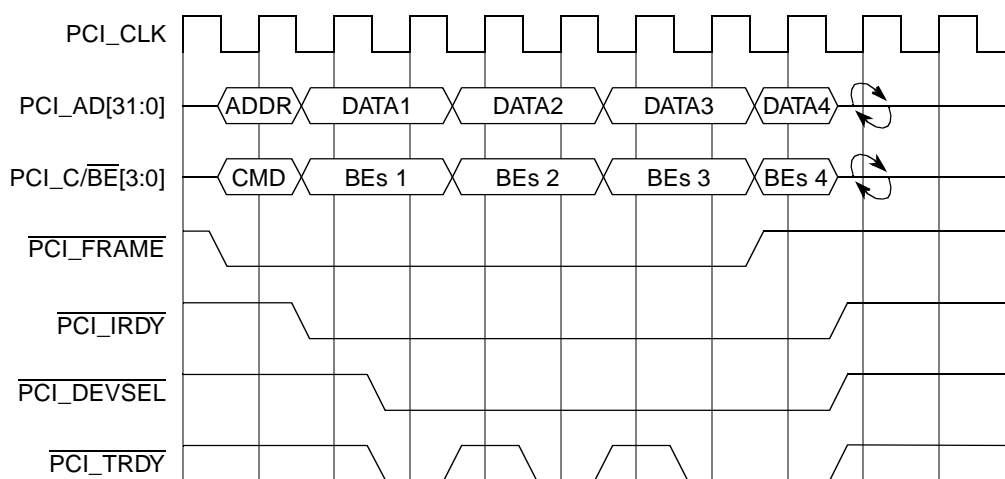


Figure 13-49. Burst Write Example

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.

13.4.3.8 Transaction Termination

The termination of a PCI transaction is orderly and systematic, regardless of the cause of the termination. All transactions end when $\overline{\text{PCI_FRAME}}$ and $\overline{\text{PCI_IRDY}}$ are both negated, indicating the idle cycle.

The PCI controller as an initiator terminates a transaction when $\overline{\text{PCI_FRAME}}$ is negated and $\overline{\text{PCI_IRDY}}$ is asserted. This indicates that the final data phase is in progress. The final data transfer occurs when both $\overline{\text{PCI_TRDY}}$ and $\overline{\text{PCI_IRDY}}$ are asserted. A master-abort is an abnormal case of a master initiated

termination. If the PCI controller detects that $\overline{\text{PCI_DEVSEL}}$ has remained negated for more than four clocks after the assertion of $\overline{\text{PCI_FRAME}}$, it negates $\overline{\text{PCI_FRAME}}$ and then, on the next clock, negates $\overline{\text{PCI_IRDY}}$. On aborted reads, the PCI controller returns 0xFFFF_FFFF. The data is lost on aborted writes.

When the PCI controller as a target needs to suspend a transaction, it asserts $\overline{\text{PCI_STOP}}$. Once asserted, $\overline{\text{PCI_STOP}}$ remains asserted until $\overline{\text{PCI_FRAME}}$ is negated. Depending on the circumstances, data may or may not be transferred during the request for termination. If $\overline{\text{PCI_TRDY}}$ and $\overline{\text{PCI_IRDY}}$ are asserted during the assertion of $\overline{\text{PCI_STOP}}$, data is transferred. This type of target-initiated termination is called a disconnect B, shown in Figure 13-50. If $\overline{\text{PCI_TRDY}}$ is asserted when $\overline{\text{PCI_STOP}}$ is asserted but $\overline{\text{PCI_IRDY}}$ is not, $\overline{\text{PCI_TRDY}}$ must remain asserted until $\overline{\text{PCI_IRDY}}$ is asserted and the data is transferred. This is called a disconnect A target-initiated termination, also shown in Figure 13-50. However, if $\overline{\text{PCI_TRDY}}$ is negated when $\overline{\text{PCI_STOP}}$ 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 13-48).

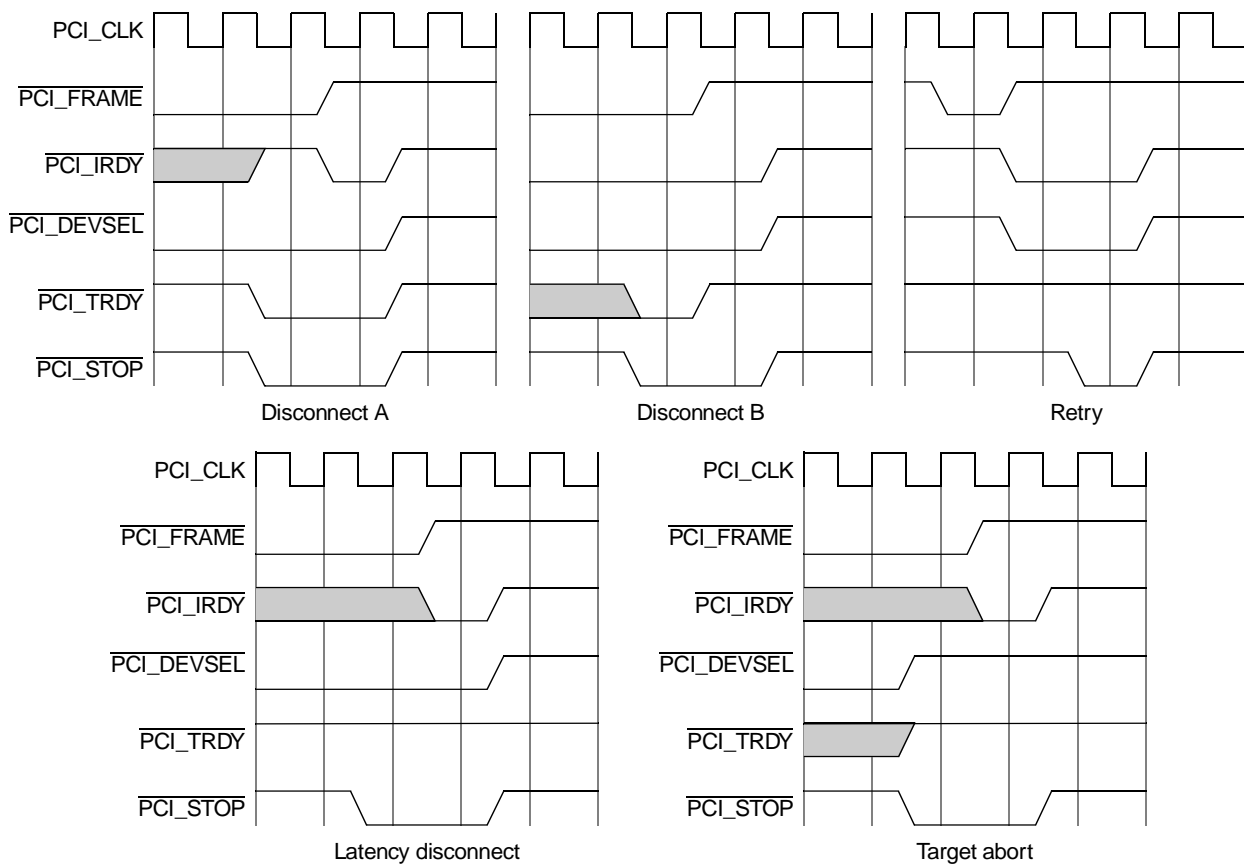


Figure 13-50. Target-Initiated Terminations

Note that when an initiator is terminated by $\overline{\text{PCI_STOP}}$, it must negate its $\overline{\text{REQn}}$ 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 $\overline{\text{REQn}}$ immediately following the two clocks or potential starvation may occur. If the initiator does not intend to complete the transaction, it can assert $\overline{\text{REQn}}$ whenever it needs to use the PCI bus again.

The PCI controller terminates a transaction in the following cases:

- Eight PCI clock cycles have elapsed between data phases. This is a ‘latency disconnect’ (see [Figure 13-48](#)).
- 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 4-Kbyte 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 PCI controller can be optionally disabled. See [Section 13.3.3.24, “PCI Function Configuration Register,”](#) for more information.

When the PCI controller is in host mode it does not respond to any PCI configuration transactions. When the PCI controller is in agent mode and the CFG_LOCK lock bit is set (see [Section 13.3.3.24, “PCI Function Configuration Register”](#)) the PCI controller retries 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 13-48](#).

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 negated. 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.

The PCI controller terminates a transaction with target-abort in the case in which it is the intended target of a read transaction from system memory and the data from memory is corrupt. If the PCI controller 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 PCI controller does not target-abort in this case.

If the PCI controller is mastering a transaction that terminates with a target-abort, undefined data is returned on a read and write data is lost. An example of a target-abort is shown in [Figure 13-48](#).

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.

13.4.4 Other Bus Operations

The following sections provide information on additional PCI bus operations.

13.4.4.1 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 PCI controller as a target supports both types of fast back-to-back transactions but does not support them as an initiator. The PCI controller 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 PCI controller detects a fast-back-to-back operation and did not drive `PCI_DEVSEL` in the previous cycle, it delays the assertion of `PCI_DEVSEL` and `PCI_TRDY` for one cycle to allow the other target to get off the bus.

13.4.4.2 Dual Address Cycles

The PCI controller 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 PCI controller supports single-beat and burst DAC transactions.

13.4.4.3 Data Streaming

The PCI controller provides data streaming for PCI transactions to and from prefetchable memory. In other words, when the PCI controller 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 PCI controller disconnects after the first data phase so streaming cannot 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 [Section 13.3.2.14, “PCI Inbound Window Attribute Registers \(PIWARn\),”](#) for more information) 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 PCI controller reads one cache line from memory. If the transaction crosses a cache line boundary, the PCI controller starts the read of a new cache line. For a memory read multiple command, the PCI controller reads two cache lines from memory. When the PCI transaction finishes the read for the first cache line, the PCI controller performs a speculative read of a third cache line. The PCI controller 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 PCI controller runs out of buffer space on writes, or the PCI controller cannot supply consecutive data beats for reads within eight PCI bus clocks of each other. A disconnect also occurs if the transaction crosses a 4-Kbyte page boundary.

13.4.4.4 Host Mode Configuration Access

The PCI controller provides two types of configuration accesses to support hierarchical bridges. To access configuration space, a value is written to the CONFIG_ADDR register specifying which PCI bus, which device, and which configuration register to be accessed.

When the PCI controller sees an access that falls inside the 4 bytes beginning at the CONFIG_DATA address, it checks the enable bit, the device number and the bus number in the CONFIG_ADDR register. If the enable bit is set and the device number is not equal to all ones, a configuration cycle translation is performed. When the device number field is equal to all ones, it has a special meaning (see [Section 13.4.4.6, “Special Cycle Command,”](#) for more information).

There are two types of translations supported:

- Type 0 translations—For when the device is on the PCI bus connected to the PCI controller.
- Type 1 translations—For when the device is on another bus somewhere behind the PCI controller.

For type 0 translations, the PCI controller decodes the device number field to assert the appropriate IDSEL line and perform a configuration cycle on the PCI bus with AD[1:0] as 0b00. All 21 IDSEL bits are decoded, starting with bit AD11. That is, if the device number field contains 0b01011, AD11 on the PCI bus is set. The IDSEL lines are bit-wise associated with increasing values for the device number such that AD12 corresponds to 0b01100, and so on up to bit 30 as shown in [Table 13-41](#). AD31 is selected with 0b01010. A device number of 0b11111 indicates a special cycle. Device number 0b00000 is used for configuring the PCI controller itself. Bits 10 through 8 are copied to the PCI bus as an encoded value for components which contain multiple functions. Bits 7 through 2 are also copied onto the PCI bus. The PCI controller implements address stepping on configuration cycles so that the target’s PCI_IDSEL, which is connected directly to one of the AD lines, reaches a stable value. This means that a valid address and command are driven on the AD and PCI_C/ $\overline{\text{BE}}$ lines one cycle before the assertion of $\overline{\text{PCI_FRAME}}$.

For type 1 translations, the PCI controller copies the contents of the CONFIG_ADDR register directly onto the PCI address/data lines during the address phase of a configuration cycle, with the exception that AD[1-0] contains 0b01 (not 0b00 as in Type 0 translations).

When the PCI controller is configured as a host device, a local master sometimes needs to perform configuration reads from unpopulated PCI slots (as part of the system configuration). To avoid getting a machine check interrupt, the following steps should be taken:

1. Mask the NORSP bit in the error mask register. See [Section 13.3.2.9, “PCI Error Control Register \(PCI_ECR\).”](#)
2. Perform the PCI configuration reads.
3. Clear the NORSP bit in the error status register.
4. Unmask (write 1) the NORSP bit in the error mask register. See [Section 13.3.2.3, “PCI Error Enable Register \(PCI_EER\).”](#)

13.4.4.5 Agent Mode Configuration Access

When the PCI controller is configured as an agent device, it responds to remote host generated PCI configuration accesses to the PCI interface. This is indicated by decoding the configuration command

along with the PCI controller's IDSEL being asserted. A remote host can access the 256-byte PCI configuration area and the memory-mapped configuration registers within the PCI controller.

13.4.4.6 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 negated. Special cycles terminate with a master-abort. (In the special cycle case, the received-master-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 PCI_CONFIG_ADDRESS register is written with a value so that the bus number matches the bridge bus, the device number is all ones, the function number is all ones, and the register number is zero. The next time the PCI_CONFIG_DATA register is accessed, the PCI controller executes either a special cycle or an interrupt acknowledge command. When the PCI_CONFIG_DATA register is written, the PCI controller generates a special cycle encoding on the command/byte enable lines during the address phase and drives the data from the PCI_CONFIG_DATA register onto the address/data lines during the first data phase.

If the bus number field of the PCI_CONFIG_ADDRESS does not match one of the PCI controller bus numbers, the PCI controller passes the write to PCI_CONFIG_DATA through to the PCI bus as a type 1 configuration cycle as it does any other time the bus number field does not match.

Table 13-44. Special Cycle Commands

Address (AD[15-0])	Message Type	Description
0x0000	SHUTDOWN (SLEEP)	Indicates the processor is entering its most power saving mode
0x0001	HALT (DOZE)	Indicates the processor is entering a power save mode where address decoding is still available
0x0002–0xFFFF	—	Reserved for future commands

13.4.4.7 Interrupt Acknowledge

When the PCI_CONFIG_ADDRESS register is 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 PCI_CONFIG_DATA register is accessed the PCI controller does either a special cycle command or an interrupt acknowledge command. When the PCI_CONFIG_DATA register is read, the PCI controller

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.

13.4.5 Error Functions

This section describes PCI bus errors.

13.4.5.1 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 PCI controller 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 PCI controller. When an address or data parity error is detected, the detected-parity-error bit in the configuration space status register is set (see [Section 13.3.3.4, “PCI Status Configuration Register.”](#))

13.4.5.2 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 [Section 13.3.3.3, “PCI Command Configuration Register,”](#) for more information). If the parity-error-response bit is cleared, the PCI controller completes all transactions regardless of parity errors (address or data). If the bit is set, the PCI controller 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. When acting as an initiator during a read transaction or as a target involved in a write to system memory the PCI controller asserts $\overline{\text{PCI_PERR}}$.

Figure 13-51 shows the possible assertion points for $\overline{\text{PCI_PERR}}$ if the PCI controller detects a data parity error.

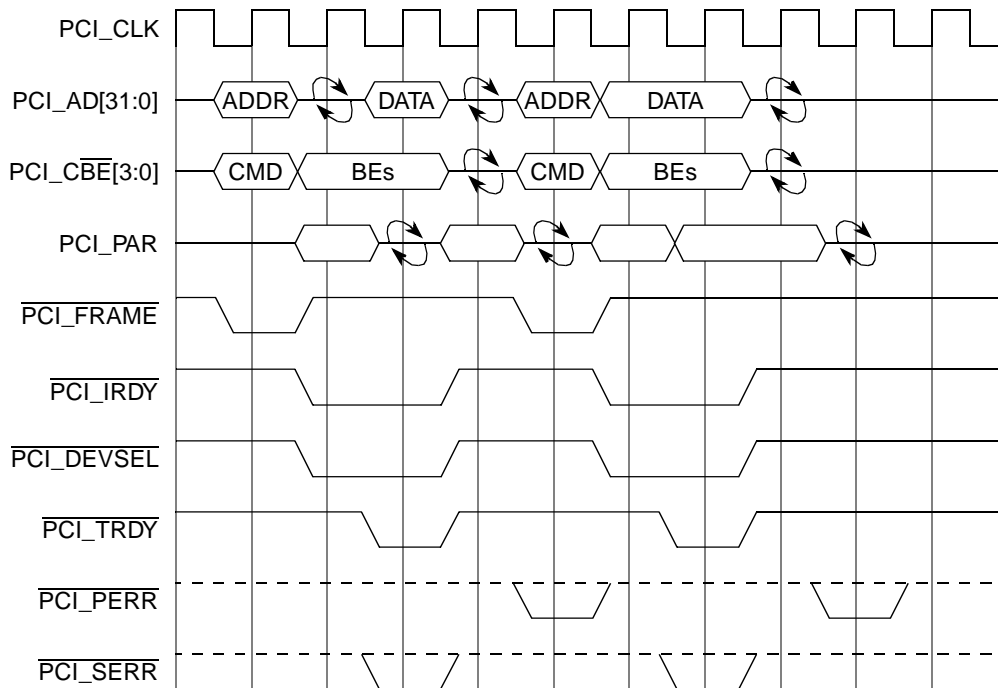


Figure 13-51. PCI Parity Operation

As an initiator, the PCI controller 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 PCI controller aborts the transaction internally. As a target, the PCI controller 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 PCI controller asserts $\overline{\text{PCI_SERR}}$, 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 $\overline{\text{PCI_SERR}}$ is conditioned on the parity-error-response bit being enabled in the command register. $\overline{\text{PCI_SERR}}$ is asserted when the PCI controller detects an address parity error while acting as a target. The system error is passed to the PCI controller's interrupt processing logic to assert $\overline{\text{MCP}}$. Figure 13-51 shows where the PCI controller could detect an address parity error and assert $\overline{\text{PCI_SERR}}$ or where the PCI controller, acting as an initiator, checks for the assertion of $\overline{\text{PCI_SERR}}$ signaled by the target detecting an address parity error.

As a target that asserts $\overline{\text{PCI_SERR}}$ on an address parity, the PCI controller completes the transaction on the PCI bus, aborting internally if the transaction is a write to system memory. If $\overline{\text{PCI_PERR}}$ is asserted during a PCI controller write to PCI, the PCI controller attempts to continue the transfer, allowing the target to abort/disconnect if desired. If the PCI controller detects a parity error on a read from PCI, the PCI controller 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; \overline{MCP} is also asserted to the core as an option.

13.4.6 PCI Inbound Address Translation

For inbound transactions (transactions generated by an external master on the PCI bus where the PCI controller responds as a slave device), the PCI controller only responds to PCI addresses within the windows mapped by the PCI inbound base address registers (PIBARs). 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 master to access local memory. Each PIBAR register is associated with a PITAR and PIWAR which are located in the PCI controller's PCI CSR space. Figure 13-52 shows an example translation window for inbound memory accesses.

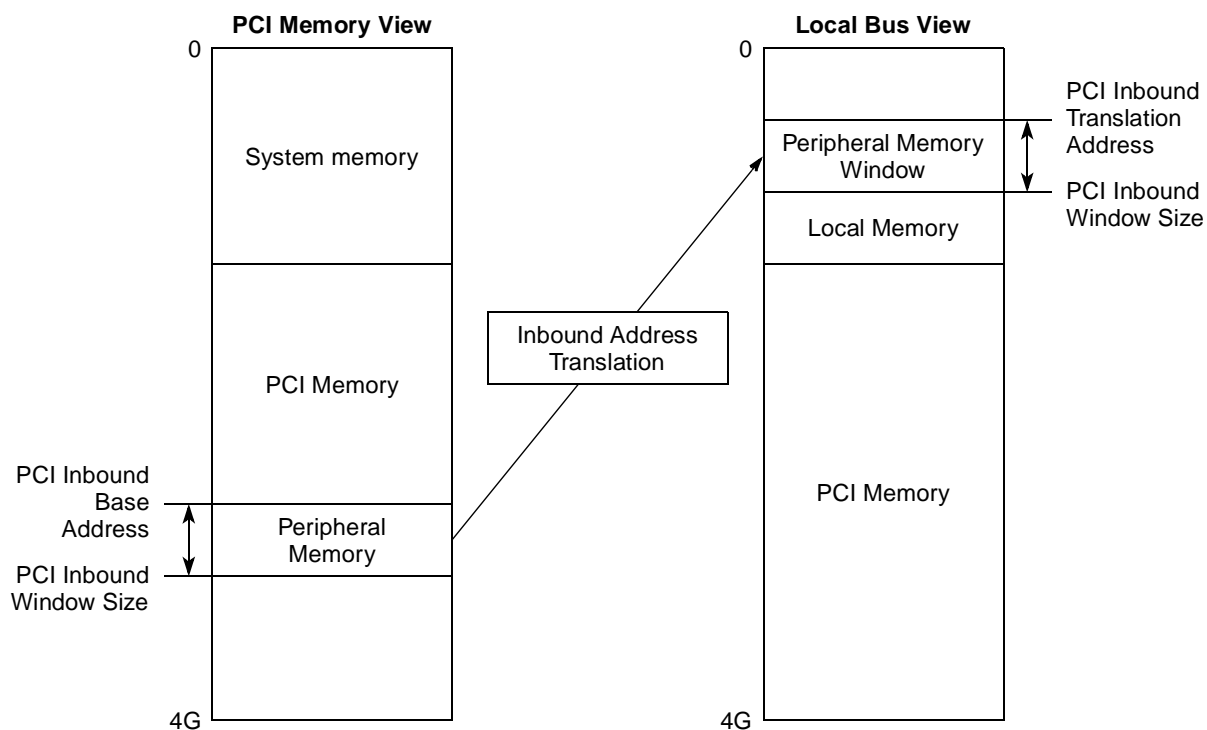


Figure 13-52. Inbound PCI Memory Address Translation

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 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.

The translation windows are disabled after reset, that is, after reset, the PCI controller does not acknowledge externally mastered transactions on the PCI bus by asserting $\overline{PCI_DEVSEL}$ until the inbound translation windows are enabled.

13.4.7 CompactPCI Hot Swap Specification Support

CompactPCI is an open specification supported by the PCI Industrial Computer Manufacturers Group (PICMG) and is intended for embedded applications using PCI. CompactPCI Hot Swap is an extension of the CompactPCI specification and allows the insertion and extraction (or “hot swapping”) of boards without adversely affecting system operation. The hot swap specification defines the following levels of support:

- Hot swap capable
- Hot swap friendly
- Hot swap ready

The PCI controller is hot swap friendly, meaning that it supports the hardware and software connection processes as defined in the hot swap specification. This level of support allows the board and system designers to build full Hot Swap and high availability systems based on the PCI controller as a PCI target device. For details on the hot swap process, refer to the *Hot Swap Specification PICMG 2.1*, R1.0, August 3, 1998.

13.4.8 Byte Ordering

Whenever data must cross a bridge between two busses, the byte ordering of data on the source and destination buses must be considered. The internal platform bus of this device is inherently big endian and the PCI bus interface is inherently little endian.

There are two methods to handle ordering of data as it crosses a bridge—address invariance and data invariance. Address invariance preserves the addressing of bytes within a scalar data element, but not the relative significance of the bytes within that scalar. Conversely, data invariance preserves the relative significance of bytes within a scalar, but not the addressing of the individual bytes that make up a scalar.

This device uses address invariance as its byte ordering policy

As stated above, address invariance preserves the byte address of each byte on an I/O interface as it is placed in memory or moved into a register. This policy can have the effect of reversing the significance order of bytes (most significant to least significant and vice versa), but it has the benefit of preserving the format of general data structures. Provided that software is aware of the endianness and format of the data structure, it can correctly interpret the data on either side of the bridge.

Figure 13-53 shows the transfer of a 4-byte scalar, 0x4142_4344, from a big endian source across an address invariant bridge to a little endian destination.

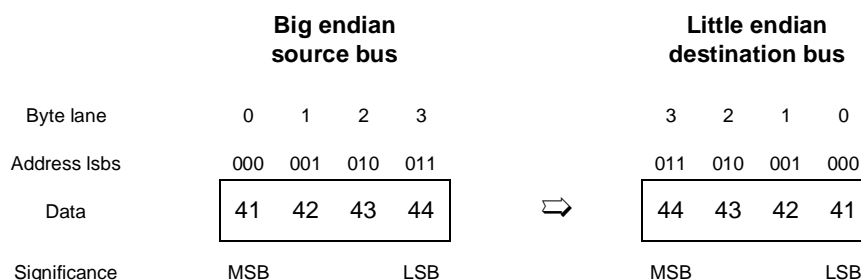


Figure 13-53. Address Invariant Byte Ordering—4 bytes Outbound

Note that although the significance of the bytes within the scalar have changed, the address of the individual bytes that make up the scalar have not changed. As long as software is aware that the source of the data used a big endian format, the data can be interpreted correctly.

Figure 13-55 shows data flowing the other way, from a little endian source to a big endian destination.

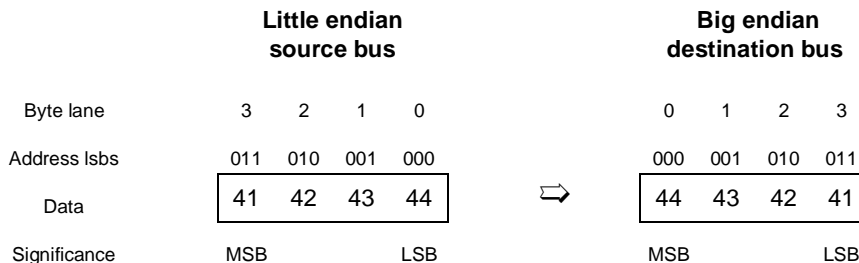


Figure 13-54. Address Invariant Byte Ordering—4 bytes Inbound

Figure 13-55 shows an outbound transfer of an 8-byte scalar, 0x5455_1617_CDCE_2728, using address invariance.

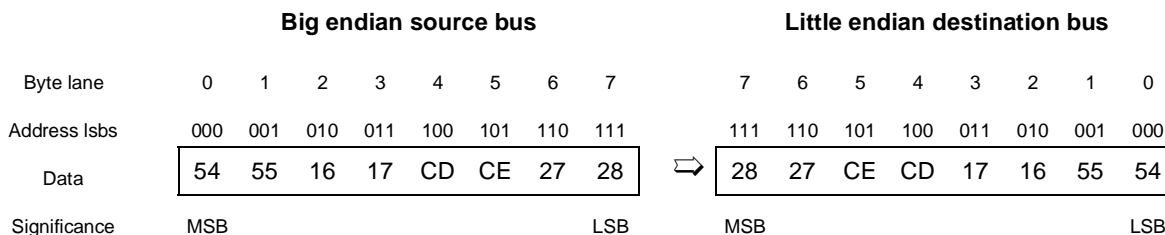


Figure 13-55. Address Invariant Byte Ordering—8 bytes Outbound

Figure 13-56 shows an inbound transfer of a 2-byte scalar, 0x5837, using address invariance.

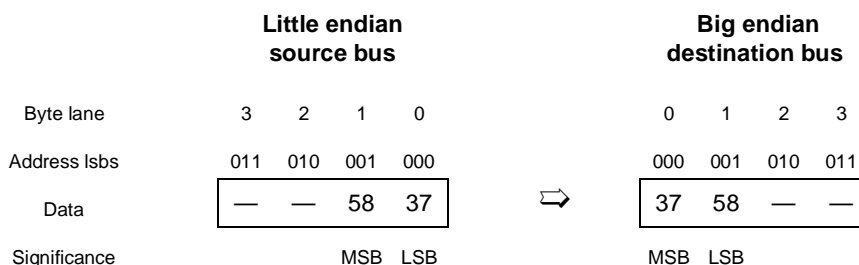


Figure 13-56. Address Invariant Byte Ordering—2 bytes Inbound

Note that in all of these examples, the original addresses of the individual bytes within the scalars (as created by the source) have been preserved.

13.4.8.1 Byte Order for Configuration Transactions

All internal memory-mapped registers in the CCSR space use big endian byte ordering. However, the PCI specification defines PCI configuration registers as little endian. All accesses to the PCI configuration port, CFG_DATA, including the those targeting the internal PCI configuration registers, use the address invariance policy as shown in Figure 13-57. Therefore, software must access CFG_DATA with

little-endian formatted data—either using the `lwbrx/stwbrx` instructions or by manipulating the data before writing to and after reading from `CFG_DATA`.

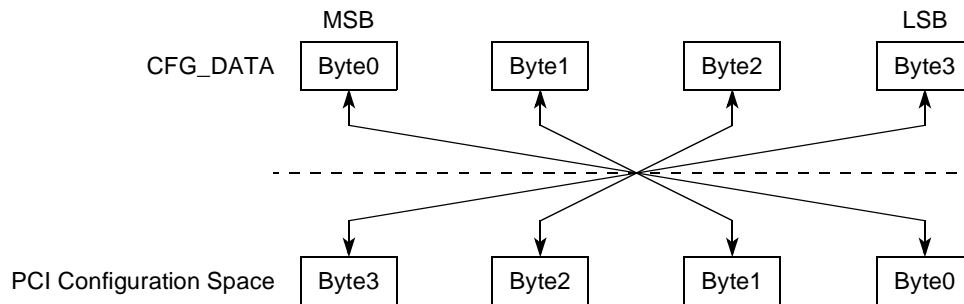


Figure 13-57. `CFG_DATA` Byte Ordering

13.5 Initialization/Application Information

The following sections describe initialization sequences for host and agent modes.

13.5.1 Initialization Sequence for Host Mode

The following sequence must be followed in host mode:

1. Enable PCI output clocks and select desired frequency ratios. See [Section 4.4.1, “Clocking in PCI Host Mode.”](#)
2. Wait for at least 1 ms to enable stable clocks into agent devices
3. Deactivate `PCI_RESET_OUT` signal for PCI. See [Table 13-3](#) for more information on `PCI_RESET_OUT` signal.
4. Wait for at least 1 ms to enable devices to complete the powerup sequence.
5. Configure PCI internal registers and PCI agents to desired modes of operation

13.5.2 Initialization Sequence for Agent Mode

The following sequence must be followed in agent mode:

1. Optionally initialize subsystem vendor ID/device ID
 - a) Initialize PCI inbound window size in `PIWAR[1:3]` desired window size
 - b) Unlock configuration lock in PCI function configuration register



Chapter 14

Security Engine (SEC) 2.4

This chapter describes the functionality of the device's integrated security engine (SEC 2.4). It addresses the following topics:

- [Section 14.1, “SEC Overview”](#)
- [Section 14.2, “Architecture Overview”](#)
- [Section 14.3, “Configuration of Internal Memory Space”](#)
- [Section 14.4, “Descriptor Overview”](#)
- [Section 14.5, “Execution Units”](#)
- [Section 14.6, “Crypto-Channels”](#)
- [Section 14.7, “Controller”](#)
- [Section 14.8, “Bus Interface”](#)
- [Section 14.9, “Power-Saving Mode”](#)

14.1 SEC Overview

The SEC 2.4 is designed to offload computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the device processor core. It is optimized to process all the algorithms associated with IPsec, IKE, SSL/TLS, iSCSI, SRTP, and 802.11i. The SEC 2.4 is derived from integrated security cores found in other members of the PowerQUICC family, including SEC 1.0, the version implemented in the MPC8272/MPC8248.

The security engine's execution units (EUs) and primary features include the following:

- PKEU—Public key execution unit that supports the following:
 - RSA and Diffie-Hellman algorithms
 - Programmable field size up to 2048 bits
 - Elliptic curve cryptography
 - F_{2^m} and $F(p)$ modes
 - Programmable field size up to 511 bits
- DEU—Data encryption standard execution unit
 - DES, 3DES algorithms
 - Two key (K1, K2) or three key (K1, K2, K3) for 3DES
 - ECB and CBC modes for both DES and 3DES
- AESU—Advanced encryption standard unit
 - Implements the Rijndael symmetric-key cipher

- ECB, CBC, CCM, and counter modes
- 128-, 192-, 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-, 224-, or 256-bit message digest
 - MD5 with 128-bit message digest
 - HMAC with either algorithm
- RNG—Random number generator
- XOR parity generation accelerator for RAID applications
- Master/slave logic, with DMA capability
 - 32-bit address/64-bit data
 - Master interface allows multiple pipelined requests
 - DMA blocks can be on any byte boundary
- Four channels, each supporting a queue of commands (descriptor pointers)
 - Dynamic assignment of crypto-execution units through an integrated controller
 - 256-byte buffer FIFOs on data input and output paths of each execution unit, with flow control for large data sizes
- Scatter/Gather capability
 - Gather capability enables the SEC 2.4 to concatenate multiple segments of memory when reading input data
 - Similarly, scatter capability enables SEC 2.4 to write to multiple segments of memory when writing output data

14.2 Architecture Overview

The SEC 2.4 (henceforth referred to as SEC) can act as a master on the internal bus. This allows the SEC to alleviate the data movement bottleneck normally associated with slave-only cores. The host processor accesses the SEC through its device drivers, using system memory for data storage. The SEC resides in the peripheral memory map of the processor; therefore, when an application requires cryptographic functions, it simply creates descriptors for the SEC that define the cryptographic function to be performed and the location of the data. The SEC's bus-mastering capability permits the host processor to set up a crypto-channel with a few short register writes, leaving the SEC to perform reads and writes on system memory to complete the required task.

Figure 14-1 shows that the SEC communicates with other modules via the internal bus.

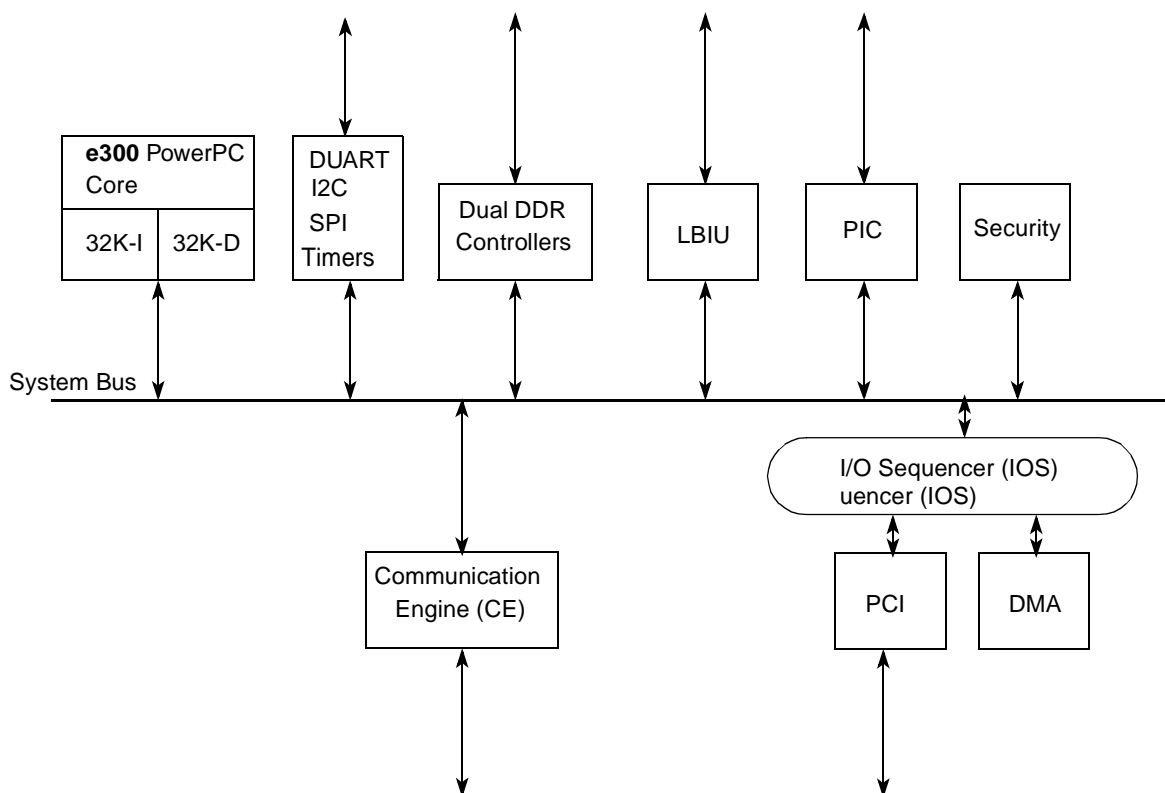


Figure 14-1. SEC Connected to MPC8360E Internal Bus

A block diagram of the SEC internal architecture is shown in Figure 14-2. The bus interface module is designed to transfer 64-bit words between the bus and any register inside the SEC.

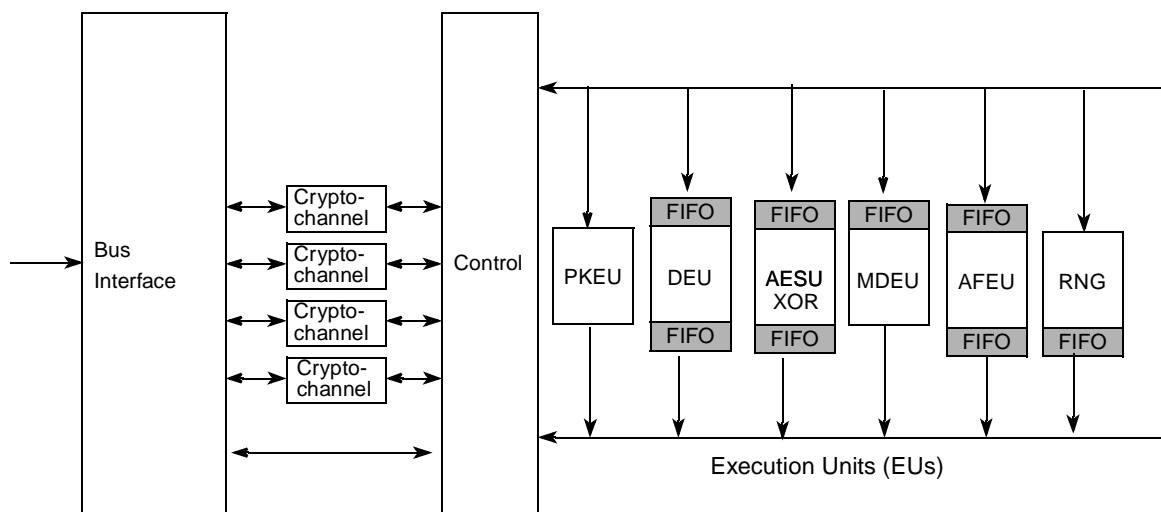


Figure 14-2. SEC Functional Modules

An operation begins when the host writes a descriptor pointer to the fetch FIFO in one of the four SEC channels. From this point on, the channel directs the sequence of operations. The channel uses the descriptor pointer to read the descriptor, then decodes the first word of the descriptor to determine the operation to be performed to select 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, and data from locations specified in the rest of the descriptor. The controller satisfies the requests by making requests to the master interface following 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 the controller to write data from the output FIFOs and registers back to system memory through the master/slave interface.

For most packets, the entire payload is too long to fit in an execution unit’s 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’s output FIFO.

14.2.1 Data Packet Descriptors

As a crypto-acceleration block, the SEC controller has been designed for easy use and integration with existing systems and software. All cryptographic functions are accessible through descriptors. A descriptor specifies a cryptographic function to be performed and contains pointers to all necessary input data and to the places where output data is to be written. Some descriptor types perform multiple functions to facilitate particular protocols. A data packet descriptor is diagrammed in [Table 14-1](#).

Table 14-1. Example Data Packet Descriptor

Field Name	Value/Type	Description
DPD_DES_CTX_CRYPT	TBD	Representative header for DES using context to encrypt
LEN_CTXIN PTR_CTXIN	Length Pointer	Number of bytes to be written Pointer to context (IV) to be written into DES engine
LEN_KEY PTR_KEY	Length Pointer	Number of bytes in key Pointer to block cipher key
LEN_DATAIN PTR_DATAIN	Length Pointer	Number of bytes of data to be ciphered Pointer to data to perform cipher upon
LEN_DATAOUT PTR_DATAOUT	Length Pointer	Number of bytes of data after ciphering Pointer to location where cipher output is to be written
LEN_CTXOUT PTR_CTXOUT	Length Pointer	Length of output context (IV) Pointer to location where altered context is to be written
Null length Null pointer	Length Pointer	Zeros for fixed length descriptor filter Zeros for fixed length descriptor filter
Null length Null pointer	Length Pointer	Zeros for fixed length descriptor filter Zeros for fixed length descriptor filter

Each descriptor contains eight long-words (64 bits each), consisting of the following:

- One long-word of header—The header describes the required services and encodes information indicating which EUs to use and which modes to set. It also indicates if notification should be sent to the host when the descriptor operation is complete.
- Seven long-words containing pointers and lengths used to locate input or output data. Each pointer can either point directly to the data, or can point to a link table that lists a set of data segments to be concatenated.

For more information, refer to [Section 14.4, “Descriptor Overview.”](#)

14.2.2 Execution Units (EUs)

Execution unit (EU) is the generic term for a functional block that performs the mathematical permutations required by protocols used in cryptographic processing. The EUs are compatible with IPsec, IKE, SSL/TLS, iSCSI, SRTP, and 802.11i processing, and can work together to perform high-level cryptographic tasks. The SEC’s execution units are as follows:

- 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
- AFEU—For performing RC-compatible stream cipher symmetric-key cryptography
- AESU—For performing the advanced encryption standard algorithm and XOR acceleration
- MDEU—For performing security hashing using MD-5, SHA-1, or SHA-256
- RNG—For random number generation

Each EU is described in detail in [Section 14.5, “Execution Units.”](#)

14.2.2.1 Public Key Execution Unit (PKEU)

The PKEU can perform many advanced mathematical functions to support both RSA and ECC public-key cryptographic algorithms. ECC is supported in both F_2^m (polynomial-basis) and $F(p)$ modes. This EU supports all levels of functions to assist the host microprocessor to perform its desired cryptographic task. For example, at the highest level, the accelerator performs modular exponentiations to support RSA and performs point multiplies to support ECC. At the lower levels, the PKEU can perform simple operations such as modular multiplies. For more information, refer to [Section 14.5.1, “Public Key Execution Unit \(PKEU\).”](#)

14.2.2.1.1 Elliptic Curve Operations

The PKEU has its own data and control units, including a general-purpose register file in the programmable-size arithmetic unit. The field or modulus size can be programmed to any value between 160 and 512 bits in programmable increments of 8, with each value i supporting all actual field sizes from $8i - 7$ to $8i$. The result is hardware supporting a wide range of cryptographic security. Larger field/modulus sizes result in greater security but lower performance; processing time is determined by field or modulus size. For example, a field size of 160 is roughly equivalent to the security provided by 1024 bit RSA. A field size set to 208 roughly equates to 2048 bits of RSA security.

The PKEU contains routines implementing the atomic functions for elliptic curve processing—point arithmetic and finite field arithmetic. The point operations (multiplication, addition and doubling) involve one or more finite field operations, which are addition, multiplication, inverse, and squaring. Point add and double each use all four finite field operations. Similarly, point multiplication uses all EC point operations as well as the finite field operations. All these functions are supported both in modular arithmetic and polynomial basis finite fields.

14.2.2.1.2 Modular Exponentiation Operations

The PKEU can also perform ordinary integer modulo arithmetic. This arithmetic is an integral part of the RSA public key algorithm; however, it can also play a role in the generation of ECC digital signatures and Diffie-Hellman key exchanges.

Modular arithmetic functions supported by the SEC’s PKEU include the following:

- $R^2 \bmod N$
- $A^E \bmod N$
- $(A \times B) R^{-1} \bmod N$
- $(A \times B) R^{-2} \bmod N$
- $(A + B) \bmod N$
- $(A - B) \bmod N$

Where the following variable definitions: $A' = AR \bmod N$, N is the modulus vector, A and B are input vectors, E is the exponent vector, R is 2^s , where s is the bit length of the N vector rounded up to the nearest multiple of 32.

The PKEU can perform modular arithmetic on operands up to 2048 bits in length. The modulus must be larger than or equal to 97 bits (13 bytes). This is not seen as a limitation since for sizes smaller than this, no useful cryptographic application exists. Furthermore, for small data sizes the overhead of using a hardware accelerator would not be justified. The PKEU uses the Montgomery modular multiplication algorithm to perform core functions. The addition and subtraction functions exist to help support known methods of the Chinese remainder theorem (CRT) for efficient implementation of the RSA algorithm.

14.2.2.2 Data Encryption Standard Execution Unit (DEU)

The DES execution unit (DEU) performs bulk data encryption/decryption, in compliance with the data encryption standard algorithm (ANSI x3.92). The DEU can also compute 3DES, an extension of the DES algorithm in which each 64-bit input block is processed three times. The SEC supports 2-key ($K1=K3$) or 3-key 3DES.

The DEU operates by permuting 64-bit data blocks with a shared 56-bit key and an initialization vector (IV). The SEC supports two modes of operation: electronic code book (ECB) and cipher block chaining (CBC).

For more information, refer to [Section 14.5.2, “Data Encryption Standard Execution Unit \(DEU\).”](#)

14.2.2.3 ARC Four Execution Unit (AFEU)

The AFEU accelerates a bulk encryption algorithm compatible with the RC4 stream cipher from RSA Security, Inc. The algorithm is byte-oriented, meaning a byte of plain text is encrypted with a key to produce a byte of ciphertext. The key is variable length and the AFEU supports key lengths from 8 to 128 bits (in byte increments), providing a wide range of security strengths. ARC4 is a symmetric algorithm, meaning each of the two communicating parties share the same key.

For more information, refer to [Section 14.5.3, “ARC Four Execution Unit \(AFEU\).”](#)

14.2.2.4 Advanced Encryption Standard Execution Unit (AESU)

The AESU is used to accelerate bulk data encryption/decryption in compliance with the Advanced Encryption Standard (Rijndael) algorithm. The AESU executes on 128-bit blocks with a choice of key sizes: 128, 192, or 256 bits.

AESA is a symmetric-key algorithm; the sender and receiver use the same key for both encryption and decryption. The session key and IV are supplied to the AESU module prior to encryption. The processor supplies data to the module that is processed as a 128-bit input. The AESU operates in ECB, CBC, CTR, and CCM modes.

The AESU is also used for performing the eXclusive OR (XOR) operation used to generate parity data for RAID storage applications. When operating in this mode, no session keys are involved, and the AESU XORs up to 3 data streams at a time to produce parity data.

For more information, refer to [Section 14.5.6, “Advanced Encryption Standard Execution Units \(AESU\).”](#)

14.2.2.5 Message Digest Execution Unit (MDEU)

The MDEU computes a single message digest (or hash or integrity check) value of all the data presented on the input bus, using either the MD5, SHA-1 or SHA-256 algorithms for bulk data hashing. With any hash algorithm, the larger message is mapped onto a smaller output space; therefore, collisions are possible, albeit not probable. The 160-bit hash value is a sufficiently large space such that collisions are extremely rare. The security of the hash function is based on the difficulty of locating collisions. That is, it is computationally infeasible to construct two distinct but similar messages that produce the same hash output.

- The MD5 generates a 128-bit hash; the algorithm is specified in RFC 1321.
- SHA-1 is a 160-bit hash function, specified by the ANSI X9.30-2 and FIPS 180-1 standards.
- SHA-224 and SHA-256 are cryptographic hash functions that provide integrity protection against collision attacks.
- The MDEU also supports HMAC computations, as specified in RFC 2104.

For more information, refer to [Section 14.5.4, “Message Digest Execution Unit \(MDEU\).”](#)

14.2.2.6 Random Number Generator (RNG)

The RNG is a functional block capable of generating 64-bit random numbers. It is designed to comply with FIPS 140-1 standards for randomness and non-determinism.

Because many cryptographic algorithms use random numbers as a source for generating a secret value (a nonce), it is desirable to have a private RNG for use by the SEC. The anonymity of each random number must be maintained, as well as the unpredictability of the next random number. The FIPS-140 ‘common criteria’ compliant private RNG allows the system to develop random challenges or random secret keys. The secret key can thus remain hidden from even the high-level application code, providing an added measure of physical security.

For more information, refer to [Section 14.5.5, “Random Number Generator \(RNG\).”](#)

14.2.3 Crypto-Channels

The SEC includes four crypto-channels that manage data and EU function. Each channel consists of the following:

- A fetch FIFO, which holds a queue of pointers to descriptors waiting to be serviced
- A configuration register, which allows the user a number of options for SEC event signaling.
- Control registers containing information about the transaction in process
- A status register containing an indication of the last unfulfilled bus request
- A descriptor buffer memory used to store the active descriptor
- Scatter and gather link table buffer memory used to store the active link table

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 EU(s) from the controller.
2. Wait for the controller to grant access to the required EU(s).
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 IV or other internal EU state that can be read out or loaded in. ‘Text-data’ refers to plaintext or ciphertext to be operated on.
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 3.
9. Release the EUs. (Note that in previous Freescale security co-processors, it was possible to reserve an EU for use on multiple descriptors. With the added capabilities in SEC 2.4, such ‘static’ assignments are no longer necessary and are not supported. EUs are always released at the completion of a descriptor.)
10. If ‘done notification’ is enabled in the descriptor header, perform this notification.

The channel can signal to the host that it is done with a descriptor via interrupt or by a writeback of the descriptor header into host memory. In the case of writeback, the value written back is identical to the header that was read, with the exception that a done field is set to all 1s. In addition, the channel can be configured to write back other status fields that indicate the result of ICV checking (if any). The user can opt to do this signaling at end of every descriptor, or at end of selected descriptors. For more about configuring signaling, see [Section 14.6.1.1, “Crypto-Channel Configuration Register \(CCCR\).”](#)

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 the primary EU, and the other as the secondary EU. The primary EU receives its data from memory through the controller, and the secondary EU receives its data by ‘snooping’ the SEC buses.

There are two types of snooping.

- Input data can be fed to the primary EU and the same input data 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.’

In the SEC, the secondary EU is always the MDEU.

For more information, refer to [Section 14.6, “Crypto-Channels.”](#)

14.2.4 SEC Controller

The SEC controller manages on-chip resources, including the individual execution units (EUs), FIFOs, the master/slave interface, and the internal buses that connect all the various modules. The controller receives service requests from the master/slave interface and various crypto-channels, and schedules the required activities. The controller can configure each of the on-chip resources in two modes:

- Host-controlled access—The host is directly responsible for all data movement into and out of the resource. This mode is typically only used in debug.
- Channel-controlled access—A channel can request a particular service from any available execution unit. This is the normal operating condition.

The system bus interface and access to system memory are critical factors in performance, and the 64-bit master/slave interface of the SEC controller allows it to achieve performance unattainable on secondary buses.

14.2.4.1 Host-Controlled Access

All execution units (EU) can be used entirely through register read/write access. The SEC operates as a slave, and the host must target write the information typically provided through the descriptor into the appropriate registers and FIFOs of the SEC. This mode is more CPU intensive, and requires a great deal of familiarity with the SEC registers. It is recommended that host-controlled access be used only for operations using a single EU, and for debug purposes.

For more information, refer to [Section 14.7, “Controller.”](#)

14.2.4.2 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.

14.2.5 Bus Interface

The master/slave interface manages communication between the SEC’s internal execution units and the device internal bus. All on-chip resources are memory mapped, and the slave accesses to the SEC may be addressed on byte boundaries. The SEC performs initiator reads on byte boundaries and internally adjusts the data to place on word boundaries as appropriate. Access to system memory is a critical factor in performance, and the 64-bit master/slave interface of the SEC allows it to achieve performance unattainable on secondary buses.

For more information, refer to [Section 14.8, “Bus Interface.”](#)

14.3 Configuration of Internal Memory Space

[Table 14-2](#) shows the base address map, while [Table 14-3](#) provides the address map, including all registers in the execution units. The 18-bit SEC address bus value is shown. These address values are offsets from the internal memory mapped registers base address, as programmed in the IMMRBAR. See [Section 2.3, “Complete IMMR Map,”](#) for more information.

Note that these tables show modulo-8 addresses; the three least significant address bits that are used to select bytes within 64-bit-words are not shown.

Table 14-2. SEC Base Address Map

Offset	Module	Description	Type	Section/Page
0x3_0000–0x3_0FFF	—	Reserved	—	—
0x3_1000–0x3_10FF	Controller	Arbiter/controller control register space	Resource control	14.7/14-93
0x3_1100–0x3_11FF	Channel_1	Crypto-channel 1	Data control	14.6/14-81
0x3_1200–0x3_12FF	Channel_2	Crypto-channel 2		
0x3_1300–0x3_13FF	Channel_3	Crypto-channel 3		
0x3_1400–0x3_14FF	Channel_4	Crypto-channel 4		

Table 14-2. SEC Base Address Map (continued)

Offset	Module	Description	Type	Section/Page
0x3_2000–0x3_2FFF	DEU	DES/3DES execution unit	Crypto EU	14.5.2/14-34
0x3_4000–0x3_4FFF	AESU	AES execution unit		14.5.6/14-68
0x3_6000–0x3_6FFF	MDEU	Message digest execution unit		14.5.4/14-51
0x3_8000–0x3_8FFF	AFEU	ARC four execution unit		14.5.3/14-42
0x3_A000–0x3_AFFF	RNG	Random number generator		14.5.5/14-63
0x3_C000–0x3_CFFF	PKEU	Public key execution unit		14.5.1/14-26

Table 14-3 shows the system address map showing all functional registers. Undefined 4-byte address spaces within offset 0x000–0xFFFF are reserved.

Table 14-3. SEC Address Map

Offset	Register	Access	Reset	Section/Page
Controller Registers				
0x3_1008	IMR—Interrupt mask register	R/W	All zeros	14.7.2.1/14-94
0x3_1010	ISR—Interrupt status register	R	All zeros	14.7.2.2/14-96
0x3_1018	ICR—Interrupt clear register	W	All zeros	14.7.2.3/14-96
0x3_1020	ID—Identification register	R	All zeros	14.7.2.4/14-98
0x3_1028	EUASR—EU assignment status register	R	0xF0F0_F0F0_00FF_F0F0	14.7.2/14-93
0x3_1030	MCR—Master control register	R/W	All zeros	14.7.2.6/14-99
Channel 1				
0x3_1108	CCCR1—Crypto-channel 1 configuration register	R/W	All zeros	14.6.1.1/14-82
0x3_1110	CCPSR1—Crypto-channel 1 pointer status register	R	0x0000_0000_0000_0007	14.6.1.2/14-85
0x3_1140	CDPR1—Crypto-channel 1 current descriptor pointer register	R	All zeros	14.6.1.3/14-90
0x3_1148	FF1—Crypto-channel 1 fetch FIFO address register	W	All zeros	14.6.1.4/14-90
0x3_1180–0x3_11BF	DBn—Crypto-channel 1 descriptor buffers[0–7]	R	All zeros	14.6.1.5/14-91
Channel 2				
0x3_1208	CCCR2—Crypto-channel 2 configuration register	R/W	All zeros	14.6.1.1/14-82
0x3_1210	CCPSR2—Crypto-channel 2 pointer status register	R	0x0000_0000_0000_0007	14.6.1.2/14-85
0x3_1240	CDPR2—Crypto-channel 2 current descriptor pointer register	R	All zeros	14.6.1.3/14-90
0x3_1248	FF2—Crypto-channel 2 fetch FIFO address register	W	All zeros	14.6.1.4/14-90
0x3_1280–0x3_12BF	DBn—Crypto-channel 2 descriptor buffers[0–7]	R	All zeros	14.6.1.5/14-91

Table 14-3. SEC Address Map (continued)

Offset	Register	Access	Reset	Section/Page
Channel 3				
0x3_1308	CCCR3—Crypto-channel 3 configuration register	R/W	All zeros	14.6.1.1/14-82
0x3_1310	CCPSR3—Crypto-channel 3 pointer status register	R	0x0000_0000_0000_0007	14.6.1.2/14-85
0x3_1340	CDPR3—Crypto-channel 3 current descriptor pointer register	R	All zeros	14.6.1.3/14-90
0x3_1348	FF3—Crypto-channel 3 fetch FIFO address register	W	All zeros	14.6.1.4/14-90
0x3_1380– 0x3_13BF	DBn—Crypto-channel 3 descriptor buffers[0–7]	R	All zeros	14.6.1.5/14-91
Channel 4				
0x3_1408	CCCR4—Crypto-channel 4 configuration register	R/W	All zeros	14.6.1.1/14-82
0x3_1410	CCPSR4—Crypto-channel 4 pointer status register	R	0x0000_0000_0000_0007	14.6.1.2/14-85
0x3_1440	CDPR4—Crypto-channel 4 current descriptor pointer register	R	All zeros	14.6.1.3/14-90
0x3_1448	FF4—Crypto-channel 4 fetch FIFO address register	W	All zeros	14.6.1.4/14-90
0x3_1480– 0x3_14BF	DBn—Crypto-channel 4 descriptor buffers[0–7]	R	All zeros	14.6.1.5/14-91
Data Encryption Standard Execution Unit (DEU)				
0x3_2000	DEUMR—DEU mode register	R/W	All zeros	14.5.2.1/14-35
0x3_2008	DEUKSR—DEU key size register	R/W	All zeros	14.5.2.2/14-36
0x3_2010	DEUDSR—DEU data size register	R/W	All zeros	14.5.2.3/14-36
0x3_2018	DEURCR—DEU reset control register	R/W	All zeros	14.5.2.4/14-37
0x3_2028	DEUSR—DEU status register	R	All zeros	14.5.2.5/14-37
0x3_2030	DEUISR—DEU interrupt status register	R	All zeros	14.5.2.6/14-38
0x3_2038	DEUICR—DEU interrupt control register	R/W	0x0000_0000_0000_3000	14.5.2.7/14-40
0x3_2050	DEUEUG—DEU EU-Go register	W	All zeros	14.5.2.8/14-41
0x3_2100	DEUIV—DEU initialization vector register	R/W	All zeros	14.5.2.9/14-42
0x3_2400	DEUK1—DEU key register 1	W	—	14.5.2.10/14-42
0x3_2408	DEUK2—DEU key register 2	W	—	14.5.2.10/14-42
0x3_2410	DEUK3—DEU key register 3	W	—	14.5.2.10/14-42
0x3_2800– 0x3_2FFF	DEU FIFO	R/W	All zeros	14.5.2.11/14-42
Advanced Encryption Standard Execution Unit (AESU)				
0x3_4000	AESUMR—AESU mode register	R/W	All zeros	14.5.6.1/1414-68
0x3_4008	AESUKSR—AESU key size register	R/W	All zeros	14.5.6.2/1414-71

Table 14-3. SEC Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_4010	AESUDSR—AESU data size register	R/W	All zeros	14.5.6.3/1414-71
0x3_4018	AESURCR—AESU reset control register	R/W	All zeros	14.5.6.4/14-72
0x3_4028	AESUSR—AESU status register	R	All zeros	14.5.6.5/14-73
0x3_4030	AESUISR—AESU interrupt status register	R	All zeros	14.5.6.6/14-74
0x3_4038	AESUICR—AESU interrupt control register	R/W	0x0000_0000_0000_1000	14.5.6.7/14-75
0x3_4050	AESUEMR—AESU end of message register	W	All zeros	14.5.6.8/14-76
0x3_4100	AESU context memory registers	R/W	All zeros	14.5.6.9/14-77
0x3_4400– 0x3_4408	AESU key memory	R/W	All zeros	14.5.6.9.5/14-81
0x3_4800– 0x3_4FFF	AESU FIFO	R/W	All zeros	14.5.6.9.6/14-81
Message Digest Execution Unit (MDEU)				
0x3_6000	MDEUMR—MDEU mode register	R/W	All zeros	14.5.4.1/14-51
0x3_6008	MDEUKSR—MDEU key size register	R/W	All zeros	14.5.4.3/14-55
0x3_6010	MDEUDSR—MDEU data size register	R/W	All zeros	14.5.4.4/14-56
0x3_6018	MDEURCR—MDEU reset control register	R/W	All zeros	14.5.4.5/14-56
0x3_6028	MDEUSR—MDEU status register	R	All zeros	14.5.4.6/14-57
0x3_6030	MDEUISR—MDEU interrupt status register	R	All zeros	14.5.4.7/14-58
0x3_6038	MDEUICR—MDEU interrupt control register	R/W	0x0000_0000_0000_1000	14.5.4.8/14-59
0x3_6040	MDEUICVSR—MDEU ICV size register	R/W	All zeros	14.5.4.9/14-60
0x3_6050	MDEUEUG—MDEU EU-Go register	W	All zeros	14.5.4.10/14-61
0x3_6100– 0x3_6120	MDEU context memory registers	R/W	All zeros	14.5.4.11/14-61
0x3_6400– 0x3_647F	MDEU key memory	W	All zeros	14.5.4.12/14-62
0x3_6800– 0x3_6FFF	MDEU FIFO	W	All zeros	14.5.4.13/14-63
ARC Four Execution Unit (AFEU)				
0x3_8000	AFEUMR—AFEU mode register	R/W	All zeros	14.5.3.1/14-43
0x3_808	AFEUKSR—AFEU key size register	R/W	All zeros	14.5.3.3/14-44
0x3_8010	AFEUDSR—AFEU data size register	R/W	All zeros	14.5.3.4/14-45
0x3_8018	AFEURCR—AFEU reset control register	R/W	All zeros	14.5.3.5/14-46
0x3_8028	AFEUSR—AFEU status register	R	All zeros	14.5.3.6/14-46
0x3_8030	AFEUISR—AFEU interrupt status register	R	All zeros	14.5.3.7/14-47

Table 14-3. SEC Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_8038	AFEUICR—AFEU interrupt control register	R/W	0x0000_0000_0000_1000	14.5.3.8/14-49
0x3_8050	AFEUEMR—AFEU end of message register	W	All zeros	14.5.3.9/14-50
0x3_8100– 0x3_81FF	AFEU context memory registers	R/W	All zeros	14.5.3.10.1/14-50
0x3_8200	AFEU context memory pointers	R/W	All zeros	14.5.3.10.2/14-51
0x3_8400	AFEUK0—AFEU key register 0	W	—	14.5.3.11/14-51
0x3_848	AFEUK1—AFEU key register 1	W	—	14.5.3.11/14-51
0x3_8800– 0x3_8FFF	AFEU FIFO	R/W	All zeros	14.5.3.11.1/14-51
Random Number Generator (RNG)				
0x3_A000	RNGMR—RNG mode register	R/W	All zeros	14.5.5.1/14-63
0x3_A010	RNGDSR—RNG data size register	R/W	All zeros	14.5.5.2/14-64
0x3_A018	RNGRCR—RNG reset control register	R/W	All zeros	14.5.5.3/14-65
0x3_A028	RNGSR—RNG status register	R	All zeros	14.5.5.4/14-65
0x3_A030	RNGISR—RNG interrupt status register	R	All zeros	14.5.5.5/14-66
0x3_A038	RNGICR—RNG interrupt control register	R/W	0x0000_0000_0000_1000	14.5.5.6/14-67
0x3_A050	RNGEUG—RNG EU-Go register	W	All zeros	14.5.5.7/14-68
0x3_A800– 0x3_AFFF	RNG FIFO	R	All zeros	14.5.5.8/14-68
Public Key Execution Unit (PKEU)				
0x3_C000	PKEUMR—PKEU mode register	R/W	All zeros	14.5.1.1/14-26
0x3_C008	PKEUKSR—PKEU key size register	R/W	All zeros	14.5.1.2/14-28
0x3_C010	PKEUDSR—PKEU data size register	R/W	All zeros	14.5.1.3/14-28
0x3_C018	PKEURCR—PKEU reset control register	R/W	All zeros	14.5.1.5/14-29
0x3_C028	PKEUSR—PKEU status register	R	All zeros	14.5.1.6/14-30
0x3_C030	PKEUISR—PKEU interrupt status register	R	All zeros	14.5.1.7/14-31
0x3_C038	PKEUICR—PKEU interrupt control register	R/W	0x0000_0000_0000_1000	14.5.1.8/14-32
0x3_C040	PKEUABS—PKEU AB size register	R/W	All zeros	14.5.1.3/14-28
0x3_C050	PKEUEUG—PKEU EU-Go	W	All zeros	14.5.1.9/14-33

Table 14-3. SEC Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_C200– 0x3_C23F	PKEU parameter memory A0	R/W	All zeros	14.5.1.10/14-34
0x3_C240– 0x3_C27F	PKEU parameter memory A1	R/W	All zeros	
0x3_C280– 0x3_C2BF	PKEU parameter memory A2	R/W	All zeros	
0x3_C2C0– 0x3_C2FF	PKEU parameter memory A3	R/W	All zeros	
0x3_C300– 0x3_C33F	PKEU parameter memory B0	R/W	All zeros	
0x3_C340– 0x3_C37F	PKEU parameter memory B1	R/W	All zeros	
0x3_C380– 0x3_C3BF	PKEU parameter memory B2	R/W	All zeros	
0x3_C3C0– 0x3_C3FF	PKEU parameter memory B3	R/W	All zeros	
0x3_C400– 0x3_C4FF	PKEU parameter memory E	W	All zeros	
0x3_C800– 0x3_C8FF	PKEU parameter memory N	R/W	All zeros	

14.4 Descriptor Overview

The host processor maintains a record of current secure sessions and the corresponding keys and contexts of those sessions. Once the host 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 host 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 host processor.

For test purposes, it is also possible for the host to write keys, context, and text-data directly to the SEC, using SEC's host-controlled mode. This method avoids use of descriptors.

14.4.1 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, i.e. eight 64-bit words (referred to as dwords). A descriptor consists of one header dword and seven pointer dwords, as shown in [Figure 14-3](#).

The header dword specifies the security operation to be performed, the execution unit(s) needed, and the modes for each execution unit. The pointer dwords, all of which have the same format, contain pointer and

	0	15	16	17	23	24	31	32	63	
Header Dword	Header							Reserved		
Pointer Dword 0	Length0	J0	Extent0	—		Pointer0				
Pointer Dword 1	Length1	J1	Extent1	—		Pointer1				
Pointer Dword 2	Length2	J2	Extent2	—		Pointer2				
Pointer Dword 3	Length3	J3	Extent3	—		Pointer3				
Pointer Dword 4	Length4	J4	Extent4	—		Pointer4				
Pointer Dword 5	Length5	J5	Extent5	—		Pointer5				
Pointer Dword 6	Length6	J6	Extent6	—		Pointer6				

Figure 14-3. Descriptor Format

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 double word 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.

14.4.2 Descriptor Format—Header Dword

Descriptors are created by the host to guide the SEC through required cryptographic operations. The descriptor header defines the operations to be performed, the mode for each operation, and internal addressing used by the controller and channel for internal data movement. The SEC device drivers allow the host to create proper headers for each cryptographic operation.

	0	3	4	11	12	15	16	23	24	28	29	30	31
Field	OP_0			OP_1				DESC_TYPE			—	DIR	DN
	EU_SEL0	MODE0		EU_SEL1	MODE1								
Field	—												

Figure 14-4. Header Dword

Table 14-4. Header Dword Bit Definitions

Bits	Name	Description
OP_0		
0–3	EU_SEL0	Primary EU select: See Section 14.4.2.1, “Selecting Execution Units—EU_SEL0 and EU_SEL1,” for possible values.
4–11	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 56–63 of the mode register in the selected EU.
OP_1		
12–15	EU_SEL1	Secondary EU select: See Section 14.4.2.1, “Selecting Execution Units—EU_SEL0 and EU_SEL1,” for possible values.
16–23	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 56–63 of the mode register in the selected EU.
24–28	DESC_TYPE	Descriptor type. This field, along with the DIR field, 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 attributes determined include the direction of data flow for each data block, which EU (primary or secondary) is accessed, what snooping options are used, and which internal EU addresses are accessed. See Section 14.4.2.2, “Selecting Descriptor Type—DESC_TYPE,” for possible values.
29	—	Reserved.
30	DIR	Direction of overall data flow 0 Outbound 1 Inbound This field, along with the DESC_TYPE field, helps determine the sequence of actions to be performed by the channel and selected EUs.
31	DN	Done notification 0 No done notification. 1 Signal done to the host on completion of this descriptor. The done notification can take the form of an interrupt, or a modified header writeback, 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. When writeback is used, the value written back is 0xFF in bits 0–7 of this long word.

14.4.2.1 Selecting Execution Units—EU_SEL0 and EU_SEL1

[Table 14-5](#) shows the values for EU_SEL0 and EU_SEL1 in the descriptor header. The following rules govern the choices for these fields:

1. EU_SEL0 values of 0000 (no EU selected) or 0111–1111 (reserved) results in an unrecognized header error condition during processing of the descriptor header.
2. The only valid choices for EU_SEL1 are 0000 (No EU selected) or 0011 (MDEU). Any other choice causes an unrecognized header error condition.
3. If EU_SEL1 is 0011 (MDEU), then EU_SEL0 must be 0010 (DEU), 0110 (AESU), or 0001 (AFEU). All other values of EU_SEL0 causes an unrecognized header error condition.

Table 14-5. EU_SEL1 and EU_SEL2 Values

Value (binary)	Selected EU
0000	No EU selected
0001	AFEU
0010	DEU
0011	MDEU
0100	RNG
0101	PKEU
0110	AESU
0111–1110	Reserved
1111	Reserved for header writeback

14.4.2.2 Selecting Descriptor Type—DESC_TYPE

Table 14-6 shows the permissible values for the DESC_TYPE field in the descriptor header. Descriptor types from the SEC 1.0, which have zero in the last bit (xxxx_0), are listed first, followed by new SEC 2.4 types, which have one in the last bit (xxxx_1).

Table 14-6. Descriptor Types

Value (binary)	Descriptor Type	Notes
0000_0	aesu_ctr_nonsnoop	AESU CTR nonsnooping ¹
0001_0	common_nonsnoop_no_afeu	Common, nonsnooping, non-PKEU, non-AFEU ¹
0010_0	hmac_snoop_no_afeu	Snooping, HMAC, non-AFEU ²
0011_0	—	Reserved
0100_0	—	Reserved
0101_0	common_nonsnoop_afeu	Common, nonsnooping, AFEU
0110_0	—	Reserved
0111_0	—	Reserved
0000_1	aesu_ctr_nonsnoop	AESU CTR nonsnooping
0001_1	common_nonsnoop	Common, nonsnooping, non-PKEU, non-AFEU
0010_1	hmac_snoop_no_afeu	Snooping, HMAC, non-AFEU
0011_1	—	Reserved
0100_1	—	Reserved
0101_1	common_nonsnoop_afeu	Common, nonsnooping, AFEU
0110_1	—	Reserved
0111_1	—	Reserved

Table 14-6. Descriptor Types (continued)

Value (binary)	Descriptor Type	Notes
1000_0	pkeu_mm	PKEU-Montgomery Multiplication
1001_0	—	Reserved
1010_0	—	Reserved
1011_0	—	Reserved
1100_0	hmac_snoop_aesu_ctr	AESU CTR hmac snooping ²
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	srtpt	SRTP encryption and hashing
0011_1	pkeu_assemble	pkeu_assemble Elliptic Curve Cryptography
0100_1	pkeu_ptmul	pkeu_ptmul Elliptic Curve Cryptography
0101_1	pkeu_ptadd_dbl	pkeu_ptadd_dbl Elliptic Curve Cryptography
others	—	Reserved
0110_1	—	Reserved
0111_1	—	Reserved
1000_1	tls_ssl_block	TLS/SSL generic block cipher
1001_1	tls_ssl_stream	TLS/SSL generic stream cipher
1010_1	raid_xor	XOR data streams
All others	—	Reserved

¹ 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.

² 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.

For more about descriptor types and the data used for each type, see [Section 14.4.5, “Descriptor Types.”](#)

14.4.3 Descriptor Format—Pointer Dwords

The descriptor contains seven pointer dwords that define where in memory the SEC should access its input and output data parcels. The channel determines how it will use each of the pointer dwords 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. The LENGTH or EXTENT used with any POINTER may be from the same pointer dword or from a different pointer dword in the same descriptor. Although the EXTENT field

exists in each pointer dword of the SEC descriptor, (only the EXTENTS in pointer dwords 3, 4, and 5 are currently in use).

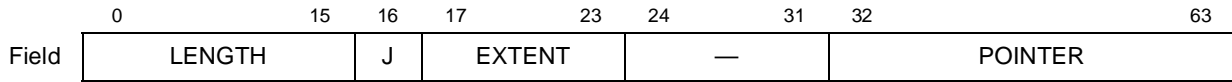


Figure 14-5. Pointer Dword

Table 14-7. Pointer Dword Field Definitions

Bits	Name	Description
0–15	LENGTH	Length. A number of bytes in the range 0 to 65535. The use of this field depends on the descriptor type and direction in the header dword. A value of zero causes the channel to skip this dword.
16	J	Jump. Determines whether to jump to a link table whenever the POINTER field in this same word is used. 0 The POINTER field points to data. 1 The POINTER field points to a link table, and scatter/gather is enabled.
17–23	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 dword.
24–31	—	Reserved
32–63	POINTER	Pointer. A memory address.

On occasion, a descriptor field may not be applicable to the requested service. With seven length/pointer pairs, it is possible that not all descriptor fields are required to load the required keys, context, and text-data. (Some operations, for example, do not require context.) Therefore, when processing descriptors the SEC skips entirely any pointer that has an associated LENGTH of zero.

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 be written to an EU.

If the channel procedure calls for reading a parcel using a nonzero LENGTH field, but the POINTER field is zero, the length value is written to the EU but no data parcel is fetched from the bus.

The J bit in each pointer dword 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 the POINTER should be set 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 the POINTER should be set to point to a link table, and the J bit should be 1. For link table format, see [Section 14.4.4, “Link Table Format.”](#)

Scatter/gather capability is available for all pointer dwords of all descriptor types, with the following exception(s):

- Raid-XOR descriptor type does not allow scatter/gather.

14.4.4 Link Table Format

Link tables implement scatter/gather capability. For gather operations, a link table specifies a list of memory segments that are to be concatenated in the process of assembling data parcels. For scatter operations, a link table specifies a list of memory segments into which the output data should be written. Scatter or gather of a data parcel may be specified by a single link table or by a chain of link tables that are linked together with pointers (see [Figure 14-7](#)).

The link table or chain of link tables accessed through some descriptor POINTER must specify enough memory segments to hold all the data that will be accessed through that pointer. In most cases, only a single data parcel is accessed through a given POINTER, and the chain of link tables specifies just that parcel. In other cases, the descriptor POINTER is used multiple times to access a sequence of 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 14-7](#).

A link table may contain any number of long word entries. There are two kinds of entries, ‘regular’ entries and ‘next’ entries. Each ‘regular’ entry specifies a memory segment by means of a 32-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 and 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). Link tables are illustrated in [Figure 14-7](#).

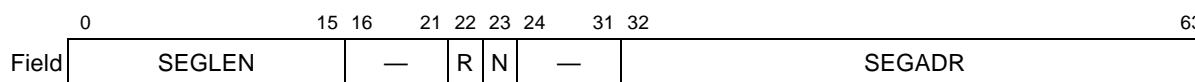


Figure 14-6. Link Table Entry Format

Table 14-8. Link Table Field Definitions

Bits	Name	Description
0–15	SEGLEN	Length. When N=0, a number in the range 1 to 65535, specifying the number of bytes in the memory segment, pointed to by SEGADR. A value of 0 will cause an error bit to be set in the Channel Pointer Status Register—GER for a gather operation or SER for a scatter operation (see 14.6.1.1/14-82). When N=1, must be zero.
16–21	—	Reserved
22	R	Return: When N=0: 0 No special action. 1 This is the last entry in the chain of link tables. If this entry does not specify the right number of bytes to complete the last data parcel, a GER or SER error will be set in the Channel Pointer Status Register (see 14.6.1.1/14-82). When N=1: ignored.

Table 14-8. Link Table Field Definitions (continued)

Bits	Name	Description
23	N	Next 0 No special action. 1 This is the last long word in the current link table. The SEGADR field is the address of the next link table in the chain.
24–31	—	Reserved
32–63	SEGADR	Segment address. A memory address.

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 (SEGLN fields). Otherwise the channel sets the appropriate error bit in the channel pointer status register—GER for a gather error or SER for a scatter error. See [Section 14.6.1.1, “Crypto-Channel Configuration Register \(CCCR\),”](#) for more information.

14.4.4.1 Link Table Example

Figure 14-7 is an example of proper construction of link tables.

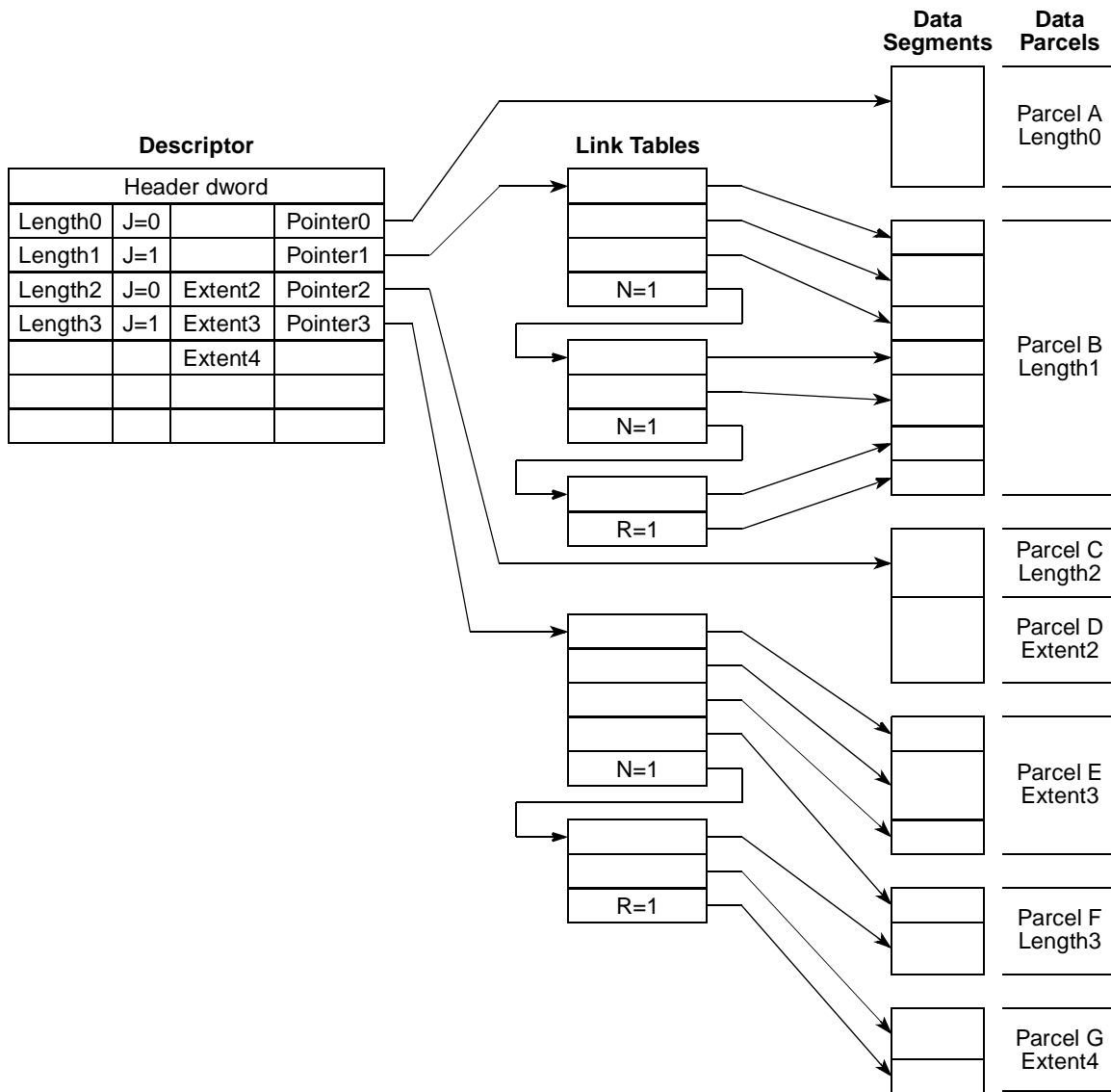


Figure 14-7. Descriptors, Link Tables, and Data Parcels

Figure 14-7 illustrates various ways that a descriptor may specify data parcels:

- The first pointer double word in the descriptor specifies Parcel A using the simplest method: the parcel is specified directly through Pointer0 and Length0.
- The next pointer double word uses a chain of link tables to specify Parcel B. Since J=1, pointer1 is used as the address of a link table. The link table specifies several ‘regular’ entries specifying data segments to be concatenated. The last word of the link table is a next entry indicating that the list continues in the ‘next’ link table. The last entry in the last link table of the chain has the R bit set.
- The last cases illustrate how one pointer in a descriptor can be used to specify multiple parcels. Pointer2 and Length2 specify Parcel C, then Parcel D follows immediately afterwards, with length

specified by Extent2. Pointer3 is used for three data parcels (E, F, and G), this time using link tables.

To demonstrate use of a link table, assume that the current descriptor type calls for the channel to access a data parcel using Pointer3 and Extent3 fields, and assume that J3=1. Due to the J3 value, Pointer3 is not used as a data address but instead used as the address of a link table. The channel begins by reading the first four long words starting at Pointer3 into an internal link table buffer.

Using the first entry of the link table, the channel starts accessing the data parcel by reading SEGLEN bytes beginning at SEGADR. If the required data parcel size (Extent3) is greater than this first SegLen, the channel moves on to the next entry of the link table, and reads SEGLEN bytes starting at SEGADR. While there are more bytes to be read in the data parcel, this process continues. If the channel's link table buffer is exhausted, the channel reads the next four long words of the link table into its link table buffer. If a link table entry is encountered in which the N bit is set, the channel uses the SEGADR field in that word to find the next link table in the chain. The last byte of the required parcel size (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.

14.4.5 Descriptor Types

Table 14-9 shows how the pointer dwords 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.

Additional explanation of the use of certain descriptor types and the meaning of the pointer dwords can be found in the *SEC 2.4 Descriptor Programmer's Guide*.

Table 14-9. Descriptor Pointer Dword Usage

Descriptor Type	Pointer DWORD 1	Pointer DWORD 2	Pointer DWORD 3	Pointer DWORD 4	EXTENT 4	Pointer DWORD 5	EXTENT 5	Pointer DWORD 6	EXTENT 6	Pointer DWORD 7
0000_0	nil	Cipher IV	Cipher Key	In FIFO	nil	Out FIFO	nil	Cipher IV Out	nil	nil
0001_0	nil	Cipher IV	Cipher Key	In FIFO	nil	Out FIFO	nil	Cipher IV Out	nil	nil
0010_0	HMAC Key	HMAC Data	Cipher Key	Cipher IV	nil	In FIFO	nil	Out FIFO	nil	HMAC Out
0011_0	Reserved									
0100_0	Reserved									
0101_0	nil	ARC-4 Context (In FIFO)	ARC-4 Key	In FIFO	nil	Out FIFO	nil	ARC-4 Context (Out FIFO)	nil	nil
0110_0	Reserved									
0111_0	Reserved									
1000_0	N	B	A	E	nil	B Out	nil	nil	nil	nil
1001_0	Reserved									
1010_0	Reserved									
1011_0	Reserved									
1100_0	HMAC Key	HMAC Data	AES Key	AES Ctx	nil	In FIFO	nil	Out FIFO	nil	HMAC Out
1101_0	Reserved									
1110_0	Reserved									
1111_0	Reserved									
0000_1	HMAC Key	HMAC Data	Cipher IV	Cipher Key	nil	In FIFO	nil	Out FIFO	HMAC Out	Cipher IV Out
0001_1	nil	AES-Ctx	AES Key	In FIFO	nil	In FIFO	nil	Out FIFO	nil	AES-Ctx-Out
0010_1	HMAC Key	AES-Ctx	AES Key	In FIFO	In FIFO	Out FIFO	In FIFO	HMAC Out	nil	AES-Ctx-Out
0011_1	A0	A1	A2	A3	nil	B0	nil	B1	nil	'Build'
0100_1	N	E	'Build'	B1 Out	nil	B2 Out	nil	B3 Out	nil	nil
0101_1	N	'Build'	B2	B3	nil	B1 Out	nil	B2 Out	nil	B3 Out
1000_1 Outbound	MAC Key	Cipher IV	Cipher Key	In FIFO	In FIFO	In FIFO	MAC In	Out FIFO	nil	Cipher IV Out
1000_1 Inbound	MAC Key	Cipher IV	Cipher Key	nil	In FIFO	In FIFO	MAC In	Out FIFO	MAC Out	Cipher IV Out
1001_1 Outbound	MAC Key	Cipher IV	Cipher Key	In FIFO	In FIFO	In FIFO	MAC Out	Out FIFO	nil	Cipher IV Out
1001_1 Inbound	MAC Key	Cipher IV	Cipher Key	nil	In FIFO	In FIFO	MAC In	Out FIFO	MAC Out	Cipher IV Out
1010_1	nil	nil	nil	In 1	undefined	In 2	undefined	In 3	undefined	Out
others	Reserved									

Table 14-10 lists the possible mode field values. Depending on the value written to PKEU[Mode] field depends on the routine used. Parameter memories are referred to for the base address, as shown.

Table 14-10. Mode Field Description

Routine Name	Routine Description	Mode [56–63]
RESERVED	Reserved	0x00
CLEARMEMORY	Clear memory	0x01
MOD_EXP	FP: Exponentiate mod N and deconvert from Montgomery format	0x02
MOD_R2MODN	FP: Compute Montgomery converter (R2 mod N)	0x03
MOD_RRMODP	FP: Compute Montgomery converter for Chinese Remainder Theorem (RnRp mod N)	0x04
EC_FP_AFF_PTMULT	FP EC: Multiply key times point in affine coordinates	0x05
EC_F2M_AFF_PTMULT	F2m EC: Multiply key times point in affine coordinates	0x06
EC_FP_PROJ_PTMULT	FP EC: Multiply key times point in projective coordinates	0x07
EC_F2M_PROJ_PTMULT	F2m EC: Multiply key times point in projective coordinates	0x08
EC_FP_ADD	FP EC: Add two points in projective coordinates	0x09
EC_FP_DOUBLE	FP EC: Double a point in projective coordinates	0x0A
EC_F2M_ADD	F2m EC: Add two points in projective coordinates	0x0B
EC_F2M_DOUBLE	F2m EC: Double a point in projective coordinates	0x0C
F2M_R2	F2m: Compute Montgomery converter (R2 mod N)	0x0D
F2M_INV ¹	F2m: Invert mod N	0x0E
MOD_INV ²	FP: Invert mod N	0x0F
MOD_ADD	FP: Add mod N	0x10
MOD_SUB	FP: Subtract mod N	0x20
MOD_MULT1_MONT	FP: Multiply mod N in Montgomery format	0x30
MOD_MULT2_DECONV	FP: Multiply mod N and deconvert from Montgomery format	0x40
F2M_ADD	F2m: Add mod N	0x50
F2M_MULT1_MONT	F2m: Multiply mod N in Montgomery format	0x60
F2M_MULT2_DECONV	F2m: Multiply mod N and deconvert from Montgomery format	0x70
RSA_SSTEP	FP: Exponentiate mod N (combines MOD_R2MODN, F2M_MULT1_MONT, and MOD_EXP)	0x80
SPK_BUILD	Build PK data structure	0xFF

¹ F2M_INV returns incorrect results for modulus sizes greater than 480 bits.

² MOD_INV returns incorrect results for modulus sizes greater than 480 bits.

14.5.1.2 PKEU Key Size Register (PKEUKSR)

The PKEU key size register, shown in [Figure 14-9](#), reflects the number of significant bytes to be used from PKEU parameter memory E in performing modular exponentiation or elliptic curve point multiplication. Note that leading zeros are not significant and are not considered part of the key (modulus) size. 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 will cause a key size error in the PKEU interrupt status register (PKEUISR[KSE] is set).

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated Interrupt Control Register prior to performing debug operations.



Figure 14-9. PKEU Key Size Register

14.5.1.3 PKEU AB Size Register (PKEUABS)

PKEU ABS ([Figure 14-10](#)) represents the operand size for the specific operands whenever it is required. The unit of the value written into PKEUABS[AB Size] is in bits, even though internally the PKEU imposes a 32-bit alignment. Any data beyond the number of bits in PKEUABS, either in A- and B-ram (operands) will be ignored. No error checking is performed whether the operand sizes are greater than the prime modulus or the field size and this may cause a wrong result. In other words, it is assumed that operands are modulo reduced before being written into the PKEU. Hence, PKEUABS[AB Size] must be less than or equal to data size for a correct result. If PKEUABS is modified during processing, an error will be generated.

An illegal data size error is generated as follows:

- For all non-ECC routines, a data size > 256 generates an illegal data size error.
- For all ECC routines, a data size > 64 generates an illegal data size error.

Setting PKEUABS[AB Size] = 0 (either intentionally or by ignoring it and not writing it at all) generates an illegal size error, except for routines such as the CLEAR_MEM routine that do not require an A or B operand.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated interrupt control register before performing debug operations.

Table 14-11 describes PKEURCR fields.

Table 14-11. PKEURCR Field Descriptions

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes PKEU interrupts signaling DONE and ERROR to be reset. It further resets the state of the PKEU interrupt status register. 0 Do not reset interrupt logic. 1 Reset interrupt logic.
62	MI	Module initialization. Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. This module initialization includes execution of an initialization routine, completion of which is indicated by the RD (reset done) bit in the PKEU status register (Section 14.5.1.6, “PKEU Status Register (PKEUSR)”) 0 Do not reset most of PKEU. 1 Reset most of PKEU.
63	SR	SW reset. Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for the PKEU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the PKEU will enter a routine to perform proper initialization of the parameter memories. The RD (reset done) bit in the PKEU status register will indicate when this initialization routine is complete. (See Section 14.5.1.6, “PKEU Status Register (PKEUSR).”) 0 Do not reset full PKEU. 1 Full PKEU reset.

14.5.1.6 PKEU Status Register (PKEUSR)

PKEUSR fields reflect the state of PKEU internal fields. PKEUSR is read only. Writing to it results in an address error being reflected in the PKEU interrupt status register (PKEUISR[AE] is set).

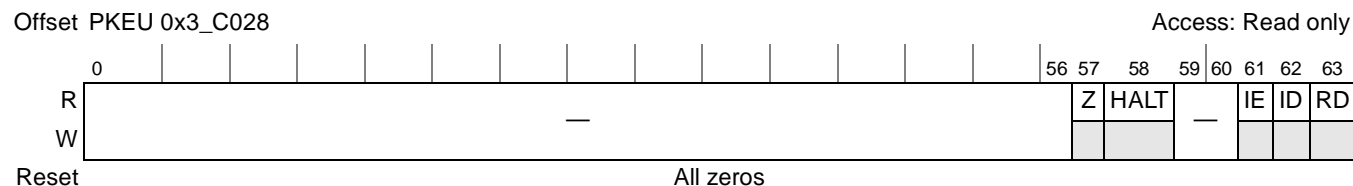


Figure 14-13. PKEU Status Register (PKEUSR)

Table 14-12 describes PKEUSR fields.

Table 14-12. PKEU Status Register Field Descriptions

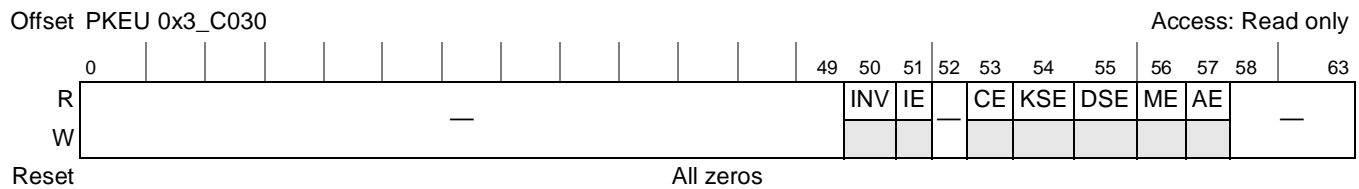
Bits	Name	Description
0–56	—	Reserved
57	Z	Zero. Reflects the state of the PKEU zero detect bit when last sampled. Only particular instructions within routines cause zero to be modified, so this bit should be used with great care.
58	HALT	Halt indicates whether the PKEU has halted due to an error. 0 PKEU not halted 1 PKEU halted Note: Because the error causing the PKEU to stop operating may be masked to the interrupt status register, the status register is used to provide a second source of information regarding errors preventing normal operation.

Table 14-12. PKEU Status Register Field Descriptions (continued)

Bits	Name	Description
59–60	—	Reserved
61	IE	Interrupt error. Reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (see Section 14.7.2.2, “Interrupt Status Register (ISR)”). 0 PKEU is not signaling error 1 PKEU is signaling error
62	ID	Interrupt done. Reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (see Section 14.7.2.2, “Interrupt Status Register (ISR)”). 0 PKEU is not signaling done 1 KEU is signaling done
63	RD	Reset done. 0 Reset in progress 1 Reset done. PKEU has completed its internal reset sequence.

14.5.1.7 PKEU Interrupt Status Register (PKEUISR)

The PKEU interrupt status register tracks the state of possible errors, if those errors are not masked, through the PKEU interrupt control register (PKEUICR). The definition of each bit in the PKEUISR is shown in [Figure 14-14](#).


Figure 14-14. PKEU Interrupt Status Register (PKEUISR)

[Table 14-13](#) describes PKEUISR fields.

Table 14-13. PKEUISR Field Descriptions

Bits	Name	Description
0–49	—	Reserved
50	INV	Inversion error. Indicates that the inversion routine has a zero operand. 0 No inversion error detected 1 Inversion error detected
51	IE	Internal error. An internal processing error was detected while the PKEU was operating. 0 No error detected 1 Internal error Note: This bit will be asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the Interrupt Control Register or by resetting the PKEU.
52	—	Reserved
53	CE	Context error. A PKEU key register, the key size register, the data size register, or mode register was modified while the PKEU was operating. 0 No error detected 1 Context error

Table 14-13. PKEUISR Field Descriptions (continued)

Bits	Name	Description
54	KSE	Key size error. Value outside the bounds of 1–256 bytes was written to the PKEU key size register 0 No error detected 1 Key size error detected
55	DSE	Data size error. Value outside the bounds 97– 2048 bits was written to the PKEU data size register 0 No error detected 1 Data size error detected
56	ME	Mode error. An illegal value was detected in the mode register. 0 No error detected 1 Mode error Note: Writing to reserved bits in a mode register is a likely source of error.
57	AE	Address error. Illegal read or write address was detected within the PKEU address space. 0 No error detected 1 Address error
58–63	—	Reserved

14.5.1.8 PKEU Interrupt Control Register (PKEUICR)

The PKEU interrupt control register controls the result of detected errors. For a given error (as defined in [Section 14.5.1.7, “PKEU Interrupt Status Register \(PKEUISR\)”](#)), 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, PKEUISR is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

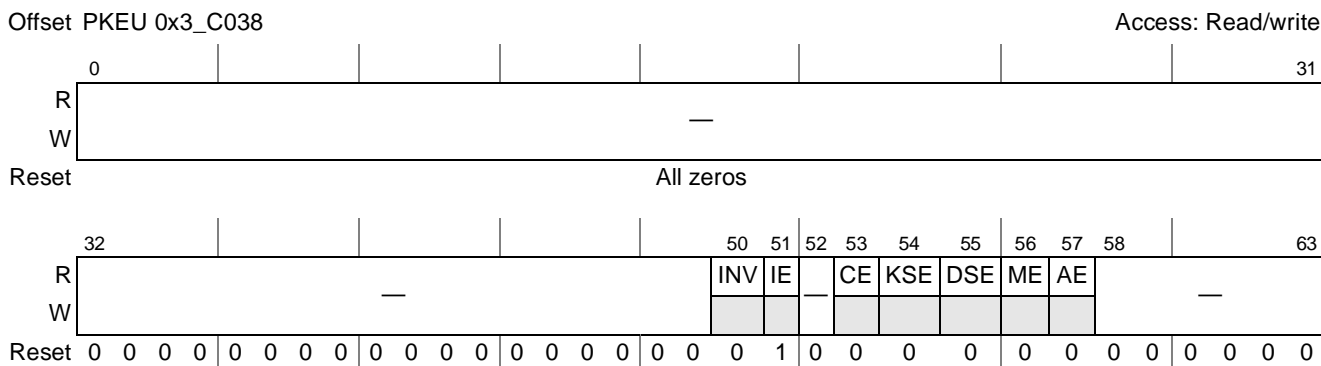


Figure 14-15. PKEU Interrupt Control Register (PKEUICR)

Table 14-14 describes PKEUICR fields.

Table 14-14. PKEUICR Field Descriptions

Bits	Name	Description
0–49	—	Reserved
50	INV	Inversion error 0 Inversion error enabled 1 Inversion error disabled
51	IE	Internal error 0 Internal error enabled 1 Internal error disabled
52	—	Reserved
53	CE	Context error 0 Context error enabled 1 Context error disabled
54	KSE	Key size error 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error 0 Mode error enabled 1 Mode error disabled
57	AE	Address error 0 Address error enabled 1 Address error disabled
58–63	—	Reserved

14.5.1.9 PKEU EU-Go Register (PKEUEUG)

The EU-Go 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 mode register, per the contents of the parameter memories listed below. Note that this register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Moreover, no read operation from this register is meaningful, but no error is generated, and a zero value is always returned. PKEUEUG is only used when the is operated as a slave. The descriptors and crypto-channel activate the PKEU (through an internally generated write to PKEUEUG) when the acts as an initiator.

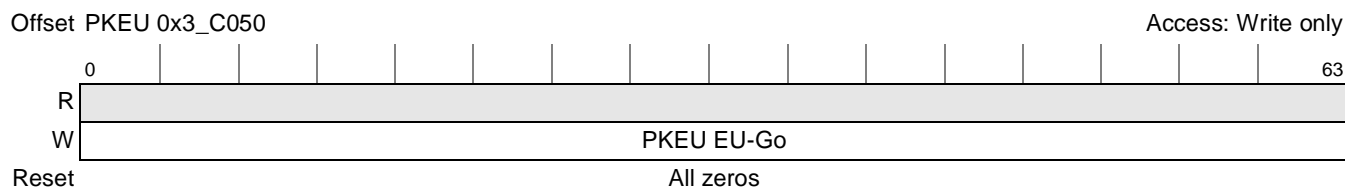


Figure 14-16. PKEU EU-Go Register (PKEUEUG)

14.5.1.10 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.

All these memories store data in the same format: least significant data byte in the least significantly addressed byte, both data significance and addressing significance increasing identically and simultaneously.

14.5.1.10.1 PKEU Parameter Memory A

This 2048-bit memory is typically used 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.

14.5.1.10.2 PKEU Parameter Memory B

This 2048-bit memory is typically used as an input parameter memory space, as well as the result memory space. For modular arithmetic routines, this memory serves 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.

14.5.1.10.3 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 PKEUISR.

14.5.1.10.4 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.

14.5.2 Data Encryption Standard Execution Unit (DEU)

This section contains details about the data encryption standard execution unit (DEU), including detailed register map, modes of operation, status and control registers, and FIFOs.

14.5.2.2 DEU Key Size Register (DEUKSR)

The DEUKSR indicates the number of bytes of key memory used in encrypting or decrypting. If DEUMR is set for single DES, any value other than 8 bytes automatically generates a key size error in the DEUISR. 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.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated Interrupt Control Register prior to performing debug operations.

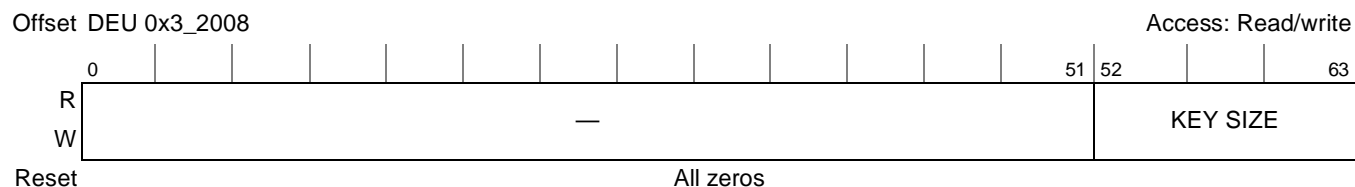


Figure 14-18. DEU Key Size Register (DEUKSR)

Table 14-16 shows the legal values for DEU key size.

Table 14-16. DEUKSR Field Descriptions

Bits	Name	Description
0–51	—	Reserved
52–63	Key Size	8 bytes = 0x08 (only legal value if mode is single DES.) 16 bytes= 0x10 (for 2 key 3DES, K1 = K3) 24 bytes= 0x18 (for 3 key 3DES)

14.5.2.3 DEU Data Size Register (DEUDSR)

The DEUDSR, shown in Figure 14-19, is used to verify that the data to be processed by the DEU is divisible by the DES algorithm block size of 64 bits. The DEU does not automatically pad messages out to 64-bit blocks; therefore, any message processed by the DEU must be divisible by 64 bits or a data size error will occur.

In normal operation, the full message length (data size) to be encrypted or decrypted by the DEU is copied from the descriptor to DEUDSR; however, only bits 58–63 are checked to determine if there is a data size error. If bits 58–63 are all zeros, the message is evenly divisible into 64-bit blocks. In target mode, the user must write the data size to DEUDSR. If the data size written is not divisible by 64-bits (bits 58–63 nonzero), a data size error will occur.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated Interrupt Control Register prior to performing debug operations.

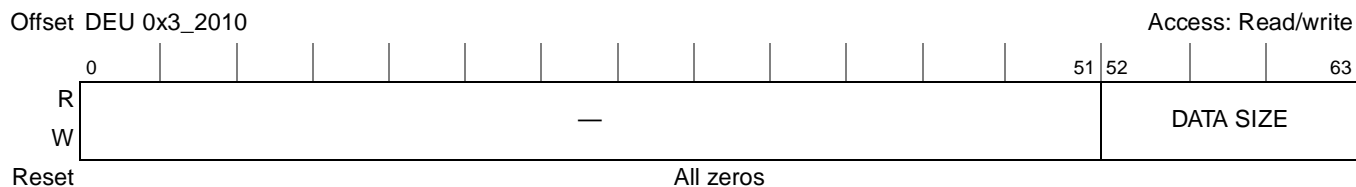


Figure 14-19. DEU Data Size Register (DEUSDR)

14.5.2.4 DEU Reset Control Register (DEURCR)

The DEURCR, shown in Figure 14-20, allows three levels reset of just DEU, as defined by the three self-clearing bits:

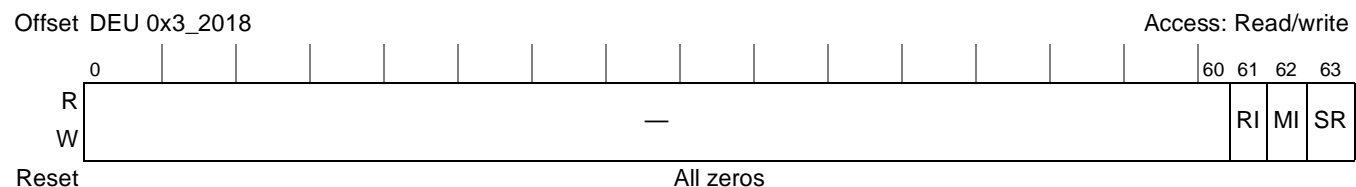


Figure 14-20. DEU Reset Control Register (DEURCR)

Table 14-17 describes DEURCR fields.

Table 14-17. DEURCR Field Descriptions

Bits	Names	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes DEU interrupts signaling DONE and ERROR to be reset. It further resets the state of DEUISR. 0 Don't reset. 1 Reset interrupt logic.
62	MI	Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. this module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in DEUSR. 0 Don't reset. 1 Reset most of DEU.
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for DEU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the DEU will enter a routine to perform proper initialization of the parameter memories. The RESET_DONE bit in DEUSR will indicate when this initialization routine is complete 0 Don't reset. 1 Full DEU reset

14.5.2.5 DEU Status Register (DEUSR)

The DEU status register, shown in Figure 14-21, contains six fields which reflect the state of DEU internal signals. DEUSR is read only; writing to DEUSR causes an address error being reflected in DEUISR.

Table 14-19 describes DEUISR fields.

Table 14-19. DEUISR Field Descriptions

Bits	Name	Description
0–49	—	Reserved
50	KPE	Key parity error. Defined parity bits in the keys written to the key registers did not reflect odd parity correctly. (Note that 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 error detected 1 Key parity error
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 No error detected 1 Internal error Note: IE is asserted anytime an enabled error condition occurs and can only be cleared by setting the corresponding bit in the DEUICR or by resetting the DEU.
52	ERE	Early read error. The DEU IV register was read while the DEU was performing encryption. 0 No error detected 1 Early read error
53	CE	Context error. A DEU Key Register, the Key Size Register, Data Size Register, Mode Register, or IV Register was modified while DEU was performing encryption. 0 No error detected 1 Context error
54	KSE	Key size error. 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 error detected 1 Key size error
55	DSE	Data size error: A value was written to the DEU Data Size Register that is not a multiple of 64 bits. 0 No error detected 1 Data size error
56	ME	Mode error. An illegal value was detected in the mode register. Note: writing to reserved bits in mode register is likely source of error. 0 No error detected 1 Mode error
57	AE	Address error. An illegal read or write address was detected within the DEU address space. 0 No error detected 1 Address error
58	OFE	Output FIFO error. The DEU output FIFO was detected non-empty upon write of DEU data size register. 0 No error detected 1 Output FIFO non-empty error
59	IFE	Input FIFO error. The DEU input FIFO was detected non-empty upon generation of DONE interrupt. 0 No error detected 1 Input FIFO non-empty error
60	IFU	Input FIFO underflow. The DEU input FIFO has been read while empty. 0 No error detected 1 Input FIFO has had underflow error

Table 14-19. DEUISR Field Descriptions (continued)

Bits	Name	Description
61	IFO	Input FIFO overflow. The DEU input FIFO has been pushed while full. 0 No error detected 1 Input FIFO has overflowed Note: When operating as a master, the implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the cannot accept FIFO inputs larger than 512 bytes without overflowing.
62	OFU	Output FIFO underflow. The DEU output FIFO has been read while empty. 0 No error detected 1 Output FIFO has underflow error
63	OFO	Output FIFO overflow. The DEU output FIFO has been pushed while full. 0 No error detected 1 Output FIFO has overflowed

14.5.2.7 DEU Interrupt Control Register (DEUICR)

The DEU interrupt control register controls the result of detected errors. For a given error (as defined in [Section 14.5.2.6, “DEU Interrupt Status Register \(DEUISR\)”](#)), if the corresponding bit in this register is set, then the error is ignored, no error interrupt occurs and DEUISR is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, DEUISR is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

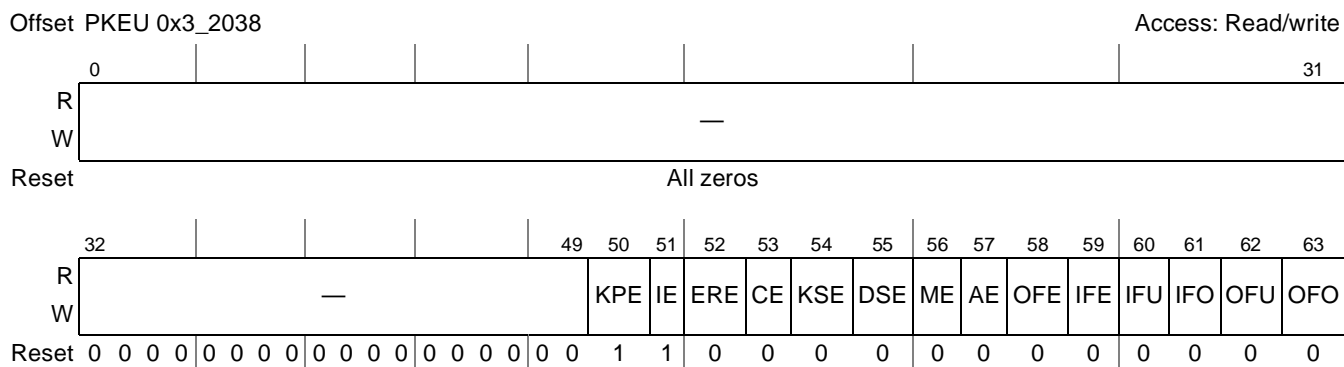


Figure 14-23. DEU Interrupt Control Register (DEUICR)

Table 14-20. DEUICR Field Descriptions

Bits	Name	Description
0–49	—	Reserved
50	KPE	Key parity error. The defined parity bits in the keys written to the key registers did not reflect odd parity correctly. (Note that key register 2 and key register 3 are only checked for parity if the appropriate DEU mode register bit indicates triple DES. 0 Key parity enabled 1 Key parity error disabled
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 Internal error enabled 1 Internal error disabled

Table 14-20. DEUICR Field Descriptions (continued)

Bits	Name	Description
52	ERE	Early read error. The DEU IV Register was read while the DEU was performing encryption. 0 Early read error enabled 1 Early read error disabled
53	CE	Context error. A DEU key register, the key size register, the data size register, the mode register, or IV register was modified while DEU was performing encryption. 0 Context error enabled 1 Context error disabled
54	KSE	Key size error. 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 Key size error enabled 1 Key size error disabled
55	DSE	Data size error (DSE): A value was written to the DEU data size register that is not a multiple of 8 bytes. 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
57	AE	Address error. An illegal read or write address was detected within the DEU address space. 0 Address error enabled 1 Address error disabled
58	OFE	Output FIFO error. The DEU output FIFO was detected non-empty upon write of DEU data size register 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
59	IFE	Input FIFO error. The DEU input FIFO was detected non-empty upon generation of DONE interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	IFU	Input FIFO underflow. The DEU input FIFO has been read while empty. 0 Input FIFO Underflow error enabled 1 Input FIFO Underflow error disabled
61	IFO	Input FIFO overflow. The DEU input FIFO has been pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled Note: When operating as a master, the implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the cannot accept FIFO inputs larger than 512 bytes without overflowing.
62	OFU	Output FIFO underflow. The DEU output FIFO has been read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	OFO	Output FIFO overflow. The DEU output FIFO has been pushed while full. 0 Output FIFO Overflow error enabled 1 Output FIFO Overflow error disabled

14.5.2.8 DEU EU-Go Register (DEUEUG)

The EU-Go register in the DEU is used to indicate a DES operation may be completed. After the final message block is written to the input FIFO, DEUEUG must be written. The value in the data size register

will be used to determine how many bits of the final message block (always 64) will be processed. Note that this register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Moreover, no read operation from this register is meaningful, but no error is generated, and a zero value is always returned. Writing to DEUEUG is merely a trigger causing the DEU to process the final block of a message, allowing it to signal DONE.

DEUEUG is only used when the SEC is operated as a slave. The descriptors and crypto-channel activate the DEU (through an internally generated write to DEUEUG) when the SEC acts as an initiator.

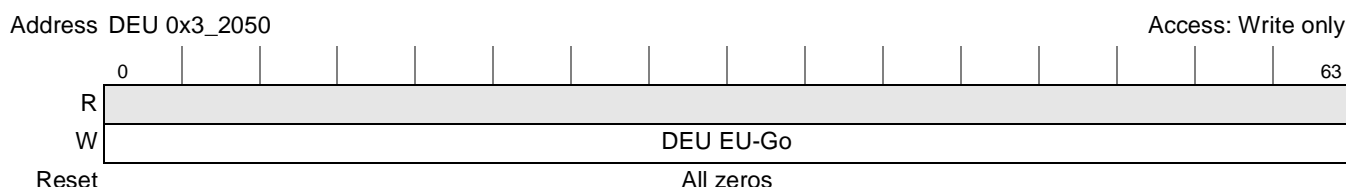


Figure 14-24. DEU EU-Go Register (DEUEUG)

14.5.2.9 DEU IV Register (DEUIV)

For CBC mode, the initialization vector is written to and read from DEUIV. The value of this register changes as a result of the encryption process and reflects the context of DEU. Reading DEUIV while the module is processing data generates an error interrupt.

14.5.2.10 DEU Key Registers (DEUK1–DEUK3)

The DEU uses three write-only key registers to perform encryption and decryption. In Single DES mode, only DEUK1 may be written. The value written to DEUK1 is simultaneously written to DEUK3, auto-enabling the DEU for 112-bit triple DES if DEUKSR indicates 2-key 3DES is to be performed (key size = 16 bytes). To operate in 168-bit triple DES, DEUK1 must be written first, followed by DEUK2, then DEUK3.

Reading any of these memory locations generates an address error interrupt.

14.5.2.11 DEU FIFOs

DEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. These FIFOs are multiply addressable, but those multiple addresses point only to the appropriate end of the appropriate FIFO. A write to anywhere in the DEU FIFO address space causes the 64-bit-word to be pushed onto the DEU input FIFO, and a read from anywhere in the DEU FIFO address space causes a 64-bit-word to be popped off the DEU output FIFO. Overflows and underflows caused by reading or writing the DEU FIFOs are reflected in the DEU interrupt status register.

14.5.3 ARC Four Execution Unit (AFEU)

This section contains details about the ARC four execution unit (AFEU), including detailed register map, modes of operation, status and control registers, S-box memory, and FIFOs.

The registers used in the AFEU are documented primarily for debug and slave mode operations. If the SEC requires the use of the AFEU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and the on-chip controller will abstract register level access from the user.

14.5.3.1 AFEU Mode Register (AFEUMR)

Shown in [Figure 14-25](#), the AFEU mode register contains three bits which are used to program the AFEU. It also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the SEC as an initiator. Burst size is not relevant to slave mode operations, where an external host pushes and pulls data from the execution units.

AFEUMR is cleared when the AFEU is reset or re-initialized. Setting a reserved mode bit generates a data error. If AFEUMR is modified during processing, a context error is generated.

14.5.3.2 Host-Provided Context through Prevent Permute

In the default mode of operation, the host 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 AFEUMR[PP] is set (prevent permute mode is enabled), the AFEU will not require a key. Rather, the host 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 AFEUMR[CS] is set.

14.5.3.2.1 Dump Context

This mode may be independently specified (using AFEUMR[DC]) in addition to host-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 through the output FIFO.

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 (through 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.

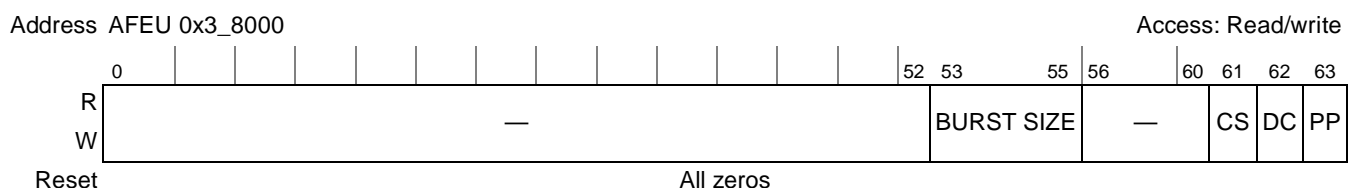


Figure 14-25. AFEU Mode Register (AFEUMR)

NOTE

The device driver will create properly formatted descriptors for situations requiring a key permute prior to ciphering. When operating the SEC as a slave (typically while in debug mode), the user must set the AFEU mode register (AFEUMR) to perform ‘permute with key’, then write the key data to the AFEU key registers, then write the key size to the key size register (AFEUKSR[Key Size]). The AFEU will start permuting the memory with the contents of the AFEU key registers immediately after AFEUKSR[Key Size] is written.

14.5.3.4 AFEU Context/Data Size Register (AFEUDSR)

The AFEU context/data size register, shown in [Figure 14-27](#), stores the number of bits in the final message block. AFEUDSR is cleared when the AFEU is reset or re-initialized. The last message block can be between 8 to 64 bits. If a data size that is not a multiple of 8 bits is written, a data size error will be generated. A data size of 0 is illegal and results in the associated crypto-channel locking, requiring a crypto-channel and AFEU reset.

AFEUDSR is also used to specify the context size. 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

In slave mode, when reloading an existing context, 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 that all context has been loaded. The user then writes the message data size to AFEUDSR. After this write, the user may begin writing message data to the FIFO.

Writing to AFEUDSR 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 will be generated.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated Interrupt Control Register prior to performing debug operations.

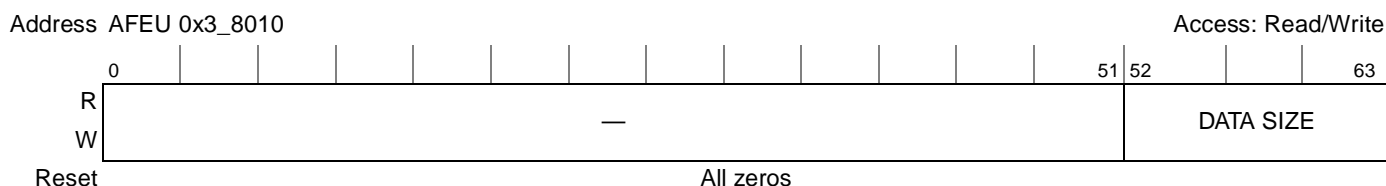


Figure 14-27. AFEU Data Size Register

14.5.3.5 AFEU Reset Control Register (AFEURCR)

The AFEU reset control register, shown in Figure 14-28, allows 3 levels reset that affect the AFEU only, as defined by 3 self-clearing bits. It should be noted that the AFEU executes an internal reset sequence for hardware reset, software reset, or module initialization, which performs proper initialization of the S-Box. To determine when this is complete, observe the RESET_DONE bit in AFEUSR.

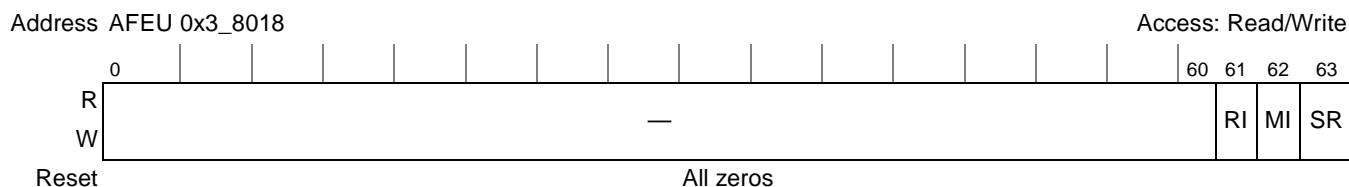


Figure 14-28. AFEU Reset Control Register (AFEURCR)

Table 14-22 describes AFEURCR fields

Table 14-22. AFEURCR Field Descriptions

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit causes AFEU interrupts signaling DONE and ERROR to be reset. It further resets the state of AFEUISR. 0 Do not reset 1 Reset interrupt logic
62	MI	Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. 0 Do not reset 1 Reset most of AFEU
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for AFEU. All registers and internal state are returned to their defined reset state. On negation of SW_RESET, the AFEU enters a routine to perform proper initialization of the S-box. 0 Do not reset 1 Full AFEU reset Note: Following a power on reset, the AFEU must be reset prior to first use. Failure to do so results in undefined behavior. Set the SR bit to perform AFEU reset.

14.5.3.6 AFEU Status Register (AFEUSR)

This status register, shown in Figure 14-29, contains 6 bits which reflect the state of the AFEU internal signals.

The AFEUSR is read-only. Writing to this location causes address error being reflected in AFEUISR.

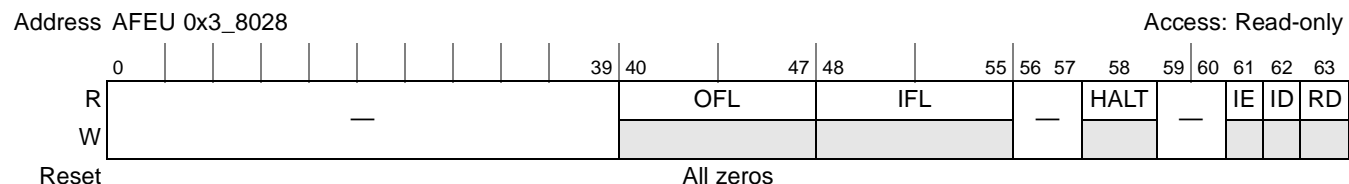


Figure 14-29. AFEU Status Register

Table 14-24 describes AFEUISR fields.

Table 14-24. AFEUISR Field Descriptions

Bits	Names	Description
0–50	—	Reserved
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 No error detected 1 Internal error
52	ERE	Early read error. The AFEU Context Memory or Control was read while the AFEU was performing encryption. 0 No error detected 1 Early read error
53	CE	Context error. AFEUMR, AFEUKR _n , AFEUKSR, AFEUDSR, or context memory was modified while AFEU processes data. 0 No error detected 1 Context error
54	KSE	Key size error. A value outside the bounds 1–16 bytes was written to AFEUKSR. 0 No error detected 1 Key size error
55	DSE	Data size error. An inconsistent value (not a multiple of 8 bits, or larger than 64 bits) was written to AFEUDSR. 0 No error detected 1 Data size error
56	ME	Mode error. An illegal value was detected in AFEUMR. Note: Writing to reserved bits in mode register is likely source of error. 0 No error detected 1 Mode error
57	AE	Address error. An illegal read or write address was detected within the AFEU address space. 0 No error detected 1 Address error
58	OFE	Output FIFO error. The AFEU output FIFO was detected non-empty upon write of AFEUDSR. 0 No Output FIFO error detected 1 Output FIFO error detected
59	IFE	Input FIFO error. The AFEU input FIFO was detected non-empty upon generation of DONE interrupt. 0 No input FIFO error detected 1 Input FIFO error detected
60	—	Reserved
61	IFO	Input FIFO overflow. The AFEU input FIFO has been pushed while full. 0 No error detected 1 Input FIFO has overflowed. Note: When operating as a master, the AFEU implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the AFEU cannot accept FIFO inputs larger than 512 bytes without overflowing.
62	OFU	Output FIFO underflow. The AFEU output FIFO has been read while empty. 0 No error detected 1 Output FIFO has underflow error.
63	—	Reserved

Table 14-25. AFEUICR Field Descriptions (continued)

Bits	Names	Description
58	OFE	Output FIFO error. The AFEU output FIFO was detected non-empty upon write of AFEUDSR. 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
59	IFE	Input FIFO error. The AFEU Input FIFO was detected non-empty upon generation of DONE interrupt. 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	—	Reserved
61	IFO	Input FIFO overflow. The AFEU Input FIFO has been pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62	OFU	Output FIFO underflow. The AFEU Output FIFO has been read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

14.5.3.9 AFEU End of Message Register (AFEUEMR)

AFEUEMR, shown in [Figure 14-32](#), is used to indicate an ARC4 operation may be completed. After the final message block is written to the input FIFO, AFEUEMR must be written. The value in AFEUDSR will be used to determine how many bits of the final message block (8–64, in multiples of 8) will be processed. Writing to this register causes the AFEU to process the final block of a message, allowing it to signal DONE. If AFEUEMR[DC] is set (dump context mode is enabled), the context is written to the output FIFO following the last message word. A read of AFEUEMR always returns a zero value.

AFEUEMR is only used when the AFEU is operated as a slave. The descriptors and crypto-channel activate the AFEU (by means of an internally generated write to AFEUEMR) when the SEC acts as an initiator.

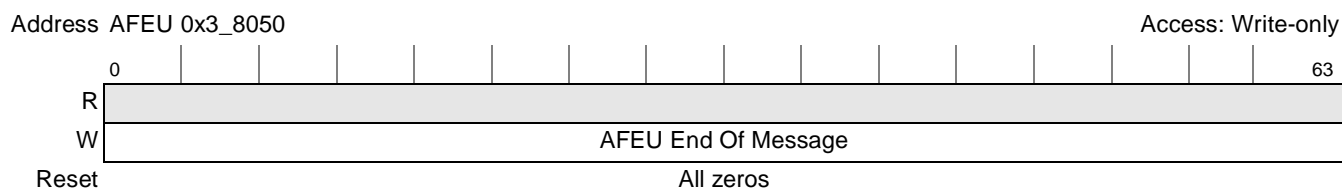


Figure 14-32. AFEU End of Message Register (AFEUEMR)

14.5.3.10 AFEU Context

This section provides additional information about the AFEU context memory and its related pointer register.

14.5.3.10.1 AFEU Context Memory

The S-Box memory consists of 32 64-bit words, each readable and writable. The S-Box contents should not be written with data unless it was previously read from the S-Box. Context data may only be written

if AFEUMR[PP] is set (prevent permutation mode is enabled, see [Figure 14-25](#)), and the context data must be written prior to the message data. If the context registers are written during message processing or AFEUMR[PP] is not set, a context error will be generated. Reading this memory while the module is not done will generate an error interrupt.

14.5.3.10.2 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 ARC4 algorithm. If this register is written during message processing, a context error will be generated.

When performing ARC4 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 the next packet. The S-Box contents (256 bytes) plus the 3 bytes of the context memory pointers are unloaded and reloaded through 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.

14.5.3.11 AFEU Key Registers (AFEUK0, AFEUK1)

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 AFEUK0, and uses as many bytes from two key registers as necessary to complete the permutation. Reading either of these memory locations generates an address error interrupt.

14.5.3.11.1 AFEU FIFOs

AFEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. These FIFOs are multiply addressable, but those multiple addresses point only to the appropriate end of the appropriate FIFO. A write to anywhere in the AFEU FIFO address space causes the 64-bit-word to be pushed onto the AFEU input FIFO, and a read from anywhere in the AFEU FIFO address space causes a 64-bit-word to be popped off the AFEU output FIFO. Overflows and underflows caused by reading or writing the AFEU FIFOs are reflected in the AFEU interrupt status register.

14.5.4 Message Digest Execution Unit (MDEU)

The registers used in the MDEU are documented primarily for debug and slave mode operations. If the SEC requires the use of the MDEU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and the on-chip controller will abstract register level access from the user.

14.5.4.1 MDEU Mode Register (MDEUMR)

The MDEU mode register (MDEUMR) is used to program the function of the MDEU. Bits 56–63 of the MDEUMR 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.

Table 14-27. MDEUMR in ‘New’ Configuration (continued)

Bits	Name	Description
56	CONT	Continue. Most operations will require this bit to be cleared. Set only when the data to be hashed is spread across multiple descriptors. 0 Do autopadding 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 will be continued in a subsequent descriptor. Do not autopad and do not complete the message digest.
57	CICV	Compare integrity check values 0 Normal operation; no ICV comparison. 1 After the message digest (ICV) is computed, compare it to the data in the MDEU's input FIFO. If the ICVs do not match, send an error interrupt to the channel. The number of bytes to be compared is given by the ICV size register.
58	SMAC	Specifies whether to perform an SSL-MAC operation 0 Normal operation 1 Perform an SSL3.0 MAC operation. This requires a key and key length. If this is set then the HMAC bit should be 0.
59	INIT	Initialization bit. Most operations will require this bit to be set. Cleared only for operations that load context from a known intermediate hash value. 0 Do not initialize digest registers. In this case 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 is set on all but the first descriptor. 1 Do an algorithm-specific initialization of the digest registers.
60	HMAC	Specifies whether to perform an HMAC operation 0 Normal operation 1 Perform an HMAC operation. This requires a key and key length. If this is set then the SMAC bit should be 0.
61	EALG	The EALG (Extended Algorithm bit) and ALG (Algorithm) bits together specify the message digest algorithm, as follows: 000 SHA-160 algorithm (full name for SHA-1) 001 SHA-256 algorithm 010 MD5 algorithm 011 SHA-224 algorithm Others: Reserved
62–63	ALG	

14.5.4.2 Recommended Settings for MDEUMR

The most common task likely to be executed through 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), the following mode register bit settings should be used:

Table 14-28. 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, the following mode register bit settings should be used:

Table 14-29. 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 14.2.1, “Data Packet Descriptors.”](#)

14.5.4.3 MDEU Key Size Register (MDEUKSR)

MDEUKSR, shown in [Figure 14-35](#), indicates the number of bytes of key memory that should be used in HMAC generation. The MDEU supports at most 64 bytes of key. The MDEU will generate a key size error if the value written to MDEUKSR exceeds 64 bytes.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated interrupt control register before performing debug operations.

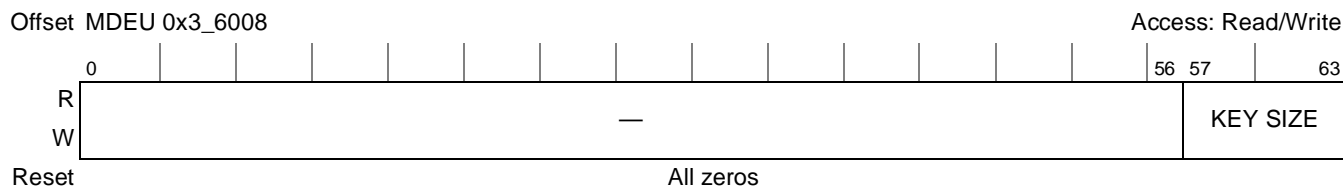


Figure 14-35. MDEU Key Size Register (MDEUKSR)

14.5.4.4 MDEU Data Size Register (MDEUDSR)

The MDEU data size register, shown in [Figure 14-36](#), stores the size of the last block of data (in bits) to be processed. Since the MDEU does not support bit offsets, any value other than 0 in bits 61–63 will cause a data size error. Bits 58–60 are used to identify the ending byte location in the last 8-byte dword. This is used to add the data padding when auto padding is selected. MDEUDSR is cleared when the MDEU is reset, re-initialized, and at the end of processing the complete message.

NOTE

Writing to MDEUDSR allows the MDEU to enter auto-start mode. Therefore, the required context data should be written prior to writing the data size.

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated interrupt control register before performing debug operations.

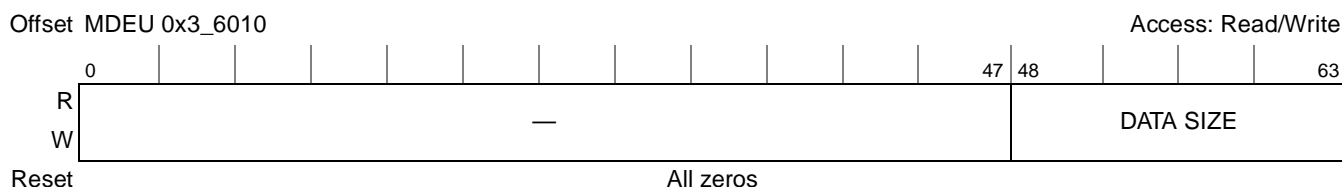


Figure 14-36. MDEU Data Size Register (MDEUDSR)

14.5.4.5 MDEU Reset Control Register (MDEURCR)

MDEURCR, shown in [Figure 14-37](#), allows three levels reset of just the MDEU, as defined by the three self-clearing bits.

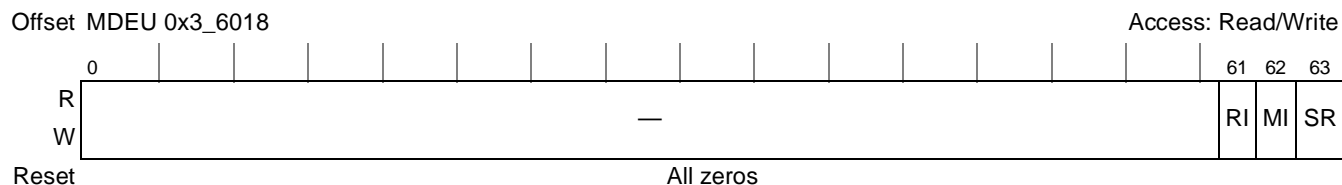


Figure 14-37. MDEU Reset Control Register (MDEURCR)

[Table 14-30](#) describes MDEU reset control register fields.

Table 14-30. MDEU Reset Control Register Field Descriptions

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes MDEU interrupts signaling DONE and ERROR to be reset. It further resets the state of the MDEUISR. 0 No reset 1 Reset interrupt logic

Table 14-30. MDEU Reset Control Register Field Descriptions (continued)

Bits	Name	Description
62	MI	Module initialization is nearly the same as software reset, except that MDEUICR is unchanged. 0 No reset 1 Reset most of MDEU
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for the MDEU. All registers and internal state are returned to their defined reset state. 0 No reset 1 Full MDEU reset

14.5.4.6 MDEU Status Register (MDEUSR)

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 and key padding, 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 bits between 0–57. The four signals shown are those most likely to be of interest to the user.

MDEUSR, shown in [Figure 14-38](#), is read only.

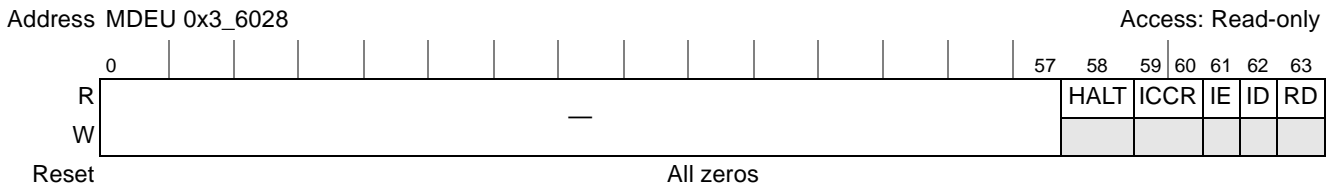


Figure 14-38. MDEU Status Register

[Table 14-23](#) describes MDEUSR fields.

Table 14-31. MDEU Status Register Field Descriptions

Bits	Name	Description
0–57	—	Reserved
58	HALT	Halt. Indicates that the MDEU halted due to an error. 0 MDEU not halted 1 MDEU halted Note: Because the error causing the MDEU to stop operating may be masked to MDEUISR, MDEUSR is used to provide a second source of information regarding errors preventing normal operation.
59–60	ICCR	Integrity check comparison result 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved Note: A passed or failed result is generated only if ICV checking is enabled.

Table 14-31. MDEU Status Register Field Descriptions (continued)

Bits	Name	Description
61	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller ISR (Section 14.7.2.2, "Interrupt Status Register (ISR)"). 0 MDEU is not signaling error. 1 MDEU is signaling error.
62	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the controller ISR (Section 14.7.2.2, "Interrupt Status Register (ISR)"). 0 MDEU is not signaling done 1 MDEU is signaling done
63	RD	Reset done. This status bit, when high, indicates that MDEU has completed its internal reset sequence. 0 Reset in progress 1 Reset done

14.5.4.7 MDEU Interrupt Status Register (MDEUISR)

The interrupt status register tracks the state of possible errors, if those errors are not masked through the MDEUISR. The definition of each field in MDEUISR is shown in Figure 14-39.

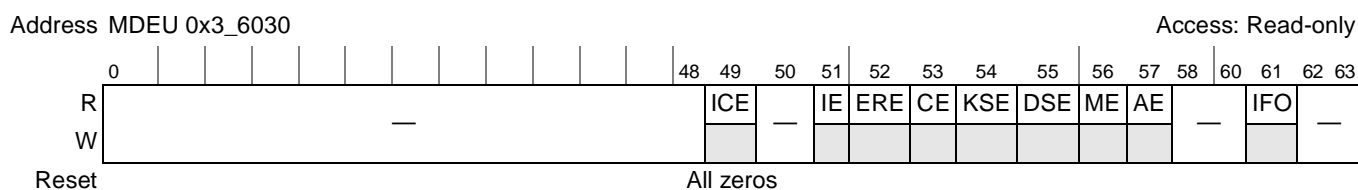


Figure 14-39. MDEU Interrupt Status Register (MDEUISR)

Table 14-32 describes MDEUISR fields.

Table 14-32. MDEUISR Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity check error. 0 No error detected 1 Integrity check error detected. An ICV check was performed and the supplied ICV did not match the one computed by the MDEU.
50	—	Reserved
51	IE	Internal error. Indicates the MDEU has been locked up and requires a reset before use. 0 No internal error detected 1 Internal error detected Note: This bit is set any time an enabled error condition occurs and can only be cleared by resetting the MDEU.
52	ERE	Early read error. The MDEU context was read before the MDEU completed the hashing operation. 0 No error detected 1 Early read error

Table 14-32 describes MDEUICR fields.

Table 14-33. MDEUICR Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity check error. The supplied ICV did not match the one computed by the MDEU. 0 Integrity check error enabled 1 Integrity check error disabled
50	—	Reserved
51	IE	Internal error. An internal processing error is detected while performing hashing. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early read error. The MDEU register is read while the MDEU is performing hashing. 0 Early read error enabled 1 Early read error disabled
53	CE	Context error. The MDEU key register, MDEUKSR, MDEUDSR, or MDEUMR is modified while the MDEU is performing hashing. 0 Context error enabled 1 Context error disabled
54	KSE	Key size error. A value outside the bounds of 64 bytes is written to the MDEU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error. An inconsistent value is written to MDEUDSR: 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error. An illegal value is detected in MDEUMR. 0 Mode error enabled 1 Mode error disabled
57	AE	Address error. An illegal read or write address is detected within the MDEU address space. 0 Address error enabled 1 Address error disabled
58–60	—	Reserved
61	IFO	Input FIFO overflow. The MDEU input FIFO has been pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62–63	—	Reserved

14.5.4.9 MDEU ICV Size Register (MDEUICVSR)

The MDEU ICV size register, shown in [Figure 14-41](#), stores the number of bytes of the ICV result to be compared if the MDEU performs ICV comparison. (See [Section 14.5.4.1](#), “MDEU Mode Register (MDEUMR).”)

This register is cleared when the MDEU is reset or re-initialized.

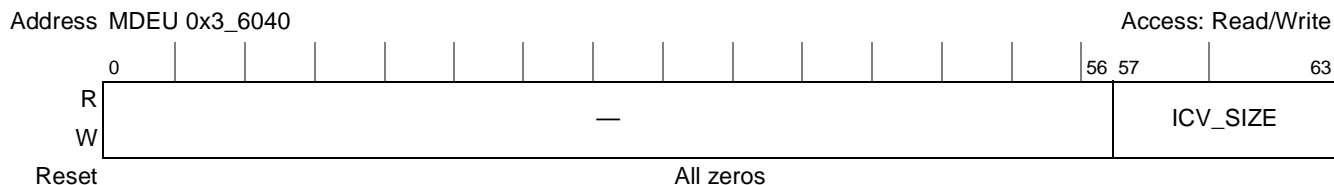


Figure 14-41. MDEU ICV Size Register

14.5.4.10 MDEU EU-Go Register (MDEUEUG)

The EU-Go register in the MDEU, see [Figure 14-42](#), is used to indicate an authentication operation may be completed. After the final message block is written to the input FIFO, the EU-Go register must be written. The value in the data size register will be used to determine how many bits of the final message block (always 512) will be processed. Note that this register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Moreover, no read operation from this register is meaningful, but no error is generated, and a zero value is always returned. Writing to this register is merely a trigger causing the MDEU to process the final block of a message, allowing it to signal DONE.

The DEU EU-Go register is only used when the SEC is operated as a slave. The descriptors and crypto-channel activate the MDEU (via an internally generated write to the EU-Go register) when the SEC acts as an initiator.

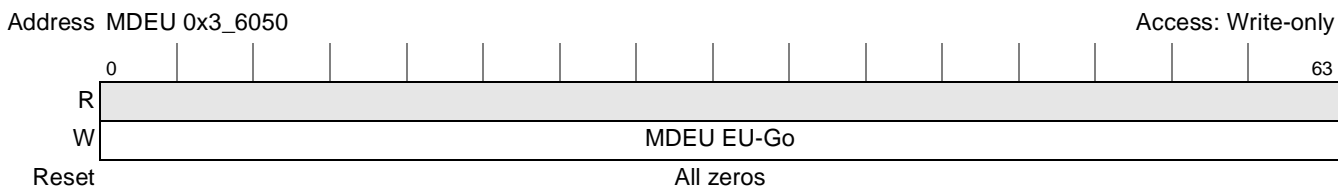


Figure 14-42. MDEU EU-Go Register

14.5.4.11 MDEU Context Registers

For MDEU, context consists of the hash plus the message length count. Write access to this register block allows continuation of a previous hash. Reading these registers provide the resulting message digest or HMAC, along with an aggregate bit count.

NOTE

SHA-1 and 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 will generate an error interrupt.

After a power-on reset, all the MDEU context register values are cleared. [Figure 14-43](#) 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 doesn't care about the MDEU Context Register initialization values, however they are documented for completeness in the event the user reads these registers during a debug operation. MDEU reset via the MDEU reset control register (Figure 14-37) or SEC global software reset (Figure 14-72) does not clear these registers.

	0	31	32	63	
Name	A		B		Context offset 0x3_6100
MD-5	0x01234567		0x89ABCDEF		
SHA-1	0x67452301		0xEFCDAB89		
SHA-256	0x6A09E667		0xBB67AE85		
Name	C		D		Context offset 0x3_6108
MD-5	0xFEDCBA98		0x76543210		
SHA-1	0x98BADCFE		0x10325476		
SHA-256	0x3C6EF372		0xA54FF53A		
Name	E		F		Context offset 0x3_6110
MD-5	0xF0E1D2C3		0x8C68059B		
SHA-1	0xC3D2E1F0		0x9B05688C		
SHA-256	0x510E527F		0x9B05688C		
Name	G		H		Context offset 0x3_6118
MD-5	0xABD9831F		0x19CDE05B		
SHA-1	0x1F83D9AB		0x5BE0CD19		
SHA-256	0x1F83D9AB		0x5BE0CD19		
Name	Message Length Count				Context offset 0x3_6120
Reset	0				

Figure 14-43. MDEU Context Register

14.5.4.12 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.

14.5.4.13 MDEU FIFO

MDEU uses an input FIFO to hold data to be hashed. The input FIFO is multiply addressable, but those multiple addresses point only to the write (push) end of the FIFO. A write to anywhere in the MDEU FIFO address space causes the 64-bit-words to be pushed onto the MDEU input FIFO, and a read from anywhere in the MDEU FIFO address space returns all zeros.

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.

14.5.5 Random Number Generator (RNG)

This section contains details about the random number generator (RNG), including detailed register map, modes of operation, status and control registers, and FIFOs.

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 (LFSR) 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.

The registers used in the MDEU are documented primarily for debug and slave mode operations. If the SEC requires the use of the MDEU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and the on-chip controller will abstract register level access from the user.

14.5.5.1 RNG Mode Register (RNGMR)

RNGMR, shown in [Figure 14-44](#), is used to control the RNG. One operational mode, randomizing, is defined. Writing any other value than 0 to 56:63 results in a data error interrupt reflected in the RNG interrupt status register. The mode register also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the as an initiator. Burst size is not relevant to slave mode operations, where an external host pushes and pulls data from the execution units.

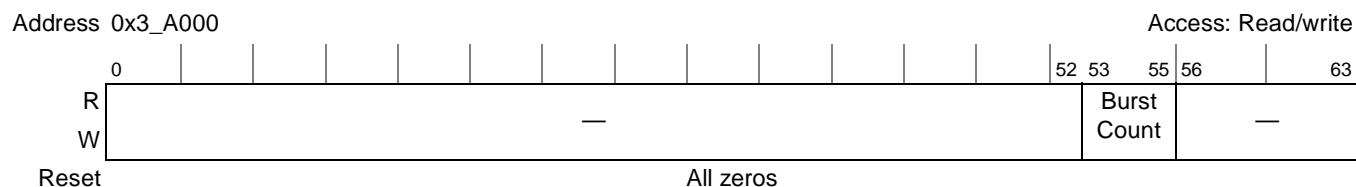


Figure 14-44. RNG Mode Register

Table 14-34. RNG Mode Register Definitions

Bits	Name	Description
0–52	—	Reserved, must be set to zero.
53–55	Burst Count	Burst Count. Implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/context. The RNG signals to the crypto-channel that a ‘Burst Size’ amount of data is available to be pulled from the FIFO. Note: The inclusion of this field in the RNG mode register is to avoid confusing a user who may read this register in debug mode. Burst Size should not be written directly to the RNG.
56–63	—	Reserved

14.5.5.2 RNG Data Size Register (RNGDSR)

RNGDSR, shown in [Figure 14-45](#), is used to tell the RNG to begin generating random data. The actual contents 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 will begin pushing data onto the FIFO. Data will be pushed onto the FIFO every 256 cycles until the FIFO is full. The RNG then attempts to keep the FIFO full.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated Interrupt Control Register prior to performing debug operations.

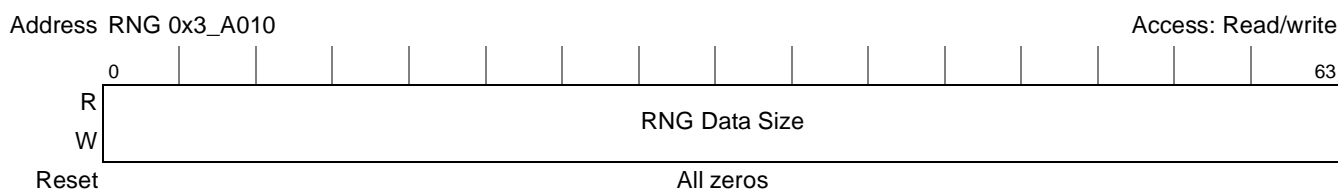


Figure 14-45. RNG Data Size Register

Table 14-38. RNG Interrupt Control Register Field Descriptions (continued)

Bits	Name	Description
57	AE	Address Error. An illegal read or write address was detected within the MDEU address space. 0 Address error enabled 1 Address error disabled
58–61	—	Reserved
62	OFU	Output FIFO Underflow. RNG Output FIFO has been read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

14.5.5.7 RNG EU-Go Register (RNGEUG)

RNGEUG, shown in [Figure 14-50](#), is a writable location but serves no function in the RNG. It is documented for the sake of consistency with the other EUs.

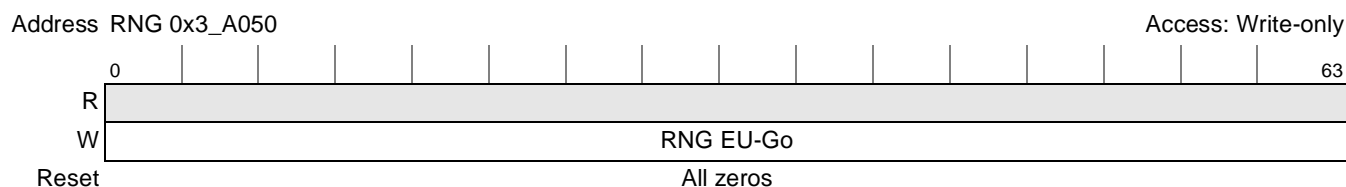


Figure 14-50. RNG EU-Go Register

14.5.5.8 RNG 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. The FIFO is multiply addressed, but those multiple addresses point only to the appropriate end of the output FIFO. A read from anywhere in the RNG FIFO address space causes a 64-bit-word to be popped off of the RNG output FIFO. 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.

14.5.6 Advanced Encryption Standard Execution Units (AESU)

This section contains details about the Advanced Encryption Standard Execution Units (AESU), including detailed register map, modes of operation, status and control registers, and FIFOs. The registers used in the AESU are documented primarily for debug and slave mode operations. If the SEC requires the use of the AESU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and the on-chip controller will abstract register level access from the user.

14.5.6.1 AESU Mode Register (AESUMR)

The AESU mode register, shown in [Figure 14-51](#), contains 7 bits which are used to program the AESU. The mode register is cleared when the AESU is reset or re-initialized. Setting a reserved mode bit will generate a data error. If the mode register is modified during processing, a context error will be generated.

Table 14-40. AES Cipher Modes

Mode	ECM (56:57)	CM (61:62)
ECB	00	00
CBC	00	01
Res	xx	10
CTR	00	11
SRT ¹	01	11
CCM (without ICV comparison)	10	00
CCM (with ICV comparison)	11	00
XOR	11	11
All Others	Reserved	

¹ 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 14.5.6.9.3, "Context for SRT Mode,"](#) for more information on how SRT mode reduces context loading overhead.

NOTE

Restore decrypt key (RDK)—In most networking applications, the decryption of an AES protected packet will be 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.)

The use of RDK is completely optional, as the Input time of the preserved decrypt key may exceed the ~12 cycles required to restore the decrypt key for processing the first block.

To use RDK, the following procedure is recommended:

The descriptor type used in decryption of the first portion of the message is "0100_0- AESU Key Expand Output". The AESU mode must be "Decrypt". See Table 14-6. for more information. The descriptor will cause the SEC to write the contents of the Context registers and the key registers (containing the expanded decrypt key) to memory.

To process the remainder of the message, use a 'common' descriptor type (0001_0), and set the restore-decrypt-key mode bit. 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).

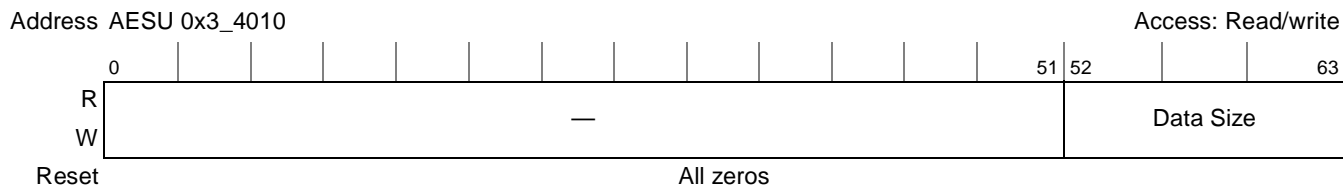


Figure 14-53. AESU Data Size Register

14.5.6.4 AESU Reset Control Register (AESURCR)

AESURCR allows three levels reset of just AESU, as defined by the three self-clearing bits:

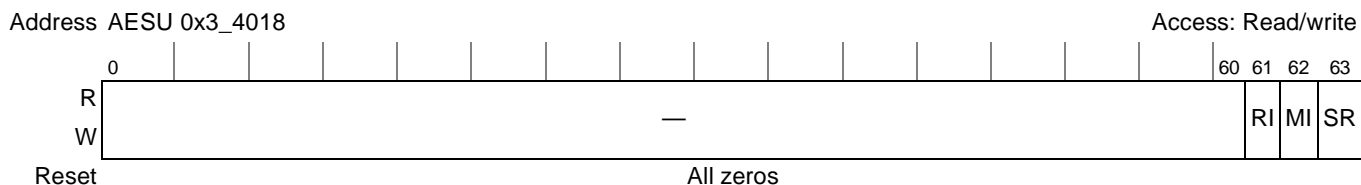


Figure 14-54. AESU Reset Control Register

Table 14-17 describes AESU reset control register fields.

Table 14-41. AESU Reset Control Register Field Descriptions

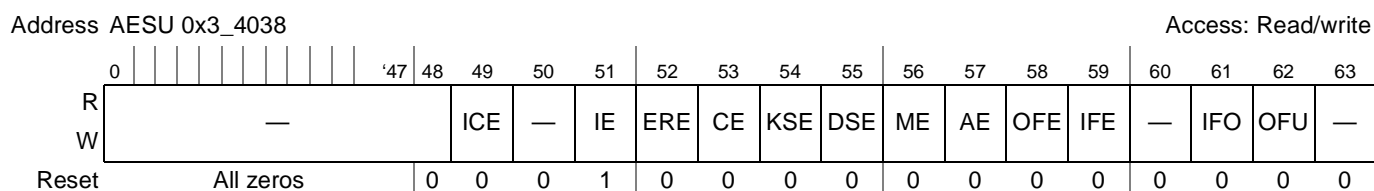
Bits	Names	Description
0–60	—	Reserved
61	RI	Reset Interrupt. Writing this bit active high causes AESU interrupts signalling DONE and ERROR to be reset. It further resets the state of the AESU interrupt status register. 0 Don't reset 1 Reset interrupt logic
62	MI	Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. This module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in the AESU status register 0 Don't reset 1 Reset most of AESU
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for AESU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the AESU will enter a routine to perform proper initialization of the parameter memories. The RESET_DONE bit in the AESU status register will indicate when this initialization routine is complete 0 Don't reset 1 Full AESU reset

Table 14-43. AESU Interrupt Status Register Field Descriptions (continued)

Bits	Name	Description
58	OFE	Output FIFO Error. The AESU output FIFO was detected non-empty upon write of AESU data size register. 0 No error detected 1 Output FIFO non-empty error
59	IFE	Input FIFO Error. The AESU input FIFO was detected non-empty upon generation of DONE interrupt. 0 No error detected 1 Input FIFO non-empty error
60	—	Reserved
61	IFO	Input FIFO Overflow. The AESU Input FIFO has been pushed while full. 0 No error detected 1 Input FIFO has overflowed Note: When operating as a master, the implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the cannot accept FIFO inputs larger than 512 Bytes without overflowing.
62	OFU	Output FIFO Underflow. The AESU Output FIFO has been read while empty. 0 No error detected 1 Output FIFO has underflow error
63	—	Reserved

14.5.6.7 AESU Interrupt Control Register (AESUICR)

The AESU interrupt control register, shown in [Figure 14-57](#), controls the result of detected errors. For a given error (as defined in [Section 14.5.6.6, “AESU Interrupt Status Register \(AESUISR\)”](#)), 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.


Figure 14-57. AESU Interrupt Control Register

[Table 14-44](#) describes the AESU interrupt control register fields.

Table 14-44. AESU Interrupt Control Register Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity check error. The supplied ICV did not match the one computed by the AESU. 0 Integrity check error enabled 1 Integrity check error disabled
50	—	Reserved

Table 14-44. AESU Interrupt Control Register Field Descriptions (continued)

Bits	Name	Description
51	IE	Internal Error. An internal processing error was detected while the AESU was processing. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early Read Error. The AESU IV Register was read while the AESU was processing. 0 Early read error enabled 1 Early read error disabled
53	CE	Context Error. An AESU Key Register, the Key Size Register, Data Size Register, Mode Register, or IV Register was modified while the AESU was processing. 0 Context error enabled 1 Context error disabled
54	KSE	Key Size Error. An inappropriate value (non 16, 24 or 32 bytes) was written to the AESU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data Size Error. Indicates that the number of bits to process is out of range. 0 Data size error enabled 1 Data size error disabled
56	ME	Mode Error. Indicates that invalid data was written to a register or a reserved mode bit was set. 0 Mode error enabled 1 Mode error disabled
57	AE	Address Error. An illegal read or write address was detected within the AESU address space. 1 Address error disabled 0 Address error enabled
58	OFE	Output FIFO Error. The AESU Output FIFO was detected non-empty upon write of AESU data size register 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
59	IFE	Input FIFO Error. The AESU Input FIFO was detected non-empty upon generation of DONE interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	—	Reserved
61	IFO	Input FIFO Overflow. The AESU Input FIFO has been pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62	OFU	Output FIFO Underflow The AESU Output FIFO has been read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

14.5.6.8 AESU End of Message Register (AESUEMR)

AESUEMR, shown in [Figure 14-58](#), is used to indicate 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 will be used to determine how many bits of the final message block (always 128) will be processed. Writing to this register causes the AESU to process the final block of a message, allowing it

to signal DONE. A read of this register always returns a zero. AESUEMR is used only when the is operated as a slave. The descriptors and crypto-channel activate the AESU (via an internally generated write to the end of message register) when the SEC acts as an initiator.

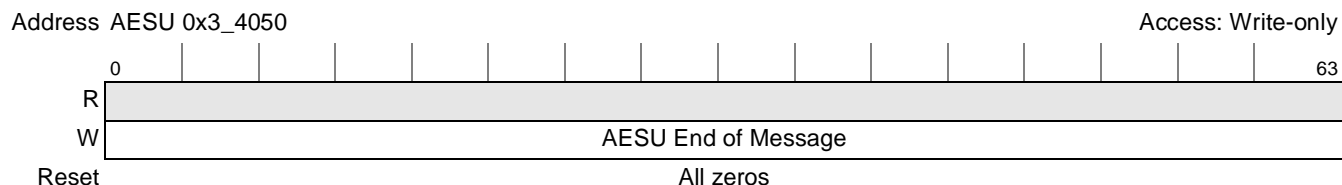


Figure 14-58. AESU End of Message Register

14.5.6.9 AESU Context Registers

There are three 64-bit context data registers that allow the host 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 will be 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 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 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.

Context should be loaded with the lower bytes in the lowest 64-bit context register. The context registers are summarized in [Figure 14-59](#).

Context Register (64-bits each)

Cipher Mode	1	2	3	4	5	6	7
ECB	—	—	—	—	—	—	—
CBC	IV1 ¹	IV2 ¹	—	—	—	—	—
CTR	—	—	—	—	Counter ¹		Counter Modulus ¹
SRT	Counter ¹		Counter Modulus ¹	—	—	—	—
CCM	IV ¹ / MAC Tag		Encrypted MAC ² /Decrypted MAC/Encrypted Counter		Counter ¹		Counter Modulus ^{1,3}

¹ Must be written at the start of a new message

² Must be written at start of new CCM decryption

³ Header size/MAC size is only used if AES-CCM processing is suspended and resumed.

Figure 14-59. AESU Context Register

14.5.6.9.1 Context for CBC Mode

Within the Context register, for use in CBC mode, are two 64-bit context data registers that allow the host to read/write the contents of the initialization vector (IV):

- IV1 holds the *most* significant bytes of the initialization vector (bytes 1–8).
- IV2 holds the *least* significant bytes of the initialization vector (bytes 9–16).

The IV must be written prior to the message data. If the IV registers are written during message processing, or the CBC mode bit is not set, a context error will be generated.

The IV registers may only be read after processing has completed, as indicated by the assertion of Interrupt Done DONE in the AESU status register as shown in [Section 14.5.6.5, “AESU Status Register \(AESUSR\).”](#) If the IV registers are read prior to assertion of Interrupt Done, an early read error will be generated.

The IV registers must be read when changing context and restored to resume processing an interrupted message (CBC mode only).

14.5.6.9.2 Context for Counter Mode

In counter mode, a random 128-bit initial counter value is incremented modulo 2^n with each block processed. The modulus size can be set between 2^8 through 2^{128} , by powers of 8. The running counter is encrypted and eXclusive-ORed with the plaintext to derive the ciphertext, or with the ciphertext to recover the plaintext.

In CTR mode, the block counter is incremented modulo 2^M . The value of M is specified by writing to Context Register 3 as described in [Table 14-45](#).

Value Written	Modulus	Value Written	Modulus
8	2^8	72	2^{72}
16	2^{16}	80	2^{80}
24	2^{24}	88	2^{88}
32	2^{32}	96	2^{96}
40	2^{40}	104	2^{104}
48	2^{48}	112	2^{112}
56	2^{56}	120	2^{120}
64	2^{64}	128	2^{128}

Table 14-45. Counter Modulus

14.5.6.9.3 Context for SRT Mode

As was noted in the AESU mode register, 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'. As with counter mode, a random 128-bit initial counter value is incremented modulo 2^n with each block processed. The modulus size can be set between 2^8 through 2^{128} , by powers of 8. The running counter is encrypted and eXclusive-ORed with the plaintext to derive the

ciphertext, or with the ciphertext to recover the plaintext. The block counter is incremented modulo 2^M . The value of M is specified by writing to Context Register 3 as described in [Table 14-45](#).

The only difference between SRT mode and CTR mode is 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.

14.5.6.9.4 Context for CCM Mode

The SEC AESU is capable of performing single pass encryption and MAC generation. The host is required to order the CCM context in such a way that the context can be fetched as a contiguous string into the context registers, prior to encryption/MAC generation or decryption/MAC validation. A complete explanation of the context and ordering can be found below.

The context for CCM encryption/MAC generation is:

Reg 1–2 Session specific 128 bit Initialization Vector (from memory)

Reg 3–4 128 bits of zero padding

Reg 5–6 Session Specific Counter (Initial Counter Value) (from memory)

Reg 7 Counter Modulus Should be fixed at 0x0000_0080.

Note: The counter modulus for CCM mode is currently defined as 2^{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.

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 128 bits of MAC data is written to context registers 1–2, for use in the next phase of CCM processing. 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.
4. 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 Reg 1–2) to produce the Encrypted MAC, which is then stored in Context Registers 3–4. At the completion of CCM encrypt processing, this Encrypted MAC is output to memory (per the descriptor pointer) for the host to append to the 802.11i frame. Note: The Encrypted MAC written out to memory by the AESU is the full 128-bits. The host must only append the most significant 64-bits to the frame as the encrypted MAC.

5. 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.
6. 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 encrypted MAC has been output, the CCM encrypt operation is complete.

The context for CCM decryption/MAC generation is as follows:

Reg 1–2 Session specific 128 bit Initialization Vector (from memory)

Reg 3–4 Encrypted MAC (from received frame) + 64 bits of zero padding

Reg 5–6 Session Specific Counter (Initial Counter Value) (from memory)

Reg 7 Counter Modulus Should be fixed at 0x0000_0080.

NOTE

The counter modulus for CCM mode is currently defined as 2^{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 will perform the following operations.

1. Initialize the IV, and encrypt 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 Reg 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 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 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.

14.5.6.9.5 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 re-initialized. 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.

14.5.6.9.6 AESU FIFOs

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.

Writing to the FIFO address space places 64 bits of message data into the input FIFO. The input FIFO may be written any time the IFW signal is asserted (as indicated in the AESU status register). This will indicate that the number of bytes of available space is at or above the threshold specified in the mode register. 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.

Reading from the FIFO address space will pop 64 bits of message data from the output FIFO. 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.

14.6 Crypto-Channels

A channel in the SEC manages the execution of each cryptographic task, making use of one or more of the SEC’s execution units (EUs). Control information and data pointers for a given task are stored in the form of a descriptor (see [Section 14.4.1, “Descriptor Structure”](#)) in system memory or in the channel itself. A descriptor determines what EUs will be used, how they will be configured, where to fetch needed data, and where to store the results. To invoke cryptographic tasks, the host constructs a descriptor, selects a channel, and writes a pointer to the descriptor into the selected channel’s Fetch FIFO. Operations performed by channels include the following (not necessarily in this order):

- If the channel is idle and its fetch FIFO is non-empty, read the next descriptor pointer from the Fetch FIFO, and use this pointer to read the descriptor into the channel’s descriptor buffer.
- Request from the controller the assignment of one or more EUs for the exclusive use of the channel. Where necessary, configure the secondary EU to snoop input or output data intended for the primary EU.

Table 14-46. CCCR Field Descriptions

Bits	Name	Description
0–29	—	Reserved, set to zero
30	CON	Continue bit 0 No special action. 1 Causes the same channel reset actions as bit R, except that the fetch FIFO and the lower half of the CCR register are not cleared. After the reset sequence is complete, this bit automatically returns to 0 and the channel resumes normal operation, servicing the next descriptor pointer in the fetch FIFO, if any.
31	R	Reset channel 0 No special action. 1 Causes a software reset of the channel, clearing all its internal state. 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 return the channel state machine to the idle state. If the R bit is set after the channel has been assigned 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 RESET bit and returns the channel state machine to the idle state.
32–54	—	Reserved, set to zero
55	BS	Burst size— The SEC accesses long text-data parcels in main memory through bursts of programmable size: 0 Burst size is 64 bytes 1 Burst size is 128 bytes
56	IWSE	ICV writeback status enable 0 No special action. 1 If the descriptor calls for ICV comparison, then at the completion of descriptor processing, write back the status of all EUs into the header dword.
57	AWSE	Always writeback status enable 0 No special action. 1 At the completion of processing each descriptor, write back the status of all EUs into the header dword. In this case, IWSE has no effect.
59	CDWE	Channel done writeback enable 0 Channel done writeback disabled. 1 Channel done writeback enabled. Upon completion of descriptor processing, if the NT bit is set for Global, or if the DN (Done Notification) bit is set in the header word of the descriptor, notify the host by writing back the descriptor header with the writeback information shown in Figure 14-61 . This enables the host to poll the memory location of the original descriptor header to determine if that descriptor has been completed.
60	—	Reserved, set to zero
61	NT	Notification type. This bit controls when the channel will generate channel done notification. Channel done notification can take the form of an interrupt or modified header writeback or both, depending on the state of the CDIE and CDWE control bits. 0 Global notification. The channel will generate channel done notification (if enabled) at the end of each descriptor. 1 Selected notification. The channel will generate channel done notification (if enabled) at the end of every descriptor with the DONE bit set in the descriptor header.

Table 14-46. CCCR Field Descriptions (continued)

Bits	Name	Description
62	CDIE	Channel done interrupt enable 0 Channel done interrupt disabled 1 Channel Done Interrupt enabled. Upon completion of descriptor processing, if the NT bit is set for Global, or if the DN (Done Notification) bit is set in the header word of the descriptor, then notify the host by asserting an interrupt. Refer to Section 14.6.2, "Interrupts," for complete description of channel interrupt operation.
63	—	Reserved, set to zero

Figure 14-61 shows the format of the header dword when the channel is configured to perform done notification via header writeback. A detailed description of the header dword fields used in writeback notification is provided in [Table 14-47](#).

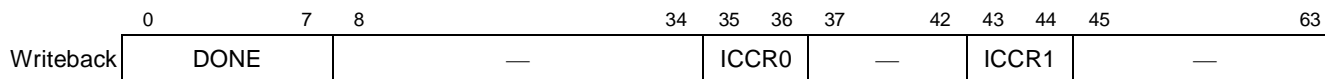


Figure 14-61. Header Dword Writeback Format

Table 14-47. Header Dword Writeback Field Descriptions

Bits	Name	Description
0–7	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 (see Table 14-47).
8–34	—	Reserved.
35–36	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
37–42	—	Reserved.
43–44	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
45–63	—	Reserved.

14.6.1.2 Crypto-Channel Pointer Status Register (CCPSR)

CCPSR contains status fields and counters which provide the user with status information regarding the channel's actual processing of a given descriptor.

Address Channel_1 0x01110 Access: Read-only
 Channel_2 0x01210
 Channel_3 0x01310
 Channel_4 0x01410

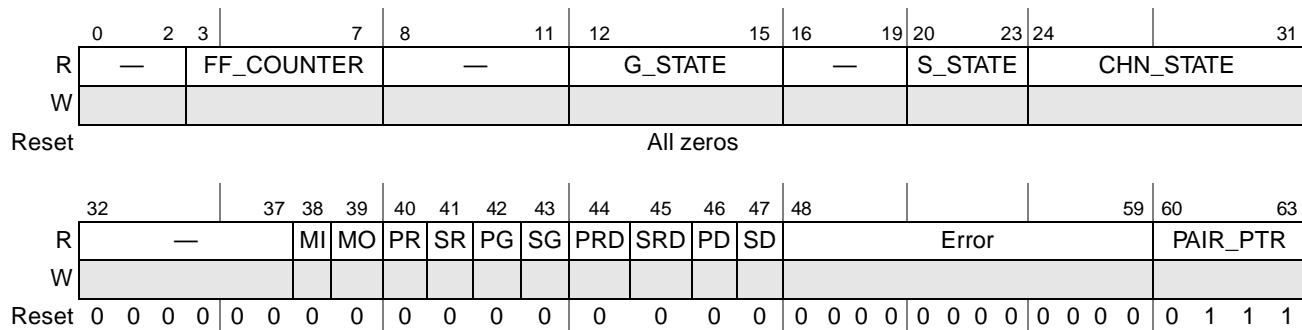


Figure 14-62. Crypto-Channel Pointer Status Register

Table 14-48 describes the crypto-channel pointer status register fields.

Table 14-48. Crypto-Channel Pointer Status Register Signals

Bits	Name	Description
0–2	—	Reserved.
3–7	FF_COUNTER	Fetch FIFO Counter. The fetch FIFO can store up to 24 pointers to descriptors in system memory. This 5 bit counter indicates how many fetch pointers are currently stored in the fifo.
8–11	—	Reserved.
12–15	G_STATE	Gather state machine state. Indicates which stage the crypto-channel is while performing the gather function. Table 14-49 shows the meaning of all possible values of G_STATE. Note: G_state is documented for information only. The user typically does not care about the gather state machine.
16–19	—	Reserved.
20–23	S_STATE	Scatter state machine state. Indicates which stage the crypto-channel is while performing the scatter function. Table 14-50 shows the meaning of all possible values of S_STATE. Note: S_state is documented for information only. The user typically does not care about the scatter state machine.
24–32	CHN_STATE	Crypto-channel state machine. Indicates the stage of the crypto-channel in the sequence of fetching and processing data descriptors. Table 14-49 shows the meaning of all possible values of STATE. Note: CHN_State is documented for information only. The user typically does not care about the crypto-channel state machine.
31–37	—	Reserved, set to zero
38	MI	Multi EU input. Reflects the type of snooping the channel will perform, as programmed by the 'Snoop Type' bit in the descriptor header. 0 Data input snooping by secondary EU disabled. 1 Data input snooping by secondary EU enabled.

Table 14-48. Crypto-Channel Pointer Status Register Signals (continued)

Bits	Name	Description
39	MO	Multi EU output. Reflects the type of snooping the channel will perform, as programmed by the 'Snoop Type' bit in the descriptor header. 0 Data output snooping by secondary EU disabled. 1 Data output snooping by secondary EU enabled.
40	PR	Request primary EU assignment. The PRI_REQ bit is set when descriptor processing is initiated in dynamic mode and the Op_0 field in the descriptor header contains a valid EU identifier. This bit is cleared when the request is granted, which will be reflected in the status register by the setting the PRI_GRANT bit. 0 Primary EU Assignment Request is inactive. 1 The crypto-channel is requesting assignment of primary EU to the channel. The channel will assert the EU request signal indicated by the op0 field in the Descriptor Header register as long as this bit remains set.
41	SR	Request secondary EU assignment. The SEC_REQ bit is set when descriptor processing is initiated in dynamic mode and the Op_1 field in the descriptor header contains a valid EU identifier. This bit is cleared when the request is granted, which will be reflected in the status register by the setting the SEC_GRANT bit. 0 Secondary EU Assignment Request is inactive. 1 The crypto-channel is requesting assignment of secondary EU to the channel. The channel will assert the EU request signal indicated by the Op_1 field in the descriptor header register as long as this bit remains set.
42	PG	Primary EU granted. The PRI_GRANT bit reflects the state of the EU grant signal for the requested primary EU from the controller. 0 The primary EU grant signal is inactive. 1 The EU grant signal is active indicating the controller has assigned the requested primary EU to the channel.
43	SG	Secondary EU granted. The SEC_GRANT bit reflects the state of the EU grant signal for the requested secondary EU from the controller. 0 The secondary EU grant signal is inactive. 1 The EU grant signal is active indicating the controller has assigned the requested secondary EU to the channel.
44	PRD	Primary EU reset done. The PRI_RST_DONE bit reflects the state of the reset done signal from the assigned primary EU. 0 The assigned primary EU reset done signal is inactive. 1 The assigned primary EU reset done signal is active indicating its reset sequence has completed and it is ready to accept data.
45	SRD	Secondary EU reset done. The SEC_RST_DONE bit reflects the state of the reset done signal from the assigned secondary EU. 0 The assigned secondary EU reset done signal is inactive. 1 The assigned secondary EU reset done signal is active indicating its reset sequence has completed and it is ready to accept data.
46	PD	Primary EU done. The PRI_DONE bit reflects the state of the DONE interrupt from the assigned primary EU. 0 The assigned primary EU done interrupt is inactive. 1 The assigned primary EU done interrupt is active indicating the EU has completed processing and is ready to provide output data.

Table 14-48. Crypto-Channel Pointer Status Register Signals (continued)

Bits	Name	Description
47	SD	Secondary EU done. The SEC_DONE bit reflects the state of the DONE interrupt from the assigned secondary EU. 0 The assigned secondary EU done interrupt is inactive. 1 The assigned secondary EU done interrupt is active indicating the EU has completed processing and is ready to provide output data.
48–59	ERROR	Crypto-channel error status. This field reflects the error status of the crypto-channel. When a channel error interrupt is generated, this field will reflect the source of the error. The bits in the ERROR field are registered at specific stages in the descriptor processing flow. Once registered, an error can only be cleared only by resetting the crypto-channel or writing the appropriate registers to initiate the processing of a new descriptor. Table 14-52 lists the conditions which can cause a crypto-channel error and how they are represented in the ERROR field.
60–63	PAIR_PTR	Descriptor buffer register length/pointer pair. This field indicates which of the length/pointer pairs are currently being processed by the channel. Table 14-53 shows the meaning of all possible values of the PAIR_PTR field.

[Table 14-49](#) shows the values of the gather state machine.

Table 14-49. G_STATE Field Values

Value	Gather State Machine	Value	Gather State Machine
0x0	idle	0x8	request_bytes_data_tarns_done
0x1	load_4pointers_frm_gather_table	0x9	increase_table_pointer
0x2	load_4pointers_frm_gather_table_done	0xA	update_gather_pointer
0x3	next_bit_set_load_next_gather_table	0xB	gather_table_done
0x4	process_gather_pointer	0xC	gather_error
0x5	request_block_data_trans	0xD	load_next_4pointers_frm_gather_table
0x6	request_block_data_trans_done	0xE–0xF	reserved
0x7	request_bytes_data_tarns	—	—

[Table 14-50](#) shows the values of the scatter state machine.

Table 14-50. S_STATE Field Values

Value	Scatter State Machine:
0x0	idle
0x1	load_4pointers_frm_scatter_table
0x2	load_4pointers_frm_scatter_table_done
0x3	next_bit_set_load_next_scatter_table
0x4	process_scatter_pointer
0x5	request_block_data_trans
0x6	request_block_data_trans_done
0x7	request_bytes_data_tarns

Table 14-50. S_STATE Field Values (continued)

Value	Scatter State Machine:
0x8	request_bytes_data_tarns_done
0x9	increase_table_pointer
0xA	update_scatter_pointer
0xB	scatter_table_done
0xC	scatter_error
0xD	load_next_4pointers_frm_scatter_table
0xE–0xF	reserved

Table 14-51 shows the values of crypto-channel states.

Table 14-51. CHN_STATE Field Values

Value	Channel State	Value	Channel State
0x00	IDLE	0x22	EVALUATE_RESET
0x01	PROCESS_HEADER	0x23	RESET_WRITE_RESET_PRI
0x02	FETCH_DESCRIPTOR	0x24	RESET_RELEASE_PRI_CHA
0x03	CHANNEL_DONE	0x25	RESET_WRITE_RESET_SEC
0x04	CHANNEL_DONE_IRQ	0x26	RESET_RELEASE_SEC_CHA
0x05	CHANNEL_DONE_WRITEBACK	0x27	RESET_CHANNEL
0x06	CHANNEL_DONE_NOTIFICATION	0x28	WRITE_DATASIZE_PRI_POST
0x07	CHANNEL_ERROR	0x29	RESET_RELEASE_ALL
0x08	REQUEST_PRI_CHA	0x2A	RESET_RELEASE_ALL_DELAY
0x09	INC_DATA_PAIR_POINTER	0x2B	REQUEST_SEC_CHA
0x0A	DELAY_DATA_PAIR_UPDATE	0x2C	WRITE_DATASIZE_SEC
0x0B	EVALUATE_DATA_PAIRS	0x2D	WRITE_ICV_SIZE
0x0C	WRITE_RESET_PRI	0x2E	WRITE_SEC_CHA_GO_SNOOPOUT
0x0D	RELEASE_PRI_CHA	0x2F	WRITE_PRI_CHA_GO_SNOOPIN
0x0E	WRITE_RESET_SEC	0x30	WRITE_SEC_CHA_GO_SNOOPIN
0x0F	RELEASE_SEC_CHA	0x31	DELAY_1CYCLE
0x10	PROCESS_DATA_PAIRS	0x33	TRANS_EXTENT_READ
0x11	WRITE_MODE_PRI	0x34	TRANS_EXTENT3
0x12	WRITE_MODE_SEC	0x35	TRANS_EXTENT4
0x13	WRITE_DATASIZE_PRI	0x36	XOR_WRITE_READ_REG
0x14	DELAY_RNGA_DONE	0x37	DELAY_SEC_DONE_TLS
0x15	WRITE_DATASIZE_SEC_SNOOPIN	0x38	MAC_TO_CIPHER
0x16	TRANS_REQUEST_WRITE_SNOOPIN	0x39	MAC_TO_CIPHER_DONE
0x17	DELAY_PRI_SEC_DONE	0x3A	READ_PRI_STATUS
0x18	TRANS_REQUEST_WRITE	0x3C	READ_SEC_STATUS
0x19	WRITE_KEY_SIZE	Others	Reserved
0x1B	DELAY_PRI_DONE		
0x1E	WRITE_DATASIZE_SEC_SNOOPOUT		
0x1F	TRANS_REQUEST_READ_SNOOPOUT		
0x20	DELAY_SEC_DONE		
0x21	TRANS_REQUEST_READ		

Table 14-52 shows the bit positions of each potential error. Multiple errors are possible.

Table 14-52. Crypto-Channel Pointer Status Register Error Field Definitions

Value	Error
48	DOF - Double Fetch Fifo write overflow Error. This bit is set when the Crypto-channel Fetch Fifo is full, SOF is set, and another write has been made to the fetch FIFO. When this bit is set the crypto-channel will stop, and an error interrupt will be activated. The Channel will not start again until a Continue or Reset is given via the CCCR Register. This bit can be cleared by writing '1' to this CCPSR bit.
49	SOF - Single Fetch Fifo write overflow Error. This bit is set when the Channel Fetch Fifo is full and another write has been made to the Fetch Fifo. The Crypto-channel will set this bit and activate an error interrupt. The Crypto-channel continues processing, but the descriptor pointer is lost. The host must clear this bit by writing '1' to this CCPSR bit.
50	MDTE- A Master Data Transfer Error was received from the master bus interface. When the SEC, while acting as a bus master, detects an 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 Continue or Reset bit in the Crypto-Channel Configuration Register, or resetting the whole SEC.
51	Scatter/Gather Data Length Zero Error- A zero length Scatter/Gather data pointer was detected.
52	Fetch Pointer Zero Error- An all zero Fetch Pointer was detected.
53	Illegal descriptor header- Possible causes of an illegal descriptor header are: <ul style="list-style-type: none"> • Invalid primary EU indicated by op0 field in descriptor header. • Invalid secondary EU indicated by op1 field in descriptor header. • Descriptor type field in descriptor header indicates secondary EU transaction when not in snoop mode
54	Invalid EU assignment request- Indicates the channel has been assigned one or more EUs not requested by the descriptor header.
55	EU error detected. An EU assigned to this channel has generated an error interrupt. This error may also be reflected in the controller's interrupt status register.
56	Gather boundary error- Indicates a gather pointer straddles both a primary and secondary EU's data transfer.
57	Gather return/length error- Indicates the total data size covered by a gather link table did not match the total data size from the main descriptor.
58	Scatter Boundary Error- Indicates a scatter pointer straddles both a primary and secondary EU's data transfer.
59	Scatter Return/Length Error- Indicates the total data size covered by a scatter link table did not match the total data size from the main descriptor.

NOTE

EU error bit (ERROR[0]) can only be cleared by first clearing the error source in the assigned EU which caused it to be set.

Table 14-53 shows the possible values of the PTR_DW field in the CCPSR.

Table 14-53. Channel Pointer Status Register PTR_DW Field Values

Value	Error
0x00	Processing Header or pointer dword 0
0x01	Processing pointer dword 1
0x02	Processing pointer dword 2
0x03	Processing pointer dword 3
0x04	Processing pointer dword 4

Table 14-53. Channel Pointer Status Register PTR_DW Field Values (continued)

Value	Error
0x05	Processing pointer dword 5
0x06	Processing pointer dword 6
0x07	Complete (or not yet begun) processing of header dword and pointer dwords
0x08– 0xFF	Reserved

14.6.1.3 Crypto-Channel Current Descriptor Pointer Register (CDPR)

The CDPR, shown in [Figure 14-63](#), contains the address of the descriptor which the crypto-channel is currently processing. CPDR, along with the PAIR_PTR in the CCPSR, can be used to determine if a new descriptor can be safely inserted into a chain of descriptors.

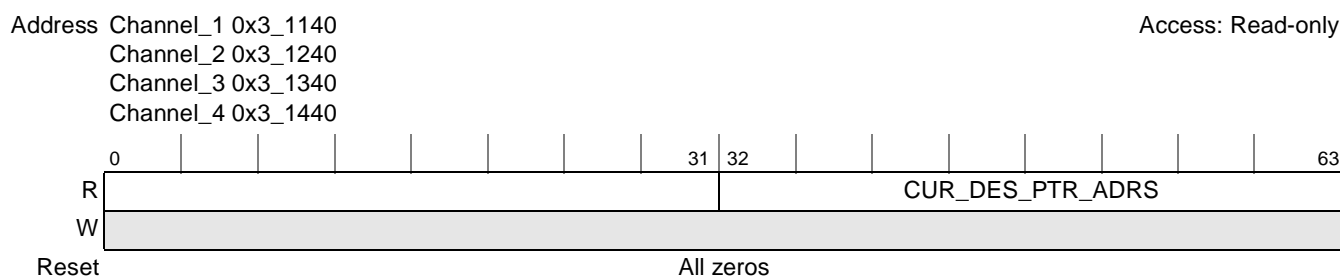


Figure 14-63. Crypto-Channel Current Descriptor Pointer Register

The bits in the current descriptor pointer register perform the functions described in [Table 14-54](#).

Table 14-54. Crypto-Channel Current Descriptor Pointer Register Signals

Bits	Name	Description
0–31	—	Reserved — Set to zero.
32–63	CUR_DES_PTR_ADRS	Current Descriptor Pointer Address. Pointer to 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 will be used as the destination of the writeback of the modified header dword, if header writeback notification is enabled.

14.6.1.4 Fetch FIFO (FF)

Each channel contains a fetch FIFO to store a queue of pointers to descriptors which the channel will process.

The fetch FIFO, shown in [Figure 14-64](#), contains the addresses of the first byte of descriptors to be processed. In typical operation, the host CPU will create a descriptor in memory containing all relevant mode and location information for the SEC, then ‘launch’ the SEC by writing the address of the descriptor to the 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 will be read to launch the next descriptor. Writing a descriptor pointer to the fetch FIFO while the FIFO is full will result in a single overflow interrupt to advise the user that the descriptor pointer was not successfully written to the fetch FIFO. The channel will continue processing and software can check the fetch FIFO counter in the crypto-channel pointer status register before attempting to re-enqueue the descriptor pointer. If a second descriptor pointer is written to the fetch FIFO before the single the single overflow error is cleared, the channel will generate a double overflow error interrupt and stop processing descriptors. The channel can be restarted by setting the Continue bit in the crypto-channel configuration register, or completely reset by writing the Reset bit in the same register.

The fetch address is written into the FIFO only if the write includes the least significant byte (bits 56–63).

Writing a fetch address of zero to the fetch FIFO causes the channel to generate an error and to stop.

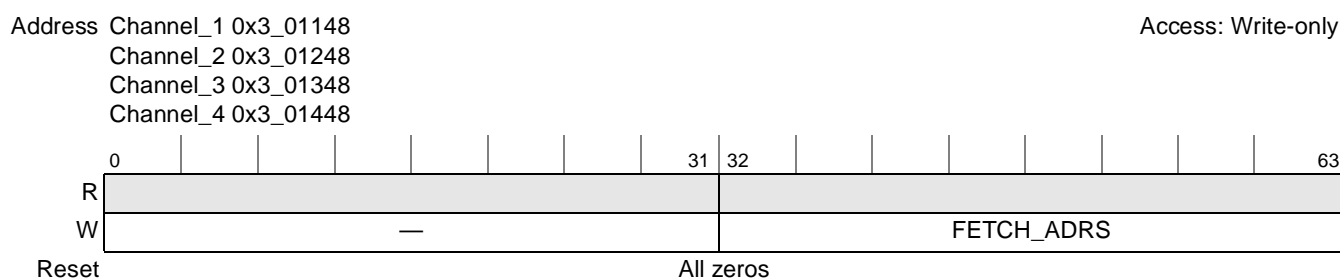


Figure 14-64. Fetch FIFO

Table 14-55 describes the Fetch FIFO fields.

Table 14-55. Fetch FIFO Field Descriptions

Bits	Name	Description
0–31	—	Reserved — Set to zero.
32–63	FETCH ADRS	Fetch Address. Pointer to system memory location of a descriptor the host wants the SEC to fetch.

14.6.1.5 Descriptor Buffer (DB)

The descriptor buffer (DB) consists of 8 dword registers (DB0–DB7), and contains the current descriptor being processed by the channel. These registers are read-only, since the descriptor is always fetched from system memory.

For more information about the fields in a descriptor, see [Section 14.4.1, “Descriptor Structure.”](#)

	0	15	16	17	23	24		31	32		63	
DB0	Header								Reserved			
DB1	Length0	J1	Extent0	—				Pointer0				
DB2	Length1	J2	Extent1	—				Pointer1				
DB3	Length2	J3	Extent2	—				Pointer2				
DB4	Length3	J4	Extent3	—				Pointer3				
DB5	Length4	J5	Extent4	—				Pointer4				
DB6	Length5	J6	Extent5	—				Pointer5				
DB7	Length6	J7	Extent6	—				Pointer6				
Address	Channel_1 0x3_1180–0x3_11BF, Channel_2 0x3_1280–0x3_12BF, Channel_3 0x3_1380–0x3_13BF, Channel_4 0x3_1480–0x3_14BF											

Figure 14-65. Data Packet Descriptor Buffer

14.6.2 Interrupts

The crypto-channel can assert both DONE and ERROR interrupts to the controller. When the interrupt generation conditions have been met, the crypto-channel will assert the appropriate interrupt. The status of the registered crypto-channel interrupts are available in the controller interrupt status register. The registered interrupts can be cleared by writing to the controller interrupt clear register. The crypto-channel does not have an internal interrupt mask bit and error interrupts are always asserted to the controller. The controller can be programmed to mask channel interrupts to the host via its interrupt mask register (IMR). See [Section 14.7.2.1, “Interrupt Mask Register \(IMR\).”](#)

14.6.2.1 Channel Done Interrupt

Whether and when a Channel Done Interrupt is generated depends on the setting of the Crypto-Channel Configuration Register NT and CDIE bits in the CCCR (see [Figure 14-60](#)). Assuming the CDIE (Channel Done Interrupt Enable) is set, the channel will generate an interrupt 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 word of the descriptor.

The Controller will queue multiple Channel Done interrupts if an interrupt is not cleared before the next Channel Done interrupt is issued. If there are multiple interrupts, the controller will re-assert the Channel Done interrupt one cycle after the previous Channel Done interrupt is cleared.

14.6.2.2 Channel Error Interrupt

The Channel Error Interrupt is generated when an error condition occurs during descriptor processing. The channel error interrupt will be 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 [Table 14-52](#) for a complete listing of error types.

14.6.2.3 Channel Reset

Channel reset is asserted when the host sets the RESET bit in the crypto-channel configuration register (CCCR). 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 crypto-channel is requesting a EU assignment from the controller, the crypto-channel will cancel its request by asserting the release output signals. The crypto-channel will then reset all the registers, clear the RESET bit and return the control state machine to the idle state.
- If the RESET bit is set after the crypto-channel has been dynamically assigned a EU, the channel will request a write from the controller to set the software reset bit of the EU. A write to reset the secondary (MDEU) EU will also be requested if one has been reserved for snooping. The crypto-channel will then assert the appropriate release output signal to notify the controller that the channel has finished with the reserved EU(s). The crypto-channel will then reset all the registers, clear the RESET bit and return the control state machine to the idle state.

14.7 Controller

The controller within the SEC is responsible for overseeing the operations of the execution units (EUs), the interface to the host processor, and the management of the crypto-channels. The controller interfaces to the host via the master/slave bus interface and to the channels and EUs via internal buses. All transfers between the host and the EUs are moderated by the controller. Some of the main functions of the controller are as follows:

- Arbitrate and control accesses to the bus
- Control the internal bus accesses to the EUs
- Arbitrate and assign EUs to the crypto-channels
- Monitor interrupts from channels and pass to host
- Realign read and write data to the proper byte alignment

14.7.1 Controller Registers

The controller registers are described in detail in the following sections.

14.7.2 EU Assignment Status Register (EUASR)

The EUASR, shown in [Figure 14-66](#), 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.

A four-bit field (see [Table 14-56](#)) indicates the channel to which an EU is assigned.

Offset 0x3_1028 Access: Read-only

	0	3	4	7	8	11	12	15	16	19	20	23	24	27	28	31																
R	—				AFEU			—			MDEU			—			AESU			—			DEU									
W																																
Reset	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
	32	47	48	51	52	55	56	59	60	63																						
R	—								—			PKEU			—			RNG														
W																																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0

Figure 14-66. EU Assignment Status Register (EUASR)

Table 14-56 shows the EU channel assignments.

Table 14-56. 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

14.7.2.1 Interrupt Mask Register (IMR)

The SEC controller generates the single interrupt output from all possible interrupt sources. These sources can be masked by the interrupt mask register. If unmasked, the interrupt source value, when active, is captured into the interrupt status register. Figure 14-67 shows the bit positions of each potential interrupt source. Each interrupt source is individually unmasked by setting its corresponding bit. At reset, all bits are masked. The bit fields are described in Table 14-57.

Address 0x3_1008

Access: Read/write

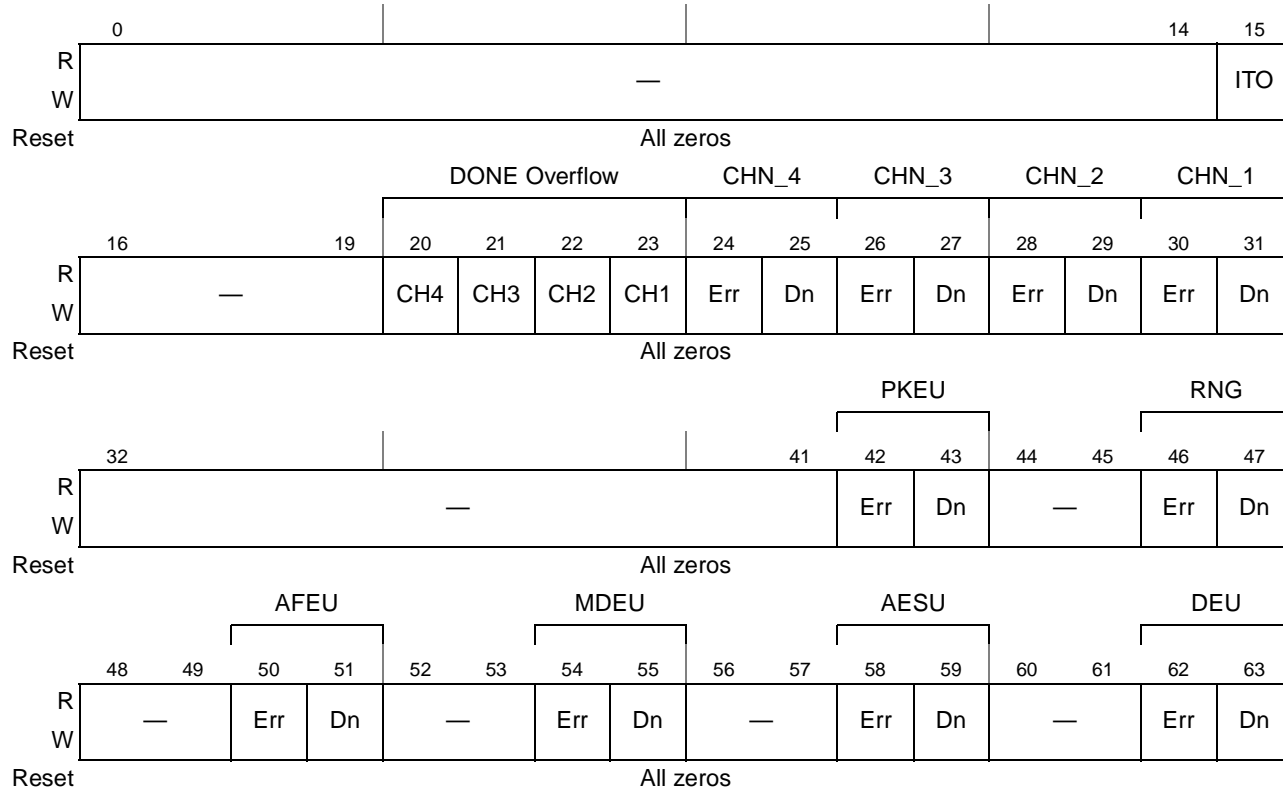


Figure 14-67. Interrupt Mask Register

14.7.2.2 Interrupt Status Register (ISR)

The ISR contains fields representing all possible sources of interrupts. It is cleared either by a reset or by writing the appropriate active ICR bits. Figure 14-68 shows the bit positions of each potential interrupt source. The bit fields are described in Table 14-57.

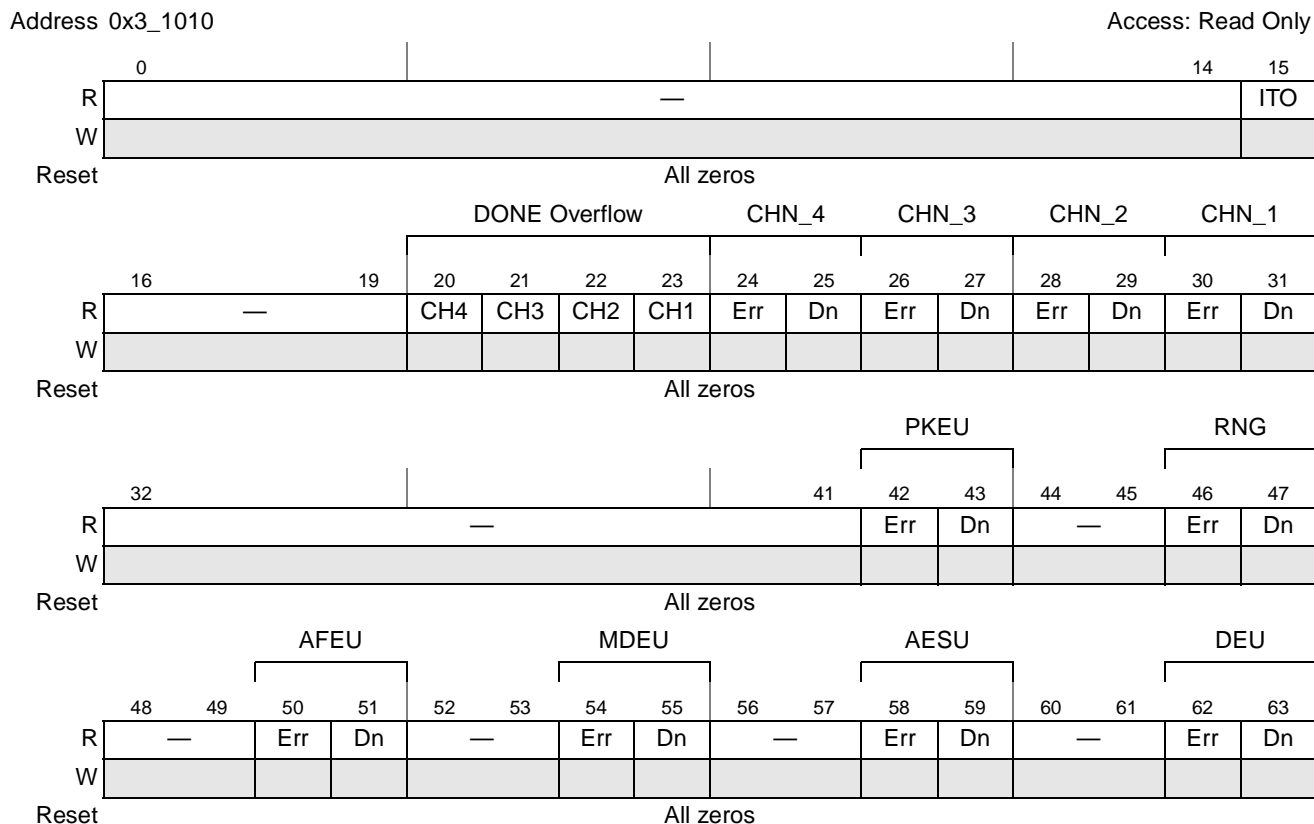


Figure 14-68. Interrupt Status Register

14.7.2.3 Interrupt Clear Register (ICR)

ICR provides a means of clearing the interrupt status register. When a bit in the ICR is written with a 1, the corresponding bit in the ISR is cleared, clearing the interrupt output pin \overline{IRQ} (assuming the cleared bit in the ISR is the only interrupt source). If the input source to the ISR is a steady-state signal that remains active, the appropriate ISR bit, and subsequently \overline{IRQ} , will be reasserted shortly thereafter. Figure 14-69 shows the bit positions of each interrupt source that can be cleared by ICR. The complete bit definitions for ICR can be found in Figure 14-69. ICR fields are described in Table 14-57.

When an ICR bit is written, it will automatically clear itself one cycle later. That is, it is not necessary to write a '0' to a bit position which has been written with a '1.'

NOTE

Interrupts are registered and sent based upon the conditions which cause them. If the cause of an interrupt is not removed, the interrupt will return a few cycles after it has been cleared using the ICR.

Address 0x3_1018

Access: Write-only

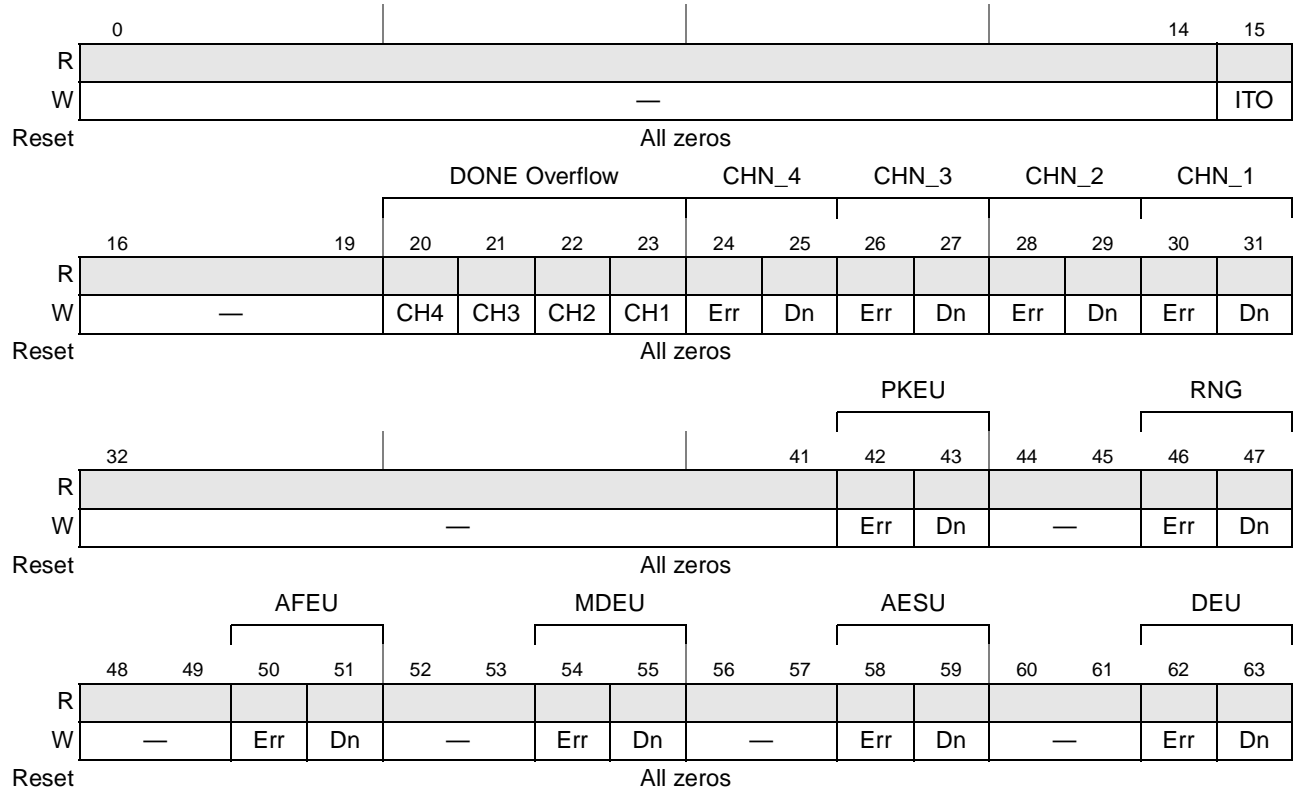


Figure 14-69. Interrupt Clear Register

Table 14-57 describes the interrupt mask, status, and clear register fields.

Table 14-57. Interrupt Mask, Status, and Clear Register Fields

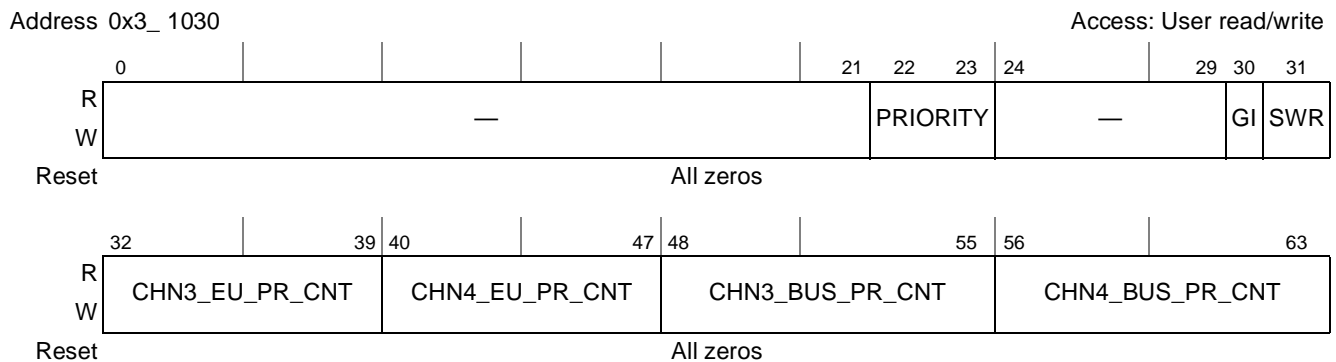
Bits	Name	Description
15	ITO	Internal Time Out 0 No Internal Time Out 1 An Internal Time Out was detected The Internal Time Out interrupt is triggered by the controller if a slave access to an SEC register does not result in successful data transfer within 16 clock cycles. With ITO enabled the SEC controller terminates the transaction and signals and interrupt.
20–23	Done Overflow	Done overflow 0 No Done Overflow 1 Done Overflow Error. Indicates that more than 15 Done interrupts were queued from the interrupting channel without an interrupt clear.
Multiple	CHN_Err_Dn	Each of the 4 channels has Error & Done bits. 0 No error detected. 1 Error detected. Indicates that execution unit status register must be read to determine exact cause of the error. 0 Not DONE. 1 DONE bit indicates that the interrupting channel or EU has completed its operation.

Table 14-58. IP Block Revision Register Fields (continued)

Bits	Name	Description
24–31	IP_MN	IP minor revision number
32–39	—	Reserved
40–47	IP_INT	IP block integration options
48–55	—	Reserved
56–63	IP_CFG	IP block configuration options

14.7.2.6 Master Control Register (MCR)

The MCR, shown in [Figure 14-72](#), controls certain functions in the controller and provides a way for software to reset the SEC.


Figure 14-72. Master Control Register

[Table 14-59](#) describes the master control register signals.

Table 14-59. Master Control Register Signals

Bits	Name	Description
0–21	—	Reserved
22–23	Priority	Priority on Master Bus. The setting of these bits determines the transaction priority level the SEC asserts to the device 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
24–29	—	Reserved
30	GI	Global Inhibit. 0 Master will always drive GBL_B active (low), meaning that the transactions are global and cache snooping is needed in order to enforce coherency. 1 Master will always drive GBL_B inactive (high).

Table 14-59. Master Control Register Signals (continued)

Bits	Name	Description
31	SWR	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 Don't reset 1 Global Reset Note: Warning: Certain SEC interrupts are not fully cleared by writing this bit. If SEC interrupts are pending, it is recommended that the user set this bit twice (two consecutive writes) to completely reset the SEC.
32–39	CHN3_EU_PR_CNT	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 top priority. Note: If set to zero, the CHN4_EU_PR_CTR must also be set to zero, and the controller will assign EUs on a pure round robin basis. If set to non-zero, CHN4_EU_PR_CTR must also be set to a different, non-zero value.
40–47	CHN4_EU_PR_CNT	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 top priority. Note: If set to zero, the CHN3_EU_PR_CTR must also be set to zero, and the controller will assign EUs on a pure round robin basis. If set to non-zero, CHN3_EU_PR_CTR must also be set to a different, non-zero value.
48–55	CHN3_BUS_PR_CNT	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 top priority. Note: If set to zero, the CHN4_BUS_PR_CTR must also be set to zero, and the controller will assign access to the bus on a pure round robin basis. If set to non-zero, CHN4_BUS_PR_CTR must also be set to a different, non-zero value.
56–63	CHN4_BUS_PR_CNT	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 top priority. Note: If set to zero, the CHN3_BUS_PR_CTR must also be set to zero, and the controller will assign access to the bus on a pure round robin basis. If set to non-zero, CHN3_BUS_PR_CTR must also be set to a different, non-zero value.

14.7.2.7 EU Access

Assignment of a EU function to a channel is done dynamically. With dynamic assignment, the channel requests a EU function, the controller checks to see if the requested EU function 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 will assert the grant signal pertaining to the request from the channel. The grant signal will remain asserted until the channel is done and releases the EU.

14.7.2.8 Multiple EU Assignment

In some cases, a channel may request two EUs. The channel will do this by first requesting the primary EU, then requesting the secondary EU. Once the controller has granted both EUs, this channel is then capable of requesting that the secondary EU snoop the bus. Snooping is described in [Section 14.8.3, “Snooping by Caches”](#).

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 which is requesting two EU functions.

14.7.2.9 Multiple Channels

Since there are multiple channels in the SEC, the controller must arbitrate for access to the execution units. To accomplish this, the controller implements an arbiter for each channel.

Each arbiter acts on either a weighted priority-based or round-robin scheme, depending on the values of CHN3_EU_PR_CNT and CHN4_EU_PR_CNT. If both CHN3_EU_PR_CNT and CHN4_EU_PR_CNT are set to a non-zero value, the arbiter will implement the weighted priority scheme. Otherwise, the arbitration will be round-robin. Setting only one of the CHN_EU_PR_CNT fields to a non-zero value causes unpredictable operation.

14.7.2.10 Priority Arbitration

When arbitrating on the priority scheme, the priority will be as follows:

- Channel 1—Highest priority
- Channel 2—Second highest priority, unless CHN3_EU_PR_CNT or CHN4_EU_PR_CNT expired
- Channel 3—Third priority, unless CHN4_EU_PR_CNT expired
- Channel 4—Lowest priority, until CHN4_EU_PR_CNT expired

For channels 1–4, the priority is channel 1, channel 2, channel 3, and channel 4, in that order. In order to prevent channels 3 and 4 from being locked out, the CHN3_EU_PR_CNT and CHN4_EU_PR_CNT fields are implemented in the Master Control Register. The value of these fields determines how many times channel 3 or channel 4 can be refused access to an EU in favor of a higher priority channel. A counter is implemented in the arbiter for each of these entities. When the channel has lost arbitration the number of times specified in its CHN_EU_PR_CNT field, then that channel has the 2nd highest priority when the requested EU becomes available. CHN1 always has the highest priority, but it cannot make back to back requests, so the 2nd highest priority channel will be serviced upon completion of the current CHN1 operation.

It is permissible for the CHN_EU_PR_CNT values to be different from the CHN_BUS_PR_CNT values, i.e., EU access may be prioritized, while bus access is pure round robin, and vice-versa.

14.7.2.11 Round-Robin Snapshot Arbiters

The controller implements seven ‘snapshot’ arbiters, one for each EU function, and one for the bus. Each arbiter takes a snapshot of the requests for its function. If there are requests, then the arbiter satisfies those requests via a round-robin scheme as the resource becomes available. When all requests have been satisfied, the arbiter takes another snapshot.

14.8 Bus Interface

The controller in the SEC (refer to [Section 14.7, “Controller”](#)) has the ability to be a bus master or a slave. This means that the controller can issue read and write commands to the bus, and it can also be written to and read from by the host.

The controller is the sole bus master in the SEC. All other modules are slave only devices. A channel may request access to system resources including the bus. In these cases, the channel must provide the starting address of the transfer for the bus(es) requested. All subsequent addresses are generated by the controller. All addresses will be sequential.

14.8.1 Bus Access

The controller attempts to maximize bus utilization by grouping outstanding bus requests from the channels by request type. The Controller will push all write requests to the Bus Interface, followed by all read requests, then repeat. Within a request type, the Controller will grant bus access via the same scheme that is used for granting EUs. When the CHN_BUS_PR_CNT values of both channel 3 and 4 are set to zero, round robin operation is in effect. In this case, the snapshot arbiter samples the requests for the bus, then grants those requests as the bus becomes available. For example, if channels 1, 2, and 4 are requesting bus access at a given time, the snapshot arbiter will register the three requests and ignore further requests. The buses will be granted to channel 1 until its transfer is completely satisfied. Then the buses will be granted to channel 2 until channel 2's transfer is completely satisfied. Finally, the buses will be granted to channel 4 until that transfer is completely satisfied. Then another snapshot of requests will be taken. Refer to [Section 14.7.2.10, “Priority Arbitration,”](#) for more information.

14.8.1.1 Master Read

The sequence for master read access is as follows:

1. Channel asserts its bus read request.
2. Channel furnishes address and transfer length.
3. Controller acknowledges request to channel.
4. Controller asserts request to Master interface.
5. Controller waits for 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 using the address supplied by the channel. Data may be realigned byte-wise by the controller if either:
 - the read did not begin on a 32-bit word boundary, or
 - the previous write to an execution unit's input FIFO did not end on a 32-bit word 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 has been read.

14.8.1.2 Master Write

Master writes are performed by transferring data from one of the EUs to the output FIFO in the controller, then transferring the data from the FIFO to the bus when the bus is granted to the controller. The sequence for a master bus write access is as follows:

1. Channel requests the bus from controller.
2. Channel furnishes address, transfer length.
3. Controller acknowledges request to channel.
4. Controller loads the write data into its FIFO, and waits for the bus to become available.
5. When the bus becomes available, controller writes data from its FIFO to the master interface.
6. Transfer continues until the bus write is completed and the controller has read all data from the appropriate internal address. The master interface will continue making bus requests until the full data length has been written.

It is probable that multiple bus bursts will be required to complete any given request. When a channel has been granted access to the bus, no other internal SEC requests to the bus will be acknowledged until that transfer has been fully satisfied; i.e., all bytes have been transferred.

14.8.1.2.1 Slave Access

As a bus slave, the controller simply responds to read and write commands from the bus. When it receives a write command, the controller takes the data from the slave interface and sends it to the internal location indicated by the address. For a read, the controller goes to the internal location and fetches the requested data from the specified address. The SEC internal memory space must be accessed modulo-4 boundaries to avoid invalid data or unpredictable operation.

14.8.2 Bus Arbitration Priority

Transaction priority is configured for all channels in the SEC master control register (MCR[Priority]) as shown in [Figure 14-72](#). The SEC does not dynamically adjust its transaction priority, however system software can dynamically adjust SEC transaction priority, with the change in priority taking effect immediately.

14.8.3 Snooping by Caches

If defined as global, SEC transactions can be snooped by the MPC8360E cache. This definition is programmed in the master control register MCR[GI]. See [14.7.2.6, “Master Control Register \(MCR\),”](#) for more details. Note that SEC transactions are defined as global by default.

14.8.4 Interrupts

The SEC generates a single interrupt to the programmable interrupt controller. Interrupts from the SEC are reported to the CPU if the mask bit in SIMSR_H[SEC] is set.

The user can control which events cause an interrupt by configuring the SEC interrupt mask register. These events are as follows:

- Done (of a channel or an execution unit)
- Error (of a channel or an execution unit)

When the user detects an interrupt request from the SEC, it should further read the SEC interrupt status register (ISR) to determine the interrupt source. To clear an interrupt, the user should write 1 to the corresponding bits in the SEC interrupt clear register (ICR).

Events may be further masked per channel by setting or clearing the related fields in the crypto-channel configuration registers. It is suggested that the user leave channel interrupts unmasked, while masking the interrupts from the EUs. Errors or Done signals coming from the EUs eventually cause the channel to signal an ERROR or DONE interrupt. Clearing an interrupt before eliminating the condition that caused the interrupt will cause the interrupt to be asserted again a few cycles later.

14.9 Power-Saving Mode

The SEC can be disabled by clearing SCCR[ENCCM]. See [Section 4.5.2.3, “System Clock Control Register \(SCCR\),”](#) for more information.

Chapter 15

I²C Interfaces

This chapter describes the two inter-IC (IIC or I²C) bus interfaces implemented on this device. Note that for most intents, the I²C interfaces are identical and are described as a single generic controller. Where necessary, differences between the two controllers are noted.

15.1 I²C Introduction

The inter-IC (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 this device and other devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. [Figure 15-1](#) shows a block diagram of the I²C interface.

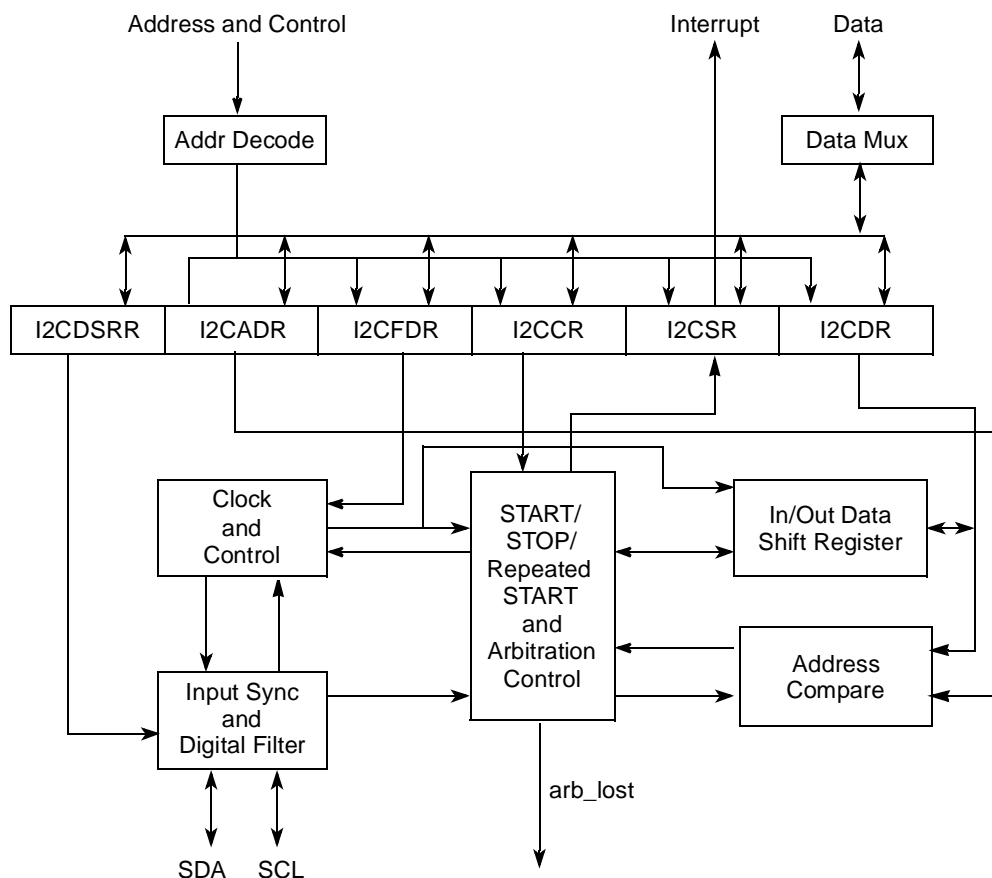


Figure 15-1. I²C Block Diagram

The two-wire I²C bus minimizes interconnections between devices. The synchronous, multiple-master I²C bus allows the connection of additional devices to the bus for expansion and system development. The bus

includes collision detection and arbitration that prevent data corruption if two or more masters attempt to control the bus simultaneously.

MPC8306 and MPC8306S have two instances of I²C controllers. I²C controller 1 is used for boot sequencing and I²C controller 2 is used for data communication.

15.1.1 I²C Features

Each I²C interface includes the following features:

- Two-wire interface
- Multiple-master operational
- Arbitration lost interrupt with automatic mode switching from master to slave
- 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

15.1.2 I²C Modes of Operation

The I²C unit on this device can operate in one of the following modes:

- Master mode. The I²C initiates a transfer, generates clock signals, and terminates a transfer. It cannot use its own slave address as a calling address. The I²C cannot be a master and a slave simultaneously.
- Slave mode. The I²C is addressed by an I²C master. The module must be enabled before a START condition from an I²C master is detected.
- Interrupt-driven byte-to-byte data transfer. When successful slave addressing is achieved (and SCL_n returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/ \overline{W} bit sent by the calling master. Each byte of data must be followed by an acknowledge bit, which is signaled from the receiving device. Several bytes can be transferred during a data transfer session.
- Boot sequencer mode. I²C1 controller supports boot sequencer mode. This mode can be used to initialize the configuration registers in the device after the I²C module is initialized. Boot sequencer mode is selected using the BOOTSEQ field in the reset configuration word high. Note that the hard-coded reset configuration word high value is boot sequencer mode disabled. This mode is not supported by I²C2 controller.
- Reset configuration load (I²C1 only). In this mode, the I²C1 interface loads the reset configuration words from an EEPROM at a specific calling address while the rest of the device is in the reset state ($\overline{\text{HRESET}}$ asserted). Once the reset configuration words are latched inside the device, I²C1 is reset until $\overline{\text{HRESET}}$ is negated. After $\overline{\text{HRESET}}$ is negated, the device may be initialized using boot

sequence mode according to the BOOTSEQ field in the reset configuration word. See [Section 15.4.5, “Boot Sequencer Mode.”](#)

Additionally, the following three I²C–specific states are defined for the I²C interface:

- **START condition.** This condition denotes the beginning of a new data transfer (each data transfer contains several bytes of data) and awakens all slaves.
- **Repeated START condition.** A START condition that is generated without a STOP condition to terminate the previous transfer.
- **STOP condition.** The master can terminate the transfer by generating a STOP condition to free the bus.

15.2 I²C External Signal Descriptions

The following sections give an overview of signals and provide detailed signal descriptions.

15.2.1 I²C Signal Overview

The I²C interface uses the SDA_{*n*} and SCL_{*n*} signals, described in [Table 15-1](#), for data transfer. Note that the signal patterns driven on SDA_{*n*} represent address, data, or read/write information at different stages of the protocol.

Table 15-1. I²C Interface Signal Descriptions

Signal Name	Idle State	I/O	State Meaning
Serial Clock (SCL1, SCL2)	High	I	When the I ² C module is idle or acts as a slave, SCL _{<i>n</i>} defaults as an input. The unit uses SCL _{<i>n</i>} to synchronize incoming data on SDA _{<i>n</i>} . The bus is assumed to be busy when SCL _{<i>n</i>} is detected low.
		O	As a master, the I ² C module drives SCL _{<i>n</i>} along with SDA _{<i>n</i>} when transmitting. As a slave, the I ² C module drives SCL _{<i>n</i>} negates for data pacing.
Serial Data (SDA1, SDA2)	High	I	When the I ² C module is idle or in a receiving mode, SDA _{<i>n</i>} defaults as an input. The unit receives data from other I ² C devices on SDA _{<i>n</i>} . The bus is assumed to be busy when SDA _{<i>n</i>} is detected low.
		O	When writing as a master or slave, the I ² C module drives data on SDA _{<i>n</i>} synchronous to SCL _{<i>n</i>} .

15.2.2 I²C Detailed Signal Descriptions

SDA_{*n*} and SCL_{*n*}, described in [Table 15-2](#), serve as a communication interconnect with other devices. All devices connected to these signals must have open-drain or open-collector outputs. The logic AND function is performed on both of these signals with external pull-up resistors. Refer to the hardware specifications for electrical characteristics.

Table 15-2. I²C Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description
SCL1, SCL2	I/O	Serial clock. Performs as an input when the device is programmed as an I ² C slave. SCL _n also performs as an output when the device is programmed as an I ² C master.
	O	As outputs for the bidirectional serial clock, these signals operate as described below.
		State Meaning
	I	As inputs for the bi-directional serial clock, these signals operate as described below.
State Meaning		Asserted/Negated—The I ² C unit uses this signal to synchronize incoming data on SDA _n . The bus is assumed to be busy when this signal is detected low.
SDA1, SDA2	I/O	Serial data. Performs as an input when the device is in a receiving mode. SDA _n also performs as an output signal when the device is transmitting (as an I ² C master or a slave).
	O	As outputs for the bi-directional serial data, these signals operate as described below.
		State Meaning
	I	As inputs for the bi-directional serial data, these signals operate as described below.
State Meaning		Asserted/Negated—Used to receive data from other devices. The bus is assumed to be busy when SDA _n is detected low.

15.3 I²C Memory Map/Register Definition

Table 15-3 lists the I²C-specific registers and their addresses.

Table 15-3. I²C Memory Map

Address	I ² C Register	Access	Reset	Section/Page
I²C Controller 1—Block Base Address 0x0_3000 I²C Controller 2—Block Base Address 0x0_3100				
0x0_3000	I2C1ADR—I ² C1 address register	R/W	0x00	15.3.1.1/15-5
0x0_3004	I2C1FDR—I ² C1 frequency divider register	R/W	0x00	15.3.1.2/15-6
0x0_3008	I2C1CR—I ² C1 control register	R/W	0x00	15.3.1.3/15-7
0x0_300C	I2C1SR—I ² C1 status register	R/W	0x81	15.3.1.4/15-8
0x0_3010	I2C1DR—I ² C1 data register	R/W	0x00	15.3.1.5/15-9
0x0_3014	I2C1DFSRR—I ² C1 digital filter sampling rate register	R/W	0x10	15.3.1.6/15-10
0x0_301C– 0x0_30FF	Reserved, should be cleared	—	—	—
0x0_3100	I2C2ADR—I ² C2 address register	R/W	0x00	15.3.1.1/15-5
0x0_3104	I2C2FDR—I ² C2 frequency divider register	R/W	0x00	15.3.1.2/15-6
0x0_3108	I2C2CR—I ² C2 control register	R/W	0x00	15.3.1.3/15-7
0x0_310C	I2C2SR—I ² C2 status register	R/W	0x81	15.3.1.4/15-8

Table 15-3. I²C Memory Map (continued)

Address	I ² C Register	Access	Reset	Section/Page
0x0_3110	I2C2DR—I ² C2 data register	R/W	0x00	15.3.1.5/15-9
0x0_3114	I2C2DFSRR—I ² C2 digital filter sampling rate register	R/W	0x10	15.3.1.6/15-10
0x0_311C– 0x0_31FF	Reserved, should be cleared	—	—	—

15.3.1 I²C Register Descriptions

This section describes the I²C registers in detail. Note that reserved bits should always be written with the value they return when read. That is, the register should be programmed by reading the value, modifying 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 does not apply to the I²C_n data register (I2C_nDR).

15.3.1.1 I²C_n Address Register (I2C_nADR)

Figure 15-2 shows the I2C_nADR register, which contains the address to which the I²C interface responds when addressed as a slave. Note that this is not the address that is sent on the bus during the address-calling cycle when the I²C module is in master mode.


Figure 15-2. I²C_n Address Register (I2C_nADR)

Table 15-4 describes the bit settings of I2C_nADR.

Table 15-4. I2C_nADR Field Descriptions

Bits	Name	Description
0–6	ADDR	Slave address. Contains the specific slave address that is used by the I ² C interface. Note that the default mode of the I ² C interface is slave mode for an address match. Note that an address match is one of the conditions that can cause I2C _n SR[MIF] to be set, signaling an interrupt pending condition.
7	—	Reserved, should be cleared

15.3.1.2 I²C_n Frequency Divider Register (I2CnFDR)

Figure 15-3 shows the bits of the I²C_n frequency divider register.

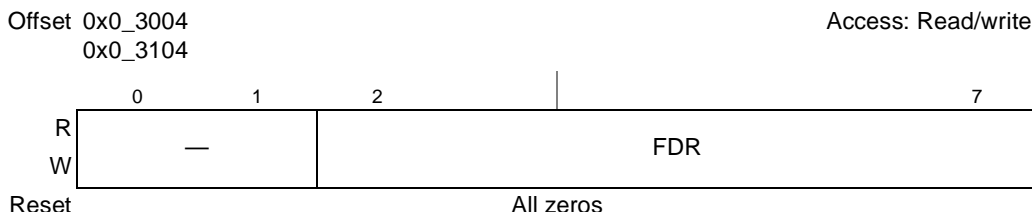


Figure 15-3. I²C_n Frequency Divider Register (I2CnFDR)

Table 15-5 describes the bit settings of I2CnFDR. It also maps I2CnFDR[FDR] to the clock divider values. Although it describes the ratio between the I²C controller internal clock and SCL, the default ratio of I²C controller clock and CSB is 1:1. Clock ratios of I²C1 are controllable but clock ratio for I²C2 is not and it is always 1:1 with CSB. Consider this factor when selecting an FDR value.

Table 15-5. I2Cn FDR Field Descriptions

Bits	Name	Description																																																																																																																																										
0–1	—	Reserved, should be cleared																																																																																																																																										
2–7	FDR	<p>Frequency divider ratio. Used to prescale the clock for bit-rate selection. The serial bit clock frequency of SCL_n is equal to the I²C_n controller clock divided by the divider. The serial bit clock frequency divider selections are described as follows:</p> <table border="0"> <thead> <tr> <th>FDR</th> <th>Divider (Decimal)</th> <th>FDR</th> <th>Divider (Decimal)</th> <th>FDR</th> <th>Divider (Decimal)</th> </tr> </thead> <tbody> <tr><td>0x00</td><td>384</td><td>0x16</td><td>12288</td><td>0x2B</td><td>1024</td></tr> <tr><td>0x01</td><td>416</td><td>0x17</td><td>15360</td><td>0x2C</td><td>1280</td></tr> <tr><td>0x02</td><td>480</td><td>0x18</td><td>18432</td><td>0x2D</td><td>1536</td></tr> <tr><td>0x03</td><td>576</td><td>0x19</td><td>20480</td><td>0x2E</td><td>1792</td></tr> <tr><td>0x04</td><td>640</td><td>0x1A</td><td>24576</td><td>0x2F</td><td>2048</td></tr> <tr><td>0x05</td><td>704</td><td>0x1B</td><td>30720</td><td>0x30</td><td>2560</td></tr> <tr><td>0x06</td><td>832</td><td>0x1C</td><td>36864</td><td>0x31</td><td>3072</td></tr> <tr><td>0x07</td><td>1024</td><td>0x1D</td><td>40960</td><td>0x32</td><td>3584</td></tr> <tr><td>0x08</td><td>1152</td><td>0x1E</td><td>49152</td><td>0x33</td><td>4096</td></tr> <tr><td>0x09</td><td>1280</td><td>0x1F</td><td>61440</td><td>0x34</td><td>5120</td></tr> <tr><td>0x0A</td><td>1536</td><td>0x20</td><td>256</td><td>0x35</td><td>6144</td></tr> <tr><td>0x0B</td><td>1920</td><td>0x21</td><td>288</td><td>0x36</td><td>7168</td></tr> <tr><td>0x0C</td><td>2304</td><td>0x22</td><td>320</td><td>0x37</td><td>8192</td></tr> <tr><td>0x0D</td><td>2560</td><td>0x23</td><td>352</td><td>0x38</td><td>10240</td></tr> <tr><td>0x0E</td><td>3072</td><td>0x24</td><td>384</td><td>0x39</td><td>12288</td></tr> <tr><td>0x0F</td><td>3840</td><td>0x25</td><td>448</td><td>0x3A</td><td>14336</td></tr> <tr><td>0x10</td><td>4608</td><td>0x26</td><td>512</td><td>0x3B</td><td>16384</td></tr> <tr><td>0x11</td><td>5120</td><td>0x27</td><td>576</td><td>0x3C</td><td>20480</td></tr> <tr><td>0x12</td><td>6144</td><td>0x28</td><td>640</td><td>0x3D</td><td>24576</td></tr> <tr><td>0x13</td><td>7680</td><td>0x29</td><td>768</td><td>0x3E</td><td>28672</td></tr> <tr><td>0x14</td><td>9216</td><td>0x2A</td><td>896</td><td>0x3F</td><td>32768</td></tr> <tr><td>0x15</td><td>10240</td><td></td><td></td><td></td><td></td></tr> </tbody> </table> <p>Note: The value's shown in the table are applicable only for the default value of DFSRR. Refer to AN2919.</p>	FDR	Divider (Decimal)	FDR	Divider (Decimal)	FDR	Divider (Decimal)	0x00	384	0x16	12288	0x2B	1024	0x01	416	0x17	15360	0x2C	1280	0x02	480	0x18	18432	0x2D	1536	0x03	576	0x19	20480	0x2E	1792	0x04	640	0x1A	24576	0x2F	2048	0x05	704	0x1B	30720	0x30	2560	0x06	832	0x1C	36864	0x31	3072	0x07	1024	0x1D	40960	0x32	3584	0x08	1152	0x1E	49152	0x33	4096	0x09	1280	0x1F	61440	0x34	5120	0x0A	1536	0x20	256	0x35	6144	0x0B	1920	0x21	288	0x36	7168	0x0C	2304	0x22	320	0x37	8192	0x0D	2560	0x23	352	0x38	10240	0x0E	3072	0x24	384	0x39	12288	0x0F	3840	0x25	448	0x3A	14336	0x10	4608	0x26	512	0x3B	16384	0x11	5120	0x27	576	0x3C	20480	0x12	6144	0x28	640	0x3D	24576	0x13	7680	0x29	768	0x3E	28672	0x14	9216	0x2A	896	0x3F	32768	0x15	10240				
FDR	Divider (Decimal)	FDR	Divider (Decimal)	FDR	Divider (Decimal)																																																																																																																																							
0x00	384	0x16	12288	0x2B	1024																																																																																																																																							
0x01	416	0x17	15360	0x2C	1280																																																																																																																																							
0x02	480	0x18	18432	0x2D	1536																																																																																																																																							
0x03	576	0x19	20480	0x2E	1792																																																																																																																																							
0x04	640	0x1A	24576	0x2F	2048																																																																																																																																							
0x05	704	0x1B	30720	0x30	2560																																																																																																																																							
0x06	832	0x1C	36864	0x31	3072																																																																																																																																							
0x07	1024	0x1D	40960	0x32	3584																																																																																																																																							
0x08	1152	0x1E	49152	0x33	4096																																																																																																																																							
0x09	1280	0x1F	61440	0x34	5120																																																																																																																																							
0x0A	1536	0x20	256	0x35	6144																																																																																																																																							
0x0B	1920	0x21	288	0x36	7168																																																																																																																																							
0x0C	2304	0x22	320	0x37	8192																																																																																																																																							
0x0D	2560	0x23	352	0x38	10240																																																																																																																																							
0x0E	3072	0x24	384	0x39	12288																																																																																																																																							
0x0F	3840	0x25	448	0x3A	14336																																																																																																																																							
0x10	4608	0x26	512	0x3B	16384																																																																																																																																							
0x11	5120	0x27	576	0x3C	20480																																																																																																																																							
0x12	6144	0x28	640	0x3D	24576																																																																																																																																							
0x13	7680	0x29	768	0x3E	28672																																																																																																																																							
0x14	9216	0x2A	896	0x3F	32768																																																																																																																																							
0x15	10240																																																																																																																																											

15.3.1.3 I²C_n Control Register (I2C_nCR)

Figure 15-4 shows the I²C_n control register.

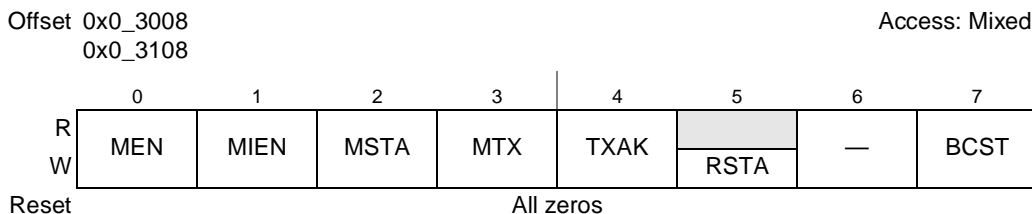


Figure 15-4. I²C_n Control Register (I2C_nCR)

Table 15-6 describes the I2C_nCR bit settings.

Table 15-6. I2C_nCR Field Descriptions

Bits	Name	Description
0	MEN	Module enable. Controls the software reset of the I ² C module. 0 The module is reset and disabled. The interface is held in reset, but the registers can still be accessed. 1 The I ² C module is enabled. MEN must be set before any other control register bits have any effect. All I ² C registers for slave receive or master START can be initialized before setting this bit.
1	MIEN	Module interrupt enable 0 Interrupts from the I ² C module are disabled. This does not clear any pending interrupt conditions. 1 Interrupts from the I ² C module are enabled. An interrupt occurs provided I2C _n SR[MIF] is also set.
2	MSTA	Master/slave mode START 0 On a transition to zero, a STOP condition is generated and the mode changes from master to slave. Cleared without generating a STOP condition when the master loses arbitration. 1 When MSTA changes from zero to one, a START condition is generated on the bus and master mode is selected.
3	MTX	Transmit/receive mode select. Selects the direction of the master and slave transfers. When configured as a slave, this bit should be set by software according to I2C _n SR[SRW]. In master mode, the bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always high. MTX is cleared when the master loses arbitration. 0 Receive mode 1 Transmit mode
4	TXAK	Transfer acknowledge. Specifies the value driven onto the SDA _n line during acknowledge cycles for both master and slave receivers. The value of this bit applies only when the I ² C module is configured as a receiver, not a transmitter. It also does not apply to address cycles; when the device is addressed as a slave, an acknowledge is always sent. 0 An acknowledge signal (low value on SDA _n) is sent out to the bus at the 9th clock bit after receiving one byte of data. 1 No acknowledge signal response (high value on SDA _n) is sent.
5	RSTA	Repeated START. Note that this bit is not readable, which means if a read is performed to RSTA, a zero value is returned. 0 No START condition is generated 1 Setting this bit always generates a repeated START condition on the bus, provides the device with the current bus master. Attempting a repeated START at the wrong time (or if the bus is owned by another master), results in loss of arbitration.

Table 15-6. I2CnCR Field Descriptions (continued)

Bits	Name	Description
6	—	Reserved, should be cleared
7	BCST	Broadcast 0 Disables the broadcast accept capability 1 Enables the I ² C to accept broadcast messages at address zero

15.3.1.4 I²Cn Status Register (I2CnSR)

I2CSR is shown in [Figure 15-5](#).

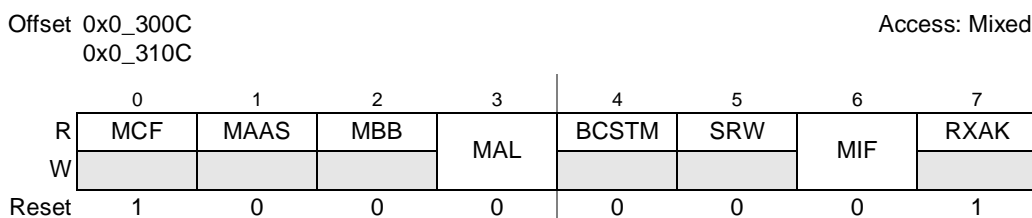


Figure 15-5. I²Cn Status Register (I2CnSR)

[Table 15-7](#) describes the bit settings of the I2CnSR.

Table 15-7. I2CnSR Field Descriptions

Bits	Name	Description
0	MCF	Data transfer. When one byte of data is transferred, the bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. 0 Byte transfer in progress. MCF is cleared under the following conditions: <ul style="list-style-type: none"> When I2CnDR is read in receive mode or when I2CnDR is written in transmit mode. After a start sequence is recognized by the I²C controller in slave mode. 1 Byte transfer is completed
1	MAAS	Addressed as a slave. When the value in I2CnADR matches the calling address or when the calling address is the broadcast address and broadcast mode is enabled (I2CnCR[BCST] is set), this bit is set. The processor is interrupted if I2CnCR[MIE] is set. Next, the processor must check the SRW bit and set I2CnCR[MTX] accordingly. Writing to the I2CnCR automatically clears this bit. 0 Not addressed as a slave 1 Addressed as a slave
2	MBB	Bus busy. Indicates the status of the bus. When a START condition is detected, MBB is set. If a STOP condition is detected, it is cleared. 0 I ² C bus is idle 1 I ² C bus is busy
3	MAL	Arbitration lost. Automatically set when the arbitration procedure is lost. Note that the device does not automatically retry a failed transfer attempt. 0 Arbitration is not lost. Can only be cleared by software 1 Arbitration is lost
4	BCSTM	Broadcast match. Writing to the I2CnCR automatically clears this bit. 0 There has not been a broadcast match. 1 The calling address matches with the broadcast address and broadcast mode is enabled. This is also set if this I ² C drives an address of all 0s.

Table 15-7. I2CnSR Field Descriptions (continued)

Bits	Name	Description
5	SRW	Slave 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 master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave. This bit is valid only when both of the following conditions are true: <ul style="list-style-type: none"> • A complete transfer occurred and no other transfers have been initiated. • The I²C interface is configured as a slave and has an address match. By checking SRW, the processor can select slave transmit/receive mode according to the command of the master.
6	MIF	Module interrupt. The MIF bit is set when an interrupt is pending, causing a processor interrupt request (provided I2CnCR[MIE] is set). 0 No interrupt is pending. Can be cleared only by software. 1 Interrupt is pending. MIF is set when one of the following events occurs: <ul style="list-style-type: none"> • One byte of data is transferred (set at the falling edge of the 9th clock). • The value in I2CnADR matches with the calling address in slave-receive mode. • Arbitration is lost.
7	RXAK	Received acknowledge. The value of SDA _n during the reception of acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates that an acknowledge signal has been received after the completion of eight bits of data transmission on the bus. If RXAK is high, it means no acknowledge signal has been detected at the 9th clock. 0 Acknowledge received 1 No acknowledge received

15.3.1.5 I²Cn Data Register (I2CnDR)

The I2Cn data register is shown in [Figure 15-6](#).

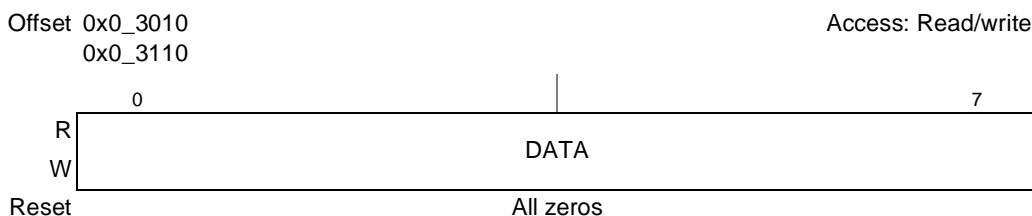


Figure 15-6. I²Cn Data Register (I2CnDR)

[Table 15-8](#) shows the bit descriptions for I2CnDR.

Table 15-8. I2CnDR Field Descriptions

Bits	Name	Description
0–7	DATA	Transmission starts when an address and the R/W bit are written to the data register and the I ² C interface performs as the master. A data transfer is initiated when data is written to the I2CnDR. The most-significant bit is sent first in both cases. In master receive mode, reading the data register allows the read to occur, but also allows the I ² C module to receive the next byte of data on the I ² C interface. In slave mode, the same function is available after it is addressed. Note that in both master receive and slave receive modes, the very first read is always a dummy read.

15.3.1.6 Digital Filter Sampling Rate Register (I2CnDFSRR)

I2CnDFSRR is shown in [Figure 15-7](#).

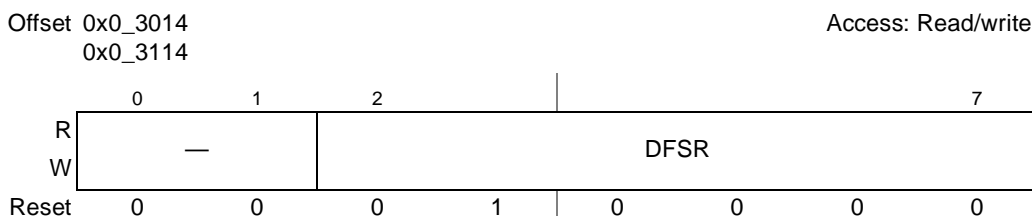


Figure 15-7. I²Cn Digital Filter Sampling Rate Register (I2CnDFSRR)

[Table 15-9](#) shows the I2CnDFSRR field descriptions.

Table 15-9. I2CnDFSRR Field Descriptions

Bits	Name	Description
0-1	—	Reserved, should be cleared
2-7	DFSRR	Digital filter sampling rate. To assist in filtering out signal noise, the sample rate is programmed. DFSRR is used to prescale the frequency at which the digital filter takes samples from the I ² C bus. The resulting sampling rate is calculated by dividing the platform frequency by the non-zero value of DFSRR. If I2CnDFSRR is cleared, the I ² C bus sample points default to the reset divisor 0x10.

15.4 Functional Description

The I²C unit always performs as a slave receiver as a default, unless explicitly programmed to be a master or slave transmitter. If boot sequencer mode is selected, the I²C interface performs as a slave receiver after the boot sequence has completed.

15.4.1 Transaction Protocol

A standard I²C transfer consists of the following:

- START condition
- Slave target address transmission
- Data transfer
- STOP condition

Figure 15-8 shows the interaction of these four parts with the calling address, data byte, and new calling address components of the I²C protocol. The details of the protocol are described in the following subsections.

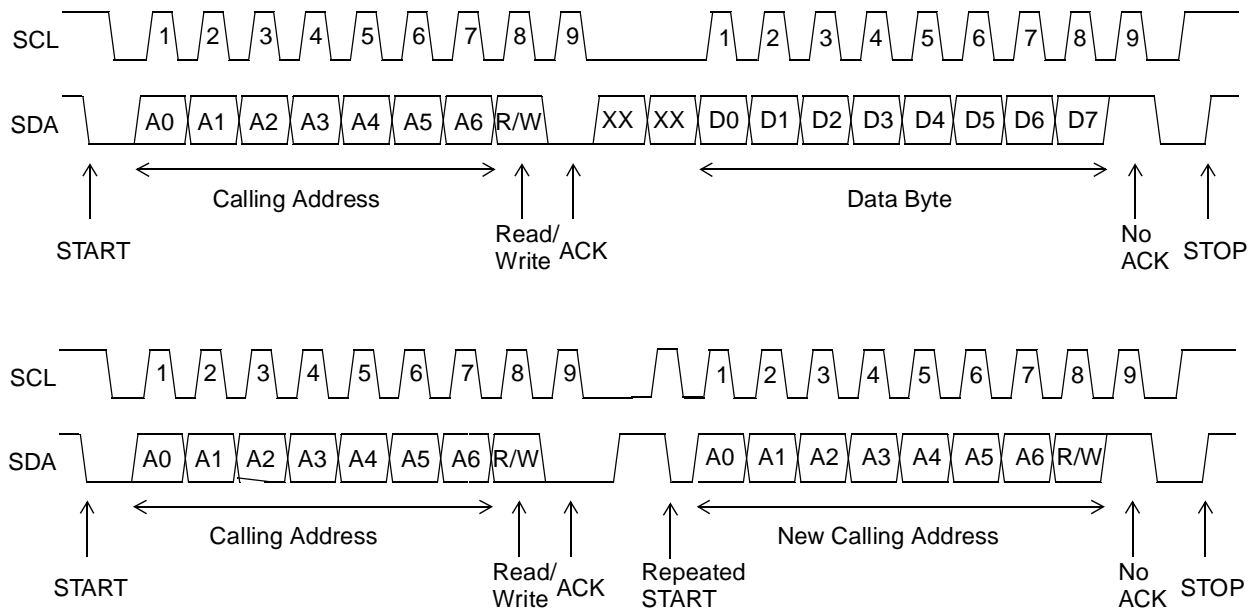


Figure 15-8. I²C Interface Transaction Protocol

15.4.1.1 START Condition

When the I²C bus is not engaged (both SDA_n and SCL_n lines are at logic high), a master can initiate a transfer by sending a START condition. As shown in Figure 15-8, a START condition is defined as a high-to-low transition of SDA_n while SCL_n is high. This condition denotes the beginning of a new data transfer. Each data transfer can contain several bytes and awakens all slaves. The START condition is initiated by a software write that sets I2CnCR[MSTA].

15.4.1.2 Slave Address Transmission

The first byte of data transferred by the master immediately after the START condition is the slave address. This is a seven-bit calling address followed by a $\overline{R/W}$ bit, which indicates the direction of the data transferred to the slave. Each slave in the system has a unique address. When the I²C module is operating as a master, it must not transmit an address that is the same as its slave address. An I²C device cannot be master and slave at the same time.

Only the slave with a calling address that matches the one transmitted by the master responds by returning an acknowledge bit (negating the SDA_n signal at the 9th clock) as shown in Figure 15-8. If no slave acknowledges the address, the master should generate a STOP condition or a repeated START condition.

When slave addressing is successful (and SCL_n returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the $\overline{R/W}$ bit sent by the calling master.

The I²C module responds to a general call (broadcast) command when I2CnCR[BCST] is set. A broadcast address is always zero; however the I²C module does not check the R/W bit. The second byte of the

broadcast message is the master address. Because the second byte is automatically acknowledged by hardware, the receiver device software must verify that the broadcast message is intended for itself by reading the second byte of the message. If the master address is for another receiver device and the third byte is a write command, software can ignore the third byte during the broadcast. If the master address is for another receiver device and the third byte is a read command, software must write 0xFF to I2CnDR with I2CnCR[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_n is low and must be held stable while SCL_n is high, as shown in [Figure 15-8](#). There is one clock pulse on SCL_n 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 SDA_n low at the 9th clock. Therefore, one complete data byte transfer takes 9 clock pulses. Several bytes can be transferred during a data transfer session.

If the slave receiver does not acknowledge the master, the SDA_n line must be left high by the slave. The master can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that the end-of-data has been reached. Then the slave releases the SDA_n line for the master to generate a STOP or a START condition.

15.4.1.3 Repeated START Condition

[Figure 15-8](#) shows a repeated START condition, which is generated without a STOP condition that can terminate the previous transfer. The master uses this method to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

15.4.1.4 STOP Condition

The master 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_n signal while SCL_n is high. For more information, see [Figure 15-8](#). Note that a master can generate a STOP even if the slave has transmitted an acknowledge bit, at which point the slave must release the bus. The STOP condition is initiated by a software write that clears I2CnCR[MSTA].

As described in [Section 15.4.1.3, “Repeated START Condition,”](#) the master 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.

15.4.1.5 Protocol Implementation Details

The following sections give details of how aspects of the protocol are implemented in the I²C module.

15.4.1.5.1 Transaction Monitoring—Implementation Details

The different conditions of the I²C data transfers are monitored as follows (see [Figure 15-8](#)):

- START conditions are detected when an SDA_n fall occurs while SCL_n is high.
- STOP conditions are detected when an SDA_n rise occurs while SCL_n is high.

- Data transfers in progress are canceled when a STOP condition is detected or if there is a slave address mismatch. Cancellation of data transactions resets the clock module.
- The bus is detected to be busy upon the detection of a START condition and idle upon the detection of a STOP condition.

15.4.1.5.2 Control Transfer—Implementation Details

The I²C module contains logic that controls the output to the serial data (SDA) and serial clock (SCL) lines of the I²C. The SCL_{*n*} output is pulled low as determined by the internal clock generated in the clock module. The SDA_{*n*} output can change only at the midpoint of a low cycle of the SCL_{*n*}, unless it is performing a START, STOP, or repeated START condition. Otherwise, the SDA_{*n*} output is held constant.

SDA_{*n*} is negated when one or more of the following conditions are true:

- Master mode
 - Data bit (transmit)
 - ACK bit (receive)
 - START condition
 - STOP condition
 - Repeated START condition
- Slave mode
 - Acknowledging address match
 - Data bit (transmit)
 - ACK bit (receive)

The SCL_{*n*} signal corresponds to the internal SCL_{*n*} signal when one or more of the following conditions are true in either master or slave mode:

- Master mode
 - Bus owner
 - Lost arbitration
 - START condition
 - STOP condition
 - Repeated START condition begin
 - Repeated START condition end
- Slave mode
 - Address cycle
 - Transmit cycle
 - ACK cycle

15.4.1.6 Address Compare—Implementation Details

The address compare block determines whether a slave has been properly addressed, either by its slave address or by the general broadcast address (which addresses all slaves). The following address comparisons are performed:

- Whether a broadcast message has been received, to update I2CnSR
- Whether the module has been addressed as a slave, to update I2CnSR and to generate an interrupt
- Whether the address transmitted by the current master matches the general broadcast address

15.4.2 Arbitration Procedure

The I²C interface is a true multiple-master bus. If two or more masters simultaneously try to control the bus, each master's clock synchronization procedure (including the I²C module) 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 masters. A bus master loses arbitration if it transmits a logic 1 on SDA_n while another master transmits a logic 0. The losing masters immediately switch to slave-receive mode and stop driving the SDA_n line. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, the I²C unit sets I2CnSR[MAL] to indicate the loss of arbitration and, as a slave, services the transaction if it is directed to itself.

If the I²C module is enabled in the middle of an ongoing byte transfer, the interface behaves as follows:

- Slave mode—the I²C module ignores the current transfer on the bus and starts operating whenever a subsequent START condition is detected.
- Master mode—the I²C module cannot tell whether the bus is busy; therefore, if a START condition is initiated, the current bus cycle can be corrupted. This ultimately causes in the current bus master to lose arbitration, after which bus operations return to normal.

15.4.2.1 Arbitration Control

The arbitration control block controls the arbitration procedure of the master mode. A loss of arbitration occurs whenever the master detects a 0 on the external SDA_n line while attempting to drive a 1, tries to generate a START or repeated START at an inappropriate time, or detects an unexpected STOP request on the line.

In master mode, arbitration by the master is lost (and I2CnSR[MAL] is set) under the following conditions:

- SDA_n samples low when the master drives high during an address or data-transmit cycle (transmit).
- SDA_n samples low when the master drives high during a data-receive cycle of the acknowledge (ACK) bit (receive).
- A START condition is attempted when the bus is busy.
- A repeated START condition is requested in slave mode.
- A repeated START condition is attempted when the requesting device is not the bus owner
- Unexpected STOP condition detected

Note that the I²C module does not automatically retry a failed transfer attempt.

15.4.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices can hold SCL_n low after completion of a 1-byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL_n line.

15.4.4 Clock Control

The clock control block handles requests from the clock signal for transferring and controlling data for multiple tasks.

A 9-cycle data transfer clock is requested for the following conditions:

- Master mode
 - Transmit slave address after START condition
 - Transmit slave address after repeated START condition
 - Transmit data
 - Receive data
- Slave mode
 - Transmit data
 - Receive data
 - Receive slave address after START or repeated START condition

15.4.4.1 Clock Synchronization

Due to the wire AND logic on the SCL_n line, a high-to-low transition on the SCL_n line affects all devices connected on the bus. The devices begin counting their low period when the master negates the SCL_n line. After a device has negated SCL_n , it holds the SCL_n 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 SCL_n if another device is still within its low period. Therefore, SCL_n 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 periods, SCL_n is released and asserted. Then there is no difference between the devices' clocks and the state of SCL_n , and all the devices begin counting their high periods. The first device to complete its high period negates SCL_n again.

15.4.4.2 Input Synchronization and Digital Filter

The following sections describes synchronization of the input signals and the filtering of SCL_n and SDA_n in detail.

15.4.4.2.1 Input Signal Synchronization

The input synchronization block synchronizes the input SCL_n and SDA_n signals to the system clock and detects transitions of these signals.

15.4.4.2.2 Filtering of SCL_n and SDA_n Lines

The SCL_n and SDA_n inputs are filtered to eliminate noise. Three consecutive samples of the SCL_n and SDA_n lines are compared to a pre-determined sampling rate. If they are all high, the output of the filter is high. If they are all low, the output is low. If they are any combination of highs and lows, the output is whatever the value of the line was in the previous clock cycle.

The sampling rate is equal to a binary value stored in the frequency register I2CDFSRR. The duration of the sampling cycle is controlled by a down counter. This allows a software write to the I2CDFSRR to control the filtered sampling rate.

15.4.4.3 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven SCL_n low, the slave can drive SCL_n low for the required period and then release it. If the slave SCL_n low period is greater than the master SCL_n low period, the resulting SCL_n low period is extended.

15.4.5 Boot Sequencer Mode

Boot sequencer mode is selected at power-on reset by the BOOTSEQ field of the high-order reset configuration word. If boot sequencer mode is selected, the I²C module communicates with one or more EEPROMs through the I²C interface. EEPROMs can be programmed to initialize one or more configuration registers. Note that as described in [Section 4.3.2.2.3, “Boot Sequencer Configuration,”](#) the default value for BOOTSEQ is 0b00, which corresponds to the I²C boot sequencer being disabled at power-up.

Boot sequencer mode also supports an extension of the standard I²C interface that uses more address bits to allow for EEPROM devices that have more than 256 bytes. This extended addressing mode is selectable using a different encoding in the BOOTSEQ field of the high-order reset configuration word (see [Section 4.3.2.2.3, “Boot Sequencer Configuration.”](#)) In this mode, only one EEPROM device can be used and the maximum number of registers is limited by the size of the EEPROM.

If the standard I²C interface is used, the I²C module addresses the first EEPROM, and reads 256 bytes. Then it issues a repeated start and addresses the next EEPROM address. This sequence continues until the CONT bit is cleared. If the last register is not detected before wrapping back to the first address, an error condition is detected. In other words, if the CONT bit for not cleared on the final 7 bytes, an error condition is detected, causing the I²C controller to hang. The I²C module continues to read from the EEPROM as long as the continue (CONT) bit is set in the EEPROM. The CONT bit resides in the address/attributes field that is transferred from the EEPROM, as described in [Section 15.4.5.2, “EEPROM Calling Address.”](#) There should be no other I²C traffic when the boot sequencer is active.

15.4.5.1 Using the Boot Sequencer for Reset Configuration

The reset configuration word can be loaded by using the I²C boot sequencer. See [Section 4.3.2.2.3, “Boot Sequencer Configuration.”](#)

Note that this usage does not prevent using the I²C boot sequencer to initiate the device in the normal functional mode, after reset state has completed. However, an I²C serial EEPROM of extended addressing type must be used and the first two EEPROM data structures must contain dedicated reset information.

15.4.5.2 EEPROM Calling Address

The EEPROM calling address is 0b101_0000. The first EEPROM to be addressed must be programmed to respond to this address, or an error is generated. Any additional EEPROMs are addressed in sequential order.

15.4.5.3 EEPROM Data Format

The I²C module expects a particular format for data to be programmed in the EEPROM. [Figure 15-9](#) shows an example of the EEPROM contents, including the preamble, data format, and CRC.

0	1	2	3	4	5	6	7	
1	0	1	0	1	0	1	0	Preamble
0	1	0	1	0	1	0	1	
1	0	1	0	1	0	1	0	
ACS	BYTE_EN			1	ADDR[12:13]			First Configuration Preload Command
ADDR[14:21]								
ADDR[22:29]								
DATA[0:7]								
DATA[8:15]								
DATA[16:23]								
DATA[24:31]								
ACS	BYTE_EN			1	ADDR[12:13]			Second Configuration Preload Command
ADDR[14:21]								
ADDR[22:29]								
DATA[0:7]								
DATA[8:15]								
DATA[16:23]								
DATA[24:31]								
.....								
ACS	BYTE_EN			1	ADDR[12:13]			Last Configuration Preload Command
ADDR[14:21]								
ADDR[22:29]								
DATA[0:7]								
DATA[8:15]								
DATA[16:23]								
DATA[24:31]								
0	0	0	0	0	0	0	0	End Command
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
CRC[0:7]								
CRC[8:15]								
CRC[16:23]								
CRC[24:31]								

Figure 15-9. EEPROM Contents

- A preamble should be the first 3 bytes programmed into the EEPROM. It should have a value of 0xAA55AA. The I²C checks to ensure that this preamble is correctly detected before proceeding.
- Following the preamble, there should be a series of configuration registers (known as register preloads). Each configuration register should be programmed according to a particular format, as shown in [Figure 15-10](#).

0	1	2	3	4	5	6	7
ACS	BYTE_EN			CONT	ADDR[12:13]		
ADDR[14:21]							
ADDR[12:29]							
DATA[0:7]							
DATA[8:15]							
DATA[16:23]							
DATA[24:31]							

Figure 15-10. EEPROM Data Format for One Register Preload Command

- The first byte holds alternate configuration space (ACS), byte enables, and continue (CONT) attributes.
- The 2 least-significant bits of the address are derived from the byte enables. address offset. Therefore, the address offset programmed into the EEPROM preload should be a word offset.
- The most significant 16 bits (assuming 36-bit addressing) of the address are prepended from either IMMRRBAR or alternate configuration space.
- After the first 3 bytes, 4 bytes of data should hold the desired value of the configuration register, regardless of the size of transaction.

Byte enables should be asserted for any byte that will be written, and they should be asserted contiguously, creating a 1, 2, or 4 byte write to a register. The boot sequencer assumes that a big-endian address is stored in the EEPROM. In addition, byte enable bit 0 (bit 1 of the byte) corresponds to the most-significant byte of data (data[0:7]), and byte enable bit 3 (bit 4 of the byte) corresponds to the least-significant byte of data (data[24:31]).

By asserting ACS, an alternate configuration space address is prepended to the write request from the boot sequencer according to the value in the ALTCBAR register. This will allow for external memories to be configured. Otherwise, IMMRBAR is prepended to the EEPROM address.

If the CONT bit is cleared, the first 3 bytes, including ACS, the byte enables, and the address, should be cleared 0. Also, the data contains the final CRC. A CRC-32 algorithm is used to check the integrity of the data. The following polynomial is used:

$$1 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$$

The CRC should cover all bytes stored in the EEPROM before the CRC. This includes the preamble, all register preloads, and the first 3 bytes of the last 7-byte preload (which should be all zeros).

15.4.5.4 Boot Sequencer Done Indication

Dedicated hardware is not provided to indicate whether the boot sequencer operation completed successfully. It is recommended to use one of the GPIO signals for that purpose. To do this, the last register preload programmed into the EEPROM should contain the address of the appropriate GPIO register and data that causes the setting of the required GPIO signal. The GPIO signal may be used for an external device or for debug purposes.

15.5 Initialization/Application Information

This section describes some programming guidelines recommended for the I²C interface. [Figure 15-11](#) is a recommended flowchart for I²C interrupt service routines.

A **sync** assembly instruction must be executed after each I²C register read/write access to guarantee that register accesses occur in order.

The I²C controller does not guarantee its recovery from all illegal I²C 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 I²C bus hangs. The recovery routine should also handle the case when the illegal I²C bus behavior causes the status bits returned after an interrupt to be inconsistent with what was expected.

15.5.1 Interrupt Service Routine Flowchart

[Figure 15-11](#) shows an example algorithm for an I²C interrupt service routine. Deviation from the flowchart may result in unpredictable I²C bus behavior. However, in the slave receive mode (not shown), the interrupt service routine may need to set I2CnCR[TXAK] when the next-to-last byte is to be accepted. It is recommended that a **sync** instruction follow each I²C register read or write to guarantee that register accesses occur in order.

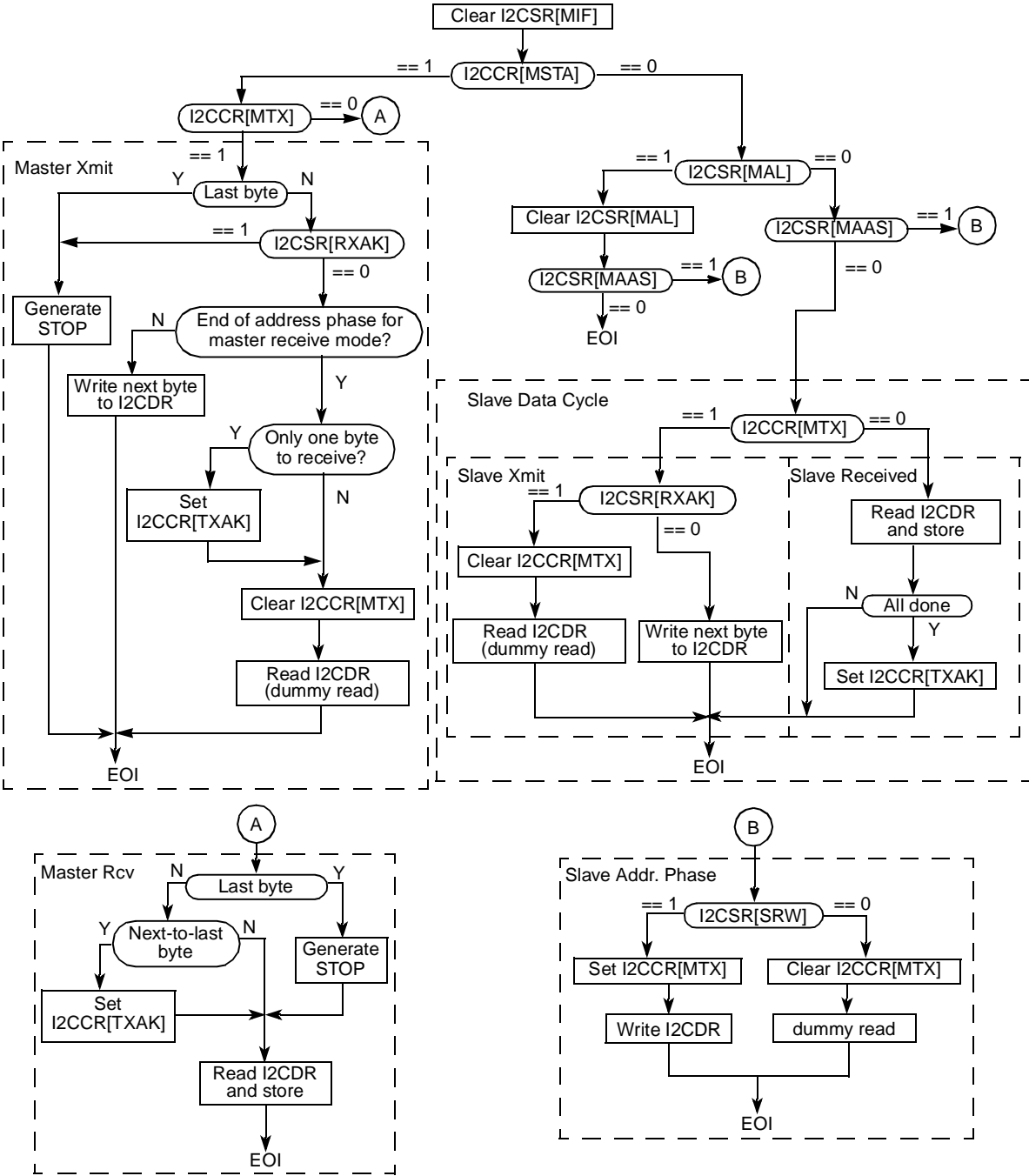


Figure 15-11. Example I²C Interrupt Service Routine Flowchart

15.5.2 Initialization Sequence

A hard reset initializes all of the I²C registers to their default states. The following initialization sequence initializes the I²C unit:

1. All I²C registers must be located in a cache-inhibited page.
2. Update I2CnFDR[FDR] and select the required division ratio to obtain the SCLn frequency from the CSB (platform) clock.
3. Update I2CnADR to define the slave address for this device.
4. Modify I2CnCR to select master/slave mode, transmit/receive mode, and interrupt-enable or disable.
5. Set the I2CnCR[MEN] to enable the I²C interface.

15.5.3 Generation of START

After initialization, the following sequence can be used to generate START:

1. If the device is connected to a multimaster I²C system, check whether the serial bus is free (I2CnSR[MBB] = 0) before switching to master mode.
2. Select master mode (set I2CnCR[MSTA]) to transmit serial data and select transmit mode (set I2CnCR[MTX]) for the address cycle.
3. Write the slave address being called into I2CnDR. The data written to I2CnDR[0–6] comprises the slave calling address. I2CnCR[MTX] indicates the direction of transfer (transmit/receive) required from the slave.

The scenario above assumes that the I²C interrupt bit (I2CnSR[MIF]) is cleared. If MIF is set at any time, an I²C interrupt is generated (provided interrupt reporting is enabled with I2CnCR[MIEN] = 1).

15.5.4 Post-Transfer Software Response

Transmission or reception of a byte automatically sets the data transferring bit (I2CnSR[MCF]), which indicates that one byte has been transferred. The I²C interrupt bit (I2CnSR[MIF]) is also set and an interrupt is generated to the processor if the interrupt function is enabled during the initialization sequence (I2CnCR[MIEN] is set). In the interrupt handler, software must take the following steps:

1. Clear I2CnSR[MIF]
2. Read the I2CnDR in receive mode or write to I2CnDR in transmit mode. Note that this causes I2CnSR[MCF] to be cleared, as shown in [Figure 15-11](#).
3. When an interrupt occurs at the end of the address cycle, the master remains in transmit mode. If master receive mode is required, I2CnCR[MTX] must be toggled at this stage (see [Figure 15-11](#)).

If the interrupt function is disabled, software can service the I2CnDR in the main program by monitoring I2CnSR[MIF]. In this case, I2CnSR[MIF] must be polled rather than I2CnSR[MCF] because MCF behaves differently when arbitration is lost. Note that interrupt or other bus conditions may be detected before the I²C signals have time to settle. Thus, when polling I2CnSR[MIF] (or any other I2CnSR bits), software delays may be needed to give the I²C signals sufficient time to settle.

During slave-mode address cycles (I2CnSR[MAAS] is set), I2CnSR[SRW] should be read to determine the direction of the subsequent transfer and I2CnCR[MTX] should be programmed accordingly. For slave-mode data cycles (MAAS is cleared), I2CnSR[SRW] is not valid and I2CnCR[MTX] must be read to determine the direction of the current transfer (see Figure 15-11).

15.5.5 Generation of STOP

A data transfer ends with a STOP condition generated by the master device. A master transmitter can generate a STOP condition after all the data has been transmitted.

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data (by setting the transmit acknowledge bit (I2CnCR[TXAK])) before reading the next-to-last byte of data. At this time, the next-to-last byte of data has been transferred on the I²C interface, so the last byte does not receive the data acknowledge (because I2CnCR[TXAK] is set). Before the interrupt service routine reads the last byte of data, a STOP condition must first be generated.

15.5.6 Generation of Repeated START

At the end of a data transfer, if the master still wants to communicate on the bus, it can generate another START condition followed by another slave address without first generating a STOP condition. This is accomplished by setting I2CnCR[RSTA].

15.5.7 Generation of SCLn When SDA_n is Negated

It is sometimes necessary to force the I²C module to become the I²C bus master out of reset and drive SCL_n (even though SDA_n may already be driven, which indicates that the bus is busy). This can occur when a system reset does not cause all I²C devices to be reset. Thus, SDA_n can be negated low by another I²C device while this I²C module is coming out of reset and will stay low indefinitely. The following procedure can be used to force this I²C module to generate SCL_n so that the device driving SDA_n can finish its transaction:

1. Disable the I²C module and set the master bit by setting I2CnCR to 0x20.
2. Enable the I²C module by setting I2CnCR to 0xA0.
3. Read I2CnDR.
4. Return the I²C module to slave mode by setting I2CnCR to 0x80.

15.5.8 Slave Mode Interrupt Service Routine

In the slave interrupt service routine, the module addressed as a slave should be tested to check if a calling of its own address has been received. If I2CnSR[MAAS] is set, software should set the transmit/receive mode select bit (I2CnCR[MTX]) according to the R \bar{W} command bit (I2CnSR[SRW]). Writing to I2CnCR clears MAAS automatically. MAAS is read as set only in the interrupt handler at the end of that address cycle where an address match occurred; interrupts resulting from subsequent data transfers clear MAAS. A data transfer can then be initiated by writing to I2CnDR for slave transmits or dummy reading from I2CnDR in slave-receive mode. The slave negates SCL_n between byte transfers. SCL_n is released when I2CnDR is accessed in the required mode.

15.5.8.1 Slave Transmitter and Received Acknowledge

In the slave transmitter routine, the received acknowledge bit (I2CnSR[RXAK]) must be tested before sending the next byte of data. The master signals an end-of-data by not acknowledging the data transfer from the slave. When no acknowledge is received (I2CnSR[RXAK] is set), the slave transmitter interrupt routine must clear I2CnCR[MTX] to switch the slave from transmitter to receiver mode. A dummy read of I2CnDR then releases SCL_n so that the master can generate a STOP condition. See [Figure 15-11](#).

15.5.8.2 Loss of Arbitration and Forcing of Slave Mode

When a master loses arbitration the following conditions all occur:

- I2CnSR[MAL] is set
- I2CnCR[MSTA] is cleared (changing the master to slave mode)
- An interrupt occurs (if enabled) at the falling edge of the 9th clock of this transfer

Thus, the slave interrupt service routine should first test I2CnSR[MAL] and software should clear it if it is set. See [Section 15.4.2.1, “Arbitration Control.”](#)

Chapter 16

DUART

This chapter describes the two (dual) universal asynchronous receiver/transmitters (UARTs) of the device. It describes the functional operation, the DUART initialization sequence, and the programming details for the DUART registers and features.

16.1 DUART Overview

The DUART consists of two (dual) universal asynchronous receiver/transmitters (UARTs). The UARTs act independently; all references to UART refer to one of these receiver/transmitters. Each UART is clocked by the system clock. The DUART programming model is compatible with the PC16552D.

The UART interface is point-to-point, meaning that only two UART devices are attached to the connecting signals. As shown in [Figure 16-1](#), each UART module consists of the following:

- Receive and transmit buffers
- Clear to send ($\overline{\text{CTS}}$) input port and request to send ($\overline{\text{RTS}}$) output port for data-flow control.
- 16-bit counter for baud rate generation
- Interrupt control logic

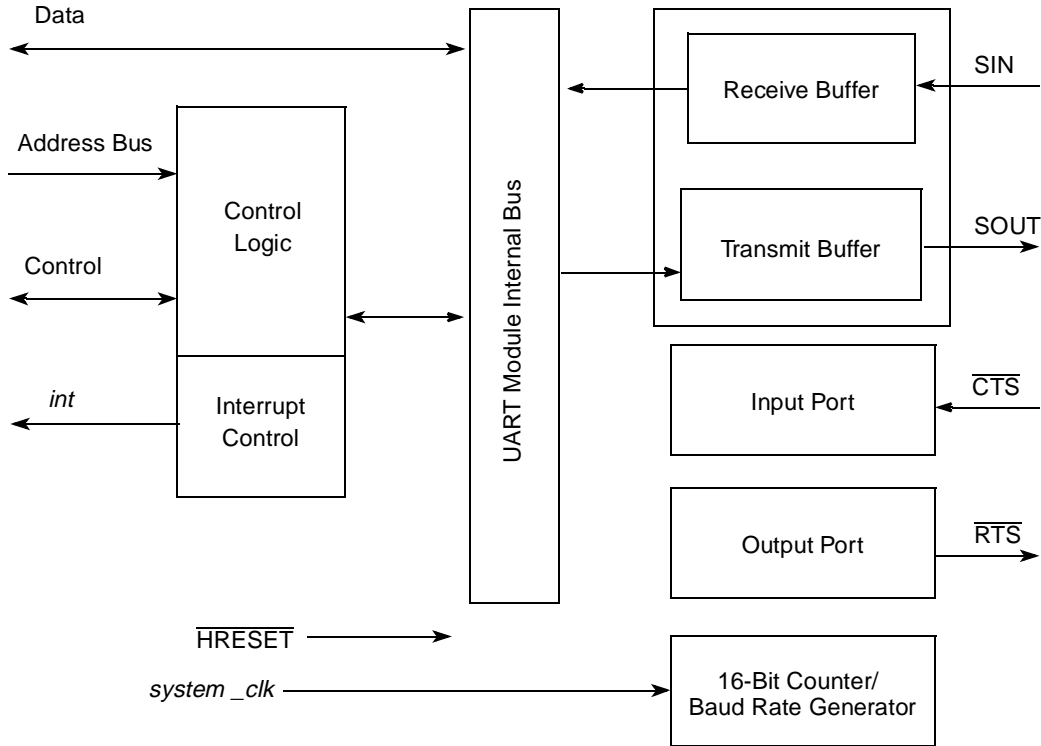


Figure 16-1. UART Block Diagram

16.1.1 DUART Features

The DUART includes these features:

- Full-duplex operation
- Programming model compatible with the original PC16450 UART and the PC16550D (an improved version of the PC16450 that also operates in FIFO mode)
- PC16450 register reset values
- FIFO mode for both transmitter and receiver, providing 16-byte FIFOs
- Serial data encapsulation and decapsulation with standard asynchronous communication bits (START, STOP, and parity)
- Maskable transmit, receive, line status, and MODEM status interrupts
- Software-programmable baud generators that divide the system clock by 1 to $(2^{16}-1)$ and generate a 16x clock for the transmitter and receiver engines
- Clear-to-send (\overline{CTS}) and ready-to-send (\overline{RTS}) MODEM control functions
- Software-selectable serial interface data format (data length, parity, 1/1.5/2 STOP bit, baud rate)
- Line and MODEM status registers
- Line-break detection and generation
- Internal diagnostic support, local loopback, and break functions
- Prioritized interrupt reporting
- Overrun, parity, and framing error detection

16.1.2 DUART Modes of Operation

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the system clock.

The transmitter accepts parallel data from a write to the transmitter holding register (UTHR). In FIFO mode, the data is placed directly into an internal transmitter shift register of the transmitter FIFO. The transmitter converts the data to a serial bit stream, inserting the appropriate START, STOP, and optional parity bits. Finally, it outputs a composite serial data stream on the channel transmitter serial data output signal (SOUT). The transmitter status may be polled or interrupt driven.

The receiver accepts serial data bits on the channel receiver serial data input signal (SIN), converts it to parallel format, checks for a START bit, parity (if any), STOP bits, and transfers the assembled character (with START, STOP, parity bits removed) from the receiver buffer (or FIFO) in response to a read of the UART's receiver buffer register (URBR). The receiver status may be polled or interrupt driven.

16.2 DUART External Signal Descriptions

This section contains a signal overview and detailed signal descriptions.

16.2.1 DUART Signal Overview

Table 16-1 summarizes the DUART signals. Note that although the actual device signal names are prepended with the 'UART_' prefix as shown in the table, the functional (abbreviated) signal names are often used throughout this chapter.

Table 16-1. DUART Signal Overview

Signal Name	I/O	Pins	Reset Value	State Meaning
UART_SIN[1:2]	I	2	1	Serial in data UART1 and UART2
UART_SOUT[1:2]	O	2	1	Serial out data UART1 and UART2
$\overline{\text{UART_CTS}}[1:2]$	I	2	1	Clear to send UART1 and UART2
$\overline{\text{UART_RTS}}[1:2]$	O	2	1	Request to send UART1 and UART2

16.2.2 DUART Detailed Signal Descriptions

The DUART signals are described in detail in Table 16-2.

Table 16-2. DUART Signals—Detailed Signal Descriptions

Signal	I/O	Description
UART_SIN[1:2]/DSP_UART_SIN	I	Serial data in. Data is received on the receivers of UART1, UART2, or DSP_UART through its respective serial data input signal, with the least significant bit received first.
		State Meaning Asserted/Negated—Represents the data being received on the UART interface.
		Timing Assertion/Negation—An internal logic sample signal, <i>rxcnt</i> , uses the frequency of the baud-rate generator to sample the data on SIN.

Table 16-2. DUART Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
UART_SOUT[1:2]/ DSP_UART_SOUT	O	Serial data out. The serial data output signals for the UART1, UART2, or DSP_UART are set (mark condition) when the transmitter is disabled, idle, or operating in the local loopback mode. Data is shifted out on these signals, with the least significant bit transmitted first.	
		State Meaning	Asserted/Negated—Represents the data transmitted on the respective UART interface.
		Timing	Assertion/Negation—An internal logic sample signal, <i>rxcnt</i> , uses the frequency of the baud-rate generator to update and drive the data on SOUT.
UART_CTS[1:2]	I	Clear to send. Connected to the respective $\overline{\text{RTS}}$ outputs of the other UART devices on the bus. They can be programmed to generate an interrupt on change-of-state of the signal.	
		State Meaning	Asserted/Negated—Represent the clear to send condition for their respective UART.
		Timing	Assertion/Negation—Sampled at the rising edge of every system clock.
UART_RTS[1:2]	O	Request to send. Can be programmed to be negated and asserted by either the receiver or transmitter. When connected to the $\overline{\text{CTS}}$ input of a transmitter, this signal can be used to control serial data flow.	
		State Meaning	Asserted/Negated—Represents the data being transmitted on the respective UART interface.
		Timing	Assertion/Negation—Updated and driven at the rising edge of every system clock.

16.3 DUART Memory Map/Register Definition

For MPC8306 and MPC8306S, there are four complete sets of DUART registers (one each for UART1, UART2, UART3, and UART4). The four UARTs are identical, except that the registers for:

- UART1 are located at offsets 0x0_4500 (local)
- UART2 are located at offsets 0x0_4600 (local)
- UART3 are located at offsets 0x0_4900 (local)
- UART4 are located at offsets 0x0_4A00 (local)

Throughout this chapter, the registers are described by a singular acronym: for example, LCR represents the line control register for either UART1, UART2, UART3, or UART4.

The registers in each UART interface are used for configuration, control, and status. The divisor latch access bit, ULCR[DLAB], is used to access the divisor latch least- and most-significant bit registers and the alternate function register. Refer to [Section 16.3.1.8, “Line Control Registers \(ULCR1 and ULCR2, ULCR3 and ULCR4\),”](#) for more information on ULCR[DLAB].

All DUART registers are one byte wide; reads and writes to these registers must be byte-wide operations. [Table 16-3](#) provides a register summary with references to the section and page that contain detailed

information about each register. Undefined byte address spaces within offset 0x4000–0x4FFF are reserved.

Table 16-3. DUART Register Summary

Offset	Register	Access	Reset	Section/Page
UART 1—Block Base Address 0x0_4000 UART 2—Block Base Address 0x0_4100				
0x0_4500	URBR—ULCR[DLAB] = 0 UART1 receiver buffer register	R	0x00	16.3.1.1/16-6
	UTHR—ULCR[DLAB] = 0 UART1 transmitter holding register	W	0x00	16.3.1.2/16-6
	UDLB—ULCR[DLAB] = 1 UART1 divisor least significant byte register	R/W	0x00	16.3.1.3/16-7
0x0_4501	UIER—ULCR[DLAB] = 0 UART1 interrupt enable register	R/W	0x00	16.3.1.4/16-8
	UDMB—ULCR[DLAB] = 1 UART1 divisor most significant byte register	R/W	0x00	16.3.1.3/16-7
0x0_4502	UIIR—ULCR[DLAB] = 0 UART1 interrupt ID register	R	0x01	16.3.1.5/16-9
	UFCR—ULCR[DLAB] = 0 UART1 FIFO control register	W	0x00	16.3.1.6/16-10
	UAFR—ULCR[DLAB] = 1 UART1 alternate function register	R/W	0x00	16.3.1.7/16-11
0x0_4503	ULCR—ULCR[DLAB] = x UART1 line control register	R/W	0x00	16.3.1.8/16-12
0x0_4504	UMCR—ULCR[DLAB] = x UART1 MODEM control register	R/W	0x00	16.3.1.9/16-14
0x0_4505	ULSR—ULCR[DLAB] = x UART1 line status register	R	0x60	16.3.1.10/16-15
0x0_4506	UMSR—ULCR[DLAB] = x UART1 MODEM status register	R	0x00	16.3.1.11/16-16
0x0_4507	USCR—ULCR[DLAB] = x UART1 scratch register	R/W	0x00	16.3.1.12/16-17
0x0_4510	UDSR—ULCR[DLAB] = x UART1 DMA status register	R	0x01	16.3.1.13/16-17
0x0_4600	URBR—ULCR[DLAB] = 0 UART2 receiver buffer register	R	0x00	16.3.1.1/16-6
	UTHR—ULCR[DLAB] = 0 UART2 transmitter holding register	W	0x00	16.3.1.2/16-6
	UDLB—ULCR[DLAB] = 1 UART2 divisor least significant byte register	R/W	0x00	16.3.1.3/16-7
0x0_4601	UIER—ULCR[DLAB] = 0 UART2 interrupt enable register	R/W	0x00	16.3.1.4/16-8
	UDMB—ULCR[DLAB] = 1 UART2 divisor most significant byte register	R/W	0x00	16.3.1.3/16-7
0x0_4602	UIIR—ULCR[DLAB] = 0 UART2 interrupt ID register	R	0x01	16.3.1.5/16-9
	UFCR—ULCR[DLAB] = 0 UART2 FIFO control register	W	0x00	16.3.1.6/16-10
	UAFR—ULCR[DLAB] = 1 UART2 alternate function register	R/W	0x00	16.3.1.7/16-11
0x0_4603	ULCR—ULCR[DLAB] = x UART2 line control register	R/W	0x00	16.3.1.8/16-12
0x0_4604	UMCR—ULCR[DLAB] = x UART2 MODEM control register	R/W	0x00	16.3.1.9/16-14
0x0_4605	ULSR—ULCR[DLAB] = x UART2 line status register	R	0x60	16.3.1.10/16-15
0x0_4606	UMSR—ULCR[DLAB] = x UART2 MODEM status register	R	0x00	16.3.1.11/16-16
0x0_4607	USCR—ULCR[DLAB] = x UART2 scratch register	R/W	0x00	16.3.1.12/16-17
0x0_4610	UDSR—ULCR[DLAB] = x UART2 DMA status register	R	0x01	16.3.1.13/16-17

16.3.1 DUART Register Descriptions

The following sections describe the UART1 and UART2 registers.

16.3.1.1 Receiver Buffer Registers (URBR1 and URBR2, URBR3 and URBR4)

These registers contain the data received from the transmitter on the UART buses. In FIFO mode, when read, they return the first byte received. For FIFO status information, refer to the UDSR[RXRDY] description.

Except for the case when there is an overrun, URBR returns the data in the order it was received from the transmitter. Refer to the ULSR[OE] description, [Section 16.3.1.10, “Line Status Registers \(ULSR1 and ULSR2, ULSR3 and ULSR4\).”](#) Figure 16-2 shows the receiver buffer registers. Note that these registers have same offset as the UTHR.

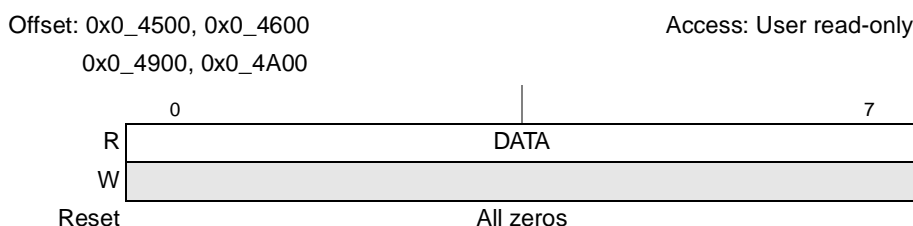


Figure 16-2. Receiver Buffer Registers (URBR1 and URBR2)

Table 16-4 describes URBR.

Table 16-4. URBR Field Descriptions

Bits	Name	Description
0–7	DATA	Data received from the transmitter on the UART bus [read only]

16.3.1.2 Transmitter Holding Registers (UTHR1 and UTHR2, UTHR3 and UTHR4)

A write to these 8-bit registers causes the UART devices to transfer 5 to 8 data bits on the UART bus in the format set up in the ULCR (line control register). In FIFO mode, data written to UTHR is placed into the FIFO. The data written to UTHR is the data sent onto the UART bus, and the first byte written to UTHR is the first byte onto the bus. UDSR[TXRDY] indicates when the FIFO is full. Refer to [Table 16-21](#) and [Table 16-22](#).

Figure 16-3 shows the bits in the UTHR.

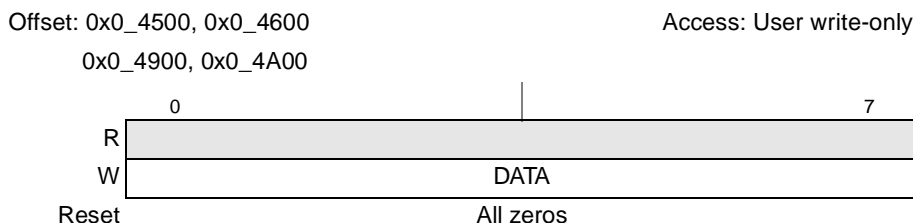


Figure 16-3. Transmitter Holding Registers (UTHR1 and UTHR2, UTHR3 and UTHR4)

Table 16-5 describes the UTHR.

Table 16-5. UTHR Field Descriptions

Bits	Name	Description
0–7	DATA	Data that is written to UTHR [Write only]

16.3.1.3 Divisor Most and Least Significant Byte Registers (UDMB and UDLB)

UDLB is concatenated with the divisor most significant byte register (UDMB) to create the divisor used to divide the input clock into the DUART. The output frequency of the baud generator is 16 times the baud rate; therefore, the desired baud rate = platform clock frequency ÷ (16 × [UDMB||UDLB]). Equivalently, [UDMB||UDLB:0b0000] = platform clock frequency/desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in Table 16-8.

Figure 16-4 shows the bits in the UDMBs.

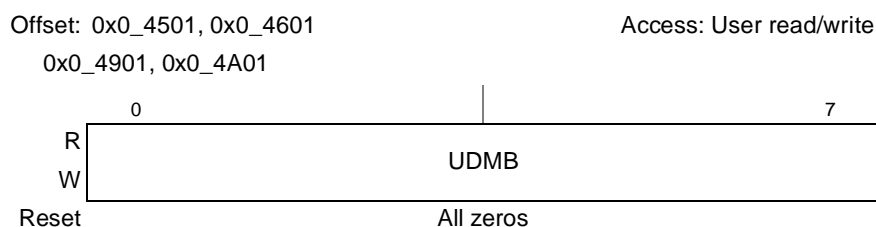


Figure 16-4. Divisor Most Significant Byte Registers (UDMB1 and UDMB2, UDMB3 and UDMB4)

Table 16-6 describes the UDMB.

Table 16-6. UDMB Field Descriptions

Bits	Name	Description
0–7	UDMB	Divisor most significant byte

Figure 16-5 shows the bits in the UDLBs.

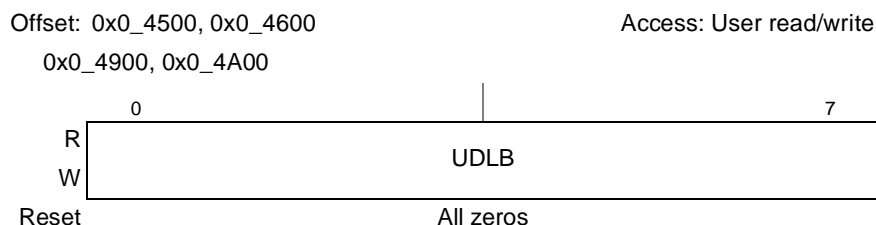


Figure 16-5. Divisor Least Significant Byte Registers (UDLB1 and UDLB2, UDLB3 and UDLB4)

Table 16-7 describes the UDLB.

Table 16-7. UDLB Field Descriptions

Bits	Name	Description
0–7	UDLB	Divisor least significant byte. This is concatenated with UDMB.

Table 16-8 shows baud rate for a variety of input clock frequencies.

Table 16-8. Baud Rate Examples

Baud Rate (Decimal)	Divisor		Input Clock (System Clock) Frequency (MHz)	Percent Error (Decimal)
	Decimal	Hex		
9,600	866	362	133	0.013
19,200	433	1B1	133	0.013
38,400	216	D8	133	0.218
56,000	148	94	133	0.300
128,000	65	41	133	0.090
256,000	32	20	133	1.471

To get the percent error value, the following three steps are taken:

1. The input clock frequency (ICF) is divided by the actual frequency input (AFI) to get the correct divisor value (ICF/AFI, where AFI = baud rate × 16 × divisor).
2. The divisor value is subtracted from 1.
3. The result from the step two is multiplied by 100 to calculate the final percent error. The result is calculated in absolute value (no negative numbers).

These steps can be described with the following equation:

$$\text{Percent error value} = (1 - \text{AFI/ICF}) \times 100$$

16.3.1.4 Interrupt Enable Registers (UIER1 and UIER2, UIER3 and UIER4)

The UIER gives the user the ability to mask specific UART interrupts to the programmable interrupt controller (PIC).

Figure 16-6 shows the bits in the UIER.

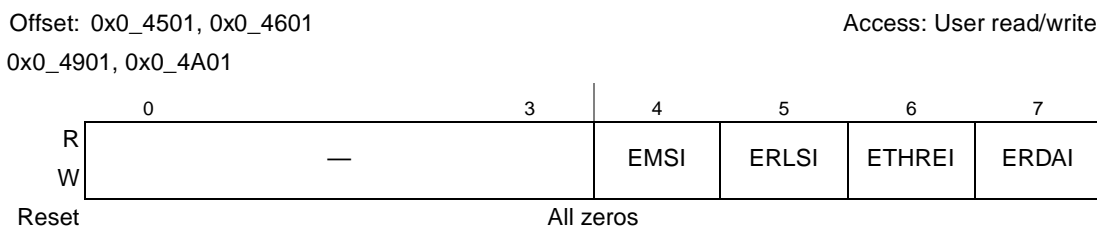


Figure 16-6. Interrupt Enable Registers (UIER1 and UIER2, UIER3 and UIER4)

Table 16-9 describes the UIER fields.

Table 16-9. UIER Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4	EMSI	Enable MODEM status interrupt 0 Mask interrupts caused by UMSR[DCTS] being set. 1 Enable and assert interrupts when UMSR[CTS] changes state.
5	ERLSI	Enable receiver line status interrupt 0 Mask interrupts when ULSR's overrun, parity error, framing error, or break interrupt bits are set. 1 Enable and assert interrupts when ULSR's overrun, parity error, framing error or break interrupt bits are set.
6	ETHREI	Enable transmitter holding register empty interrupt 0 Mask interrupt when ULSR[THRE] is set. 1 Enable and assert interrupts when ULSR[THRE] is set.
7	ERDAI	Enable received data available interrupt 0 Mask interrupt when new receive data is available or receive data time-out has occurred. 1 Enable and assert interrupts when a new data character is received from the external device and/or a time-out interrupt occurs in FIFO mode.

16.3.1.5 Interrupt ID Registers (UIIR1 and UIIR2, UIIR3 and UIIR4)

The UIIRs indicate when an interrupt is pending from the corresponding UART and what type of interrupt is active. They also indicate if the FIFOs are enabled.

The DUART prioritizes interrupts into four levels and records these in the corresponding UIIR. The four levels of interrupt conditions in order of priority are as follows:

1. Receiver line status
2. Received data ready/character time-out
3. Transmitter holding register empty
4. MODEM status

See Table 16-11 for more details.

When the UIIR is read, the associated DUART serial channel freezes all interrupts and indicates the highest priority pending interrupt. While this read transaction is occurring, the associated DUART serial channel records new interrupts, but does not change the contents of UIIR until the read access is complete.

Figure 16-7 shows the bits in the UIIR.

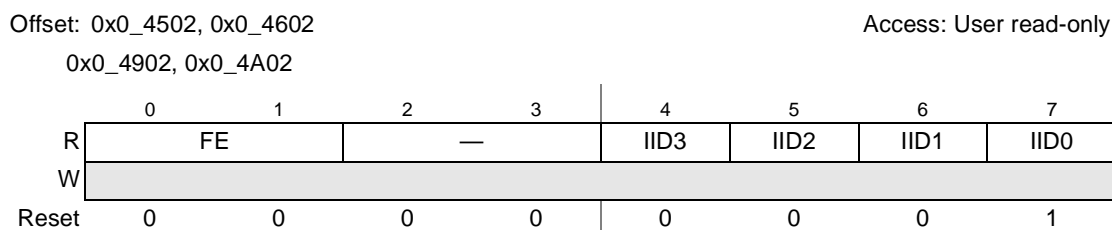


Figure 16-7. Interrupt ID Registers (UIIR1 and UIIR2, UIIR3 and UIIR4)

Table 16-10 describes the fields of the UIIR.

Table 16-10. UIIR Field Descriptions

Bits	Name	Description
0–1	FE	FIFOs enabled. Reflects the setting of UFCR[FEN].
2–3	—	Reserved
4	IID3	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in Table 16-11. IID3 is set along with IID2 only when a time out interrupt is pending for FIFO mode.
5–6	IID2–IID1	Interrupt ID bits identify the highest priority pending interrupt as indicated in Table 16-11.
7	IID0	IID0 indicates when an interrupt is pending. 0 The UART has an active interrupt ready to be serviced. 1 No interrupt is pending.

The bits contained in the UIIR registers are described in Table 16-11.

Table 16-11. UIIR IID Bits Summary

IID3–IID0	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
0001	—	—	—	—
0110	Highest	Receiver line status	Overrun error, parity error, framing error, or break interrupt	Reading the line status register
0100	Second	Received data available	Receiver data available or trigger level reached in FIFO mode.	Reading the receiver buffer register or if the number of bytes in the receiver FIFO drops below the trigger level.
1100	Second	Character time-out	No characters were removed from or input to the receiver FIFO during the last four character times and at least one character is in the receiver FIFO.	Reading the receiver buffer register
0010	Third	UTHR empty	Transmitter holding register is empty.	Reading UIIR or writing to UTHR
0000	Fourth	MODEM status	$\overline{\text{CTS}}$ input value changed since last read of UMSR.	Reading UMSR

16.3.1.6 FIFO Control Registers (UFCR1 and UFCR2, UFCR3 and UFCR4)

UFCR is used to enable and clear the receiver and transmitter FIFOs, set a receiver FIFO trigger level to control the received data available interrupt, and select the type of DMA signaling.

UFCR bits cannot be programmed unless FIFO enable bits are set. When changing from FIFO mode to 16450 mode (non-FIFO mode) and vice versa, data is automatically cleared from the FIFOs.

After all of the bytes in the receiver FIFO are cleared, the receiver internal shift register is not cleared. Similarly, the bytes are cleared in the transmitter FIFO, but the transmitter internal shift register is not cleared. Both TFR and RFR are self clearing.

Figure 16-8 shows the bits in the UFCRs.

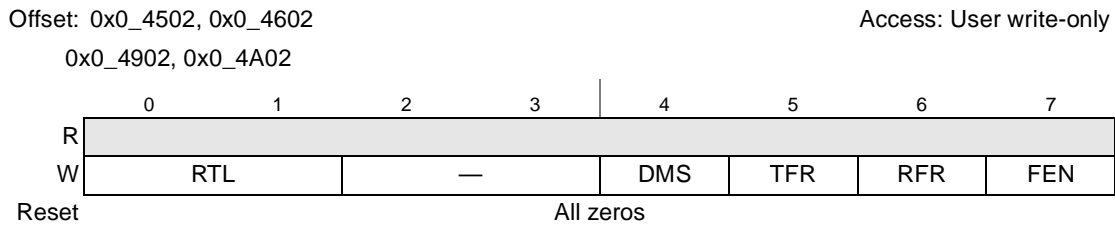


Figure 16-8. FIFO Control Registers (UFCR1 and UFCR2, UFCR3 and UFCR4)

Table 16-12 describes the fields of the UFCRs.

Table 16-12. UFCR Field Descriptions

Bits	Name	Description
0–1	RTL	Receiver trigger level. A received data available interrupt occurs when UIER[ERDAI] is set and the number of bytes in the receiver FIFO equals RTL value. 00 1 byte 01 4 bytes 10 8 bytes 11 14 bytes
2–3	—	Reserved
4	DMS	DMA mode select. See Section 16.4.5.2, “DMA Mode Select” 0 UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 0. 1 UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 1 if UFCR[FEN] = 1.
5	TFR	Transmitter FIFO reset 0 No action 1 Clears all bytes in the transmitter FIFO and resets the FIFO counter/pointer to 0
6	RFR	Receiver FIFO reset 0 No action 1 Clears all bytes in the receiver FIFO and resets the FIFO counter/pointer to 0
7	FEN	FIFO enable 0 FIFOs are disabled and cleared 1 Transmitter and receiver FIFOs are enabled.

16.3.1.7 Alternate Function Registers (UAFR1 and UAFR2, UAFR3 and UAFR4)

The UAFRs, shown in [Figure 16-9](#), allow software to write to both UART1 and UART2 registers, and UART3 and UART4 registers simultaneously with the same write operation. The UAFRs also provide a means for the device's performance monitor to track the baud clock.

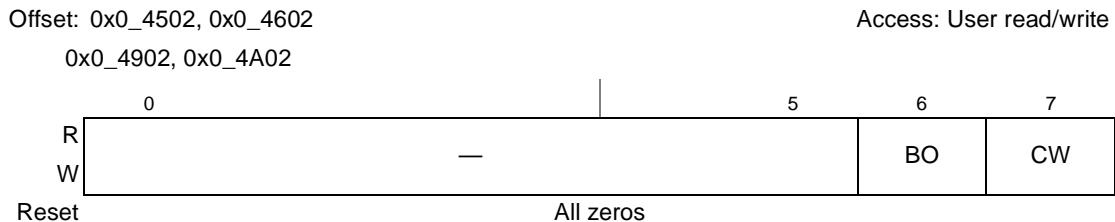


Figure 16-9. Alternate Function Register (UAFR)

Table 16-13 describes UAFR fields.

Table 16-13. UAFR Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	BO	Baud clock select 0 The baud clock is not gated off. 1 The baud clock is gated off.
7	CW	Concurrent write enable 0 Disables writing to both UART1 and UART2. 1 Enables concurrent writes to corresponding UART registers. A write to a register in UART1 is also a write to the corresponding register in UART2 and vice versa.

16.3.1.8 Line Control Registers (ULCR1 and ULCR2, ULCR3 and ULCR4)

The ULCRs specify the data format for the UART bus and set the divisor latch access bit ULCR[DLAB], which controls the ability to access the divisor latch least and most significant bit registers and the alternate function register.

After initializing ULCR, the software should not rewrite the ULCR while valid transfers on the UART bus are active. The software should not rewrite the ULCR until the last STOP bit is received and no new characters are being transferred on the bus.

The stick parity bit, ULCR[SP], assigns a set parity value for the parity bit time slot sent on the UART bus. The set value is defined as mark parity (logic 1) or space parity (logic 0). ULCR[PEN] and ULCR[EPS] help determine the set parity value. See Table 16-15. ULCR[NSTB] defines the number of STOP bits to be sent at the end of the data transfer. The receiver checks only the first STOP bit, regardless of the number of STOP bits selected. The word length select bits (1 and 0) define the number of data bits transmitted or received as a serial character. The word length does not include START, parity, and STOP bits.

Figure 16-10 shows the bits in the ULCRs.

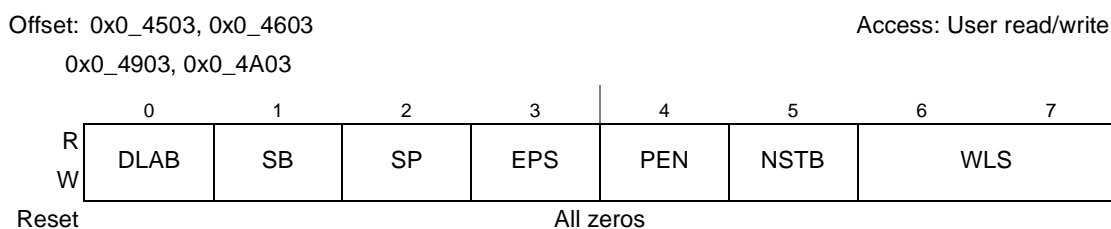


Figure 16-10. Line Control Register (ULCR1 and ULCR2, ULCR3 and ULCR4)

Table 16-14 describes the ULCR fields.

Table 16-14. ULCR Field Descriptions

Bits	Name	Description
0	DLAB	Divisor latch access bit 0 Access to all registers except UDLB, UAFR, and UDMB. 1 Ability to access UDMB, UDLB, and UAFR.
1	SB	Set break 0 Send normal UTHR data onto the SOUT signal. 1 Force logic 0 to be on SOUT. Data in the UTHR is not affected.
2	SP	Stick parity 0 Stick parity is disabled. 1 If PEN = 1 and EPS = 1, space parity is selected; if PEN = 1 and EPS = 0, mark parity is selected.
3	EPS	Even parity select. See Table 16-15 . 0 If PEN = 1 and SP = 0 then odd parity is selected. 1 If PEN = 1 and SP = 0 then even parity is selected.
4	PEN	Parity enable 0 No parity generation and checking. 1 Generate parity bit as a transmitter, and check parity as a receiver.
5	NTSB	Number of STOP bits 0 One STOP bit is generated in the transmitted data. 1 When a 5-bit data length is selected, 1 1/2 STOP bits are generated. When either a 6-, 7-, or 8-bit word length is selected, two STOP bits are generated.
6–7	WLS	Word length select. Number of bits that comprise the character length. 00 5 bits 01 6 bits 10 7 bits 11 8 bits

Table 16-15. Parity Selection Using ULCR[PEN], ULCR[SP], and ULCR[EPS]

PEN	SP	EPS	Parity Selected
0	0	0	No parity
0	0	1	No parity
0	1	0	No parity
0	1	1	No parity
1	0	0	Odd parity
1	0	1	Even parity
1	1	0	Mark parity
1	1	1	Space parity

16.3.1.9 MODEM Control Registers (UMCR1 and UMCR2, UMCR3 and UMCR4)

The UMCRs, shown in [Figure 16-11](#), control the interface with the external peripheral device on the UART bus.

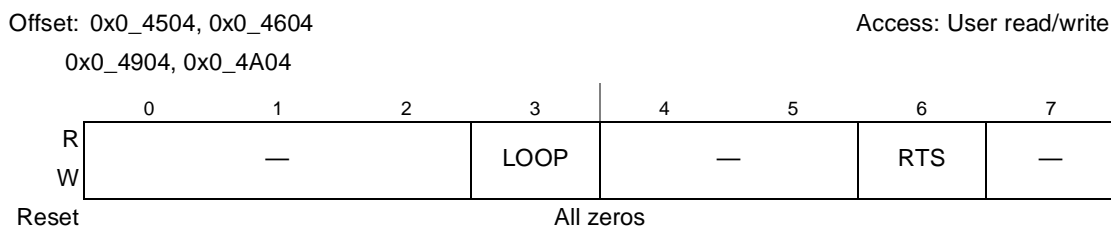


Figure 16-11. Modem Control Register (UMCR1 and UMCR2, UMCR3 and UMCR4)

[Table 16-16](#) describes the UMCR fields.

Table 16-16. UMCR Field Descriptions

Bits	Name	Description
0–2	—	Reserved, should be cleared
3	LOOP	Local loopback mode 0 Normal operation. 1 Functionally, the data written to UTHR can be read from URBR of the same UART, and UMCR[RTS] is tied to UMSR[CTS].
4–5	—	Reserved
6	RTS	Ready to send 0 Negates corresponding $\overline{\text{UART_RTS}}$ output. 1 Assert corresponding $\overline{\text{UART_RTS}}$ output. Informs external MODEM or peripheral that the UART is ready for sending/receiving data.
7	—	Reserved

16.3.1.10 Line Status Registers (ULSR1 and ULSR2, ULSR3 and ULSR4)

The ULSRs, shown in [Figure 16-12](#), monitor the status of the data transfer on the UART buses. To isolate the status bits from the proper character received through the UART bus, software should read the ULSR and then the URBR.

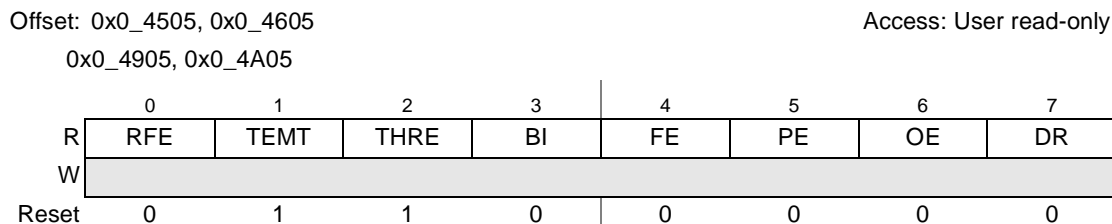


Figure 16-12. Line Status Register (ULSR1 and ULSR2, ULSR3 and ULSR4)

[Table 16-17](#) describes the ULSR fields.

Table 16-17. ULSR Field Descriptions

Bits	Name	Description
0	RFE	Receiver FIFO error. 0 Cleared when there are no errors in the receiver FIFO or on a read of the ULSR with no remaining receiver FIFO errors. 1 Set when one of the characters in the receiver FIFO encounters an error (framing, parity, or break interrupt).
1	TEMT	Transmitter empty 0 Either or both the UTHR or the internal transmitter shift register has a data character. In FIFO mode, a data character is in the transmitter FIFO or the internal transmitter shift register. 1 Both the UTHR and the internal transmitter shift register are empty. In FIFO mode, both the transmitter FIFO and the internal transmitter shift register are empty.
2	THRE	Transmitter holding register empty 0 UTHR is not empty. 1 A data character has transferred from the UTHR into the internal transmitter shift register. In FIFO mode, the transmitter FIFO contains no data character.
3	BI	Break interrupt 0 Cleared when the ULSR is read or when a valid data transfer is detected (that is, STOP bit is received). 1 Received data of logic 0 for more than START bit + Data bits + Parity bit + one STOP bits length of time. A new character is not loaded until SIN returns to the mark state (logic 1) and a valid START is detected. In FIFO mode, a zero character is encountered in the FIFO (the zero character is at the top of the FIFO). In FIFO mode, only one zero character is stored.
4	FE	Framing error 0 Cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register. 1 Invalid STOP bit for receive data (only the first STOP bit is checked). In FIFO mode, FE is set when the character that detected a framing error is encountered in the FIFO (that is the character at the top of the FIFO). An attempt to resynchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit, so it assumes this logic 0 sample is a true START bit and then will receive the following new data.

Table 16-17. ULSR Field Descriptions (continued)

Bits	Name	Description
5	PE	Parity error 0 Cleared when ULSR is read or when a new character is loaded into URBR. 1 Unexpected parity value encountered when receiving data. In FIFO mode, the character with the error is at the top of the FIFO.
6	OE	Overrun error 0 Cleared when ULSR is read 1 Before URBR was read, it was overwritten with a new character. The old character is lost. In FIFO mode, the receiver FIFO is full (regardless of the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. The old character was overwritten by the new character. Data in the receiver FIFO was not overwritten.
7	DR	Data ready 0 Cleared when URBR is read or when all of the data in the receiver FIFO is read. 1 A character was received in the URBR or the receiver FIFO.

16.3.1.11 MODEM Status Registers (UMSR1 and UMSR2, UMSR3 and UMSR4)

The UMSRs, shown in [Figure 16-13](#), track the status of the MODEM (or external peripheral device) \overline{CTS} , set for the corresponding UART.

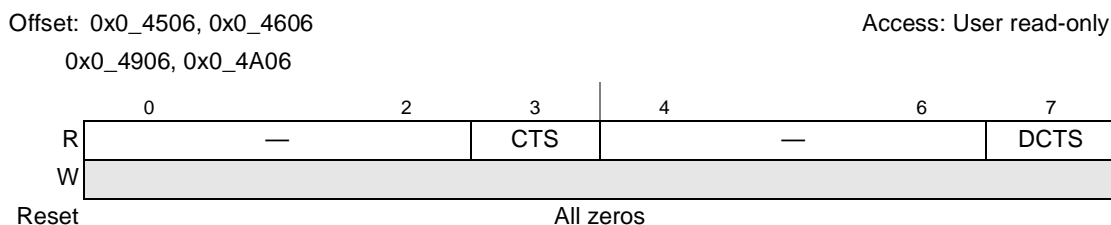


Figure 16-13. Modem Status Register (UMSR1 and UMSR2, UMSR3 and UMSR4)

[Table 16-18](#) describes UMSR fields.

Table 16-18. UMSR Field Descriptions

Bits	Name	Description
0–2	—	Reserved, should be cleared
3	CTS	Clear to send. Represents the inverted value of the \overline{CTS} input pin from the external peripheral device. 0 Corresponding \overline{CTS}_n is negated. 1 Corresponding \overline{CTS}_n is asserted. The MODEM or peripheral device is ready for data transfers.
4–6	—	Reserved, should be cleared
7	DCTS	Delta clear to send 0 No change on the corresponding \overline{CTS}_n signal since the last read of UMSR[CTS]. 1 \overline{CTS}_n changed since the last read of UMSR[CTS]. Causes an interrupt if UIER[EMSI] is set to detect this condition.

16.3.1.12 Scratch Registers (USCR1 and USCR2, USCR3 and USCR4)

USCR, shown in [Figure 16-14](#), are for debugging software or the DUART hardware. The USCRs do not affect the operation of the DUART.

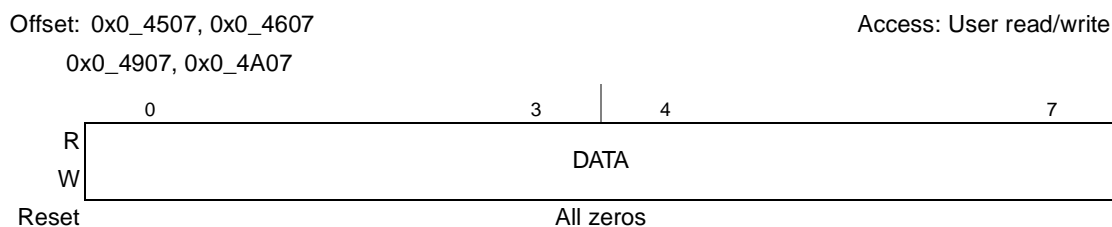


Figure 16-14. Scratch Register (USCR)

[Table 16-19](#) describes USCR fields.

Table 16-19. USCR Field Descriptions

Bits	Name	Description
0–7	DATA	Data

16.3.1.13 DMA Status Registers (UDSR1 and UDSR2, UDSR3 and UDSR4)

The DMA status registers (UDSRs), shown in [Figure 16-15](#), return transmitter and receiver FIFO status and provide the ability to assist DMA data operations to and from the FIFOs.

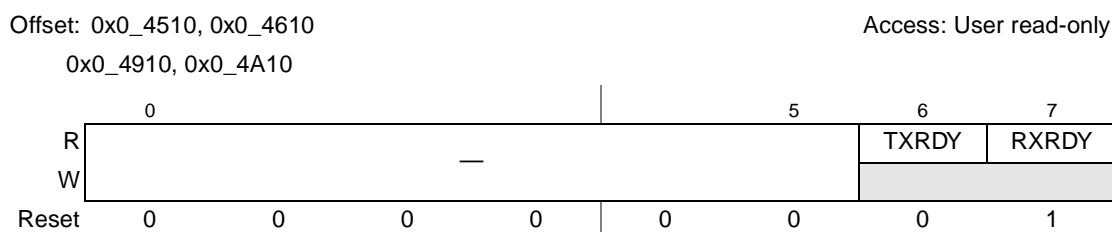


Figure 16-15. DMA Status Register (UDSR)

[Table 16-20](#) describes the fields of the UDSRs.

Table 16-20. UDSR Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	TXRDY	Transmitter ready. Reflects the status of the transmitter FIFO or the UTHR. The status depends on the DMA mode selected, which is determined by UFCR[DMS] and UFCR [FEN]. 0 The bit is cleared, as shown in Table 16-22 . 1 This bit is set, as shown in Table 16-21 .
7	RXRDY	Receiver ready. This read-only bit reflects the status of the receiver FIFO or URBR. The status depends on the DMA mode selected, which is determined by UFCR[DMS] and UFCR [FEN]. 0 The bit is cleared, as shown in Table 16-24 . 1 This bit is set, as shown in Table 16-23 .

Table 16-21. UDSR[TXRDY] Set Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	TXRDY is set after the first character is loaded into the transmitter FIFO or UTHR.
0	1	0	
1	0	0	
1	1	1	TXRDY is set when the transmitter FIFO is full.

Table 16-22. UDSR[TXRDY] Cleared Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	TXRDY is cleared when there are no characters in the transmitter FIFO or UTHR.
0	1	0	
1	0	0	
1	1	1	TXRDY is cleared when there are no characters in the transmitter FIFO or UTHR. TXRDY remains clear while the transmitter FIFO is not yet full.

Table 16-23. UDSR[RXRDY] Set Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	RXRDY is set when there are no characters in the receiver FIFO or URBR.
0	1	0	
1	0	0	
1	1	1	RXRDY is set when the trigger level has not been reached and there has been no time out.

Table 16-24. UDSR[RXRDY] Cleared

DMS	FEN	DMA Mode	Meaning
0	0	0	RXRDY is cleared when there is at least one character in the receiver FIFO or URBR.
0	1	0	
1	0	0	
1	1	1	RXRDY is cleared when the trigger level or a time-out has been reached. RXRDY remains cleared until the receiver FIFO is empty.

16.4 Functional Description

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the system clock signal.

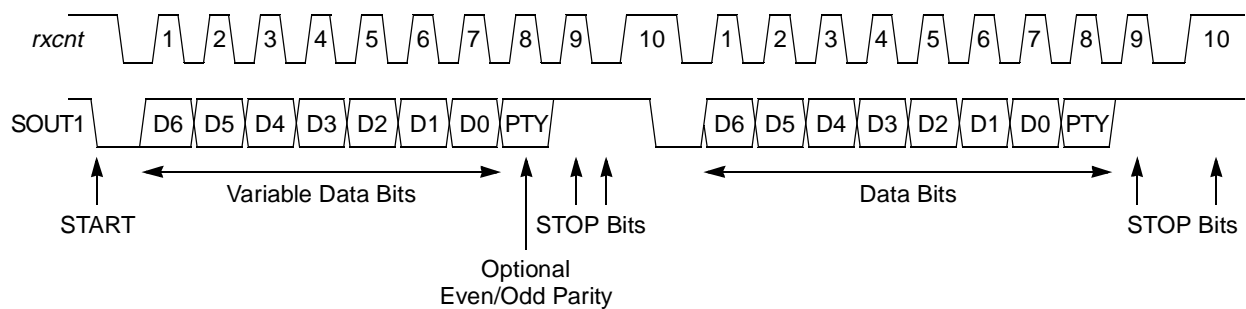
The transmitter accepts parallel data with a write access to UTHR. In FIFO mode, the data is placed directly into an internal transmitter shift register, or into the transmitter FIFO—see [Section 16.4.5, “FIFO Mode.”](#) The transmitting registers convert the data to a serial bit stream by inserting the appropriate

START, STOP, and optional parity bits. Finally, the registers output a composite serial data stream on the channel transmitter serial data output (SOUT). The transmitter status may be polled or interrupt driven.

The receiver accepts serial data on the channel receiver serial data input (SIN), converts the data into parallel format, and checks for START, STOP, and parity bits. In FIFO mode, the receiver removes the START, STOP, and parity bits and then transfers the assembled character from the receiver buffer, or receiver FIFO. This transfer occurs in response to a read of the UART receiver buffer register (URBR). The receiver status may be polled or interrupt-driven.

16.4.1 Serial Interface

The UART bus is a serial, full-duplex, point-to-point bus as shown in [Figure 16-16](#). Therefore, only two devices are attached to the same signals and there is no need for address or arbitration bus cycles.



Two 7-Bit Data Transmissions with Parity and 2-Bit STOP Transactions

Figure 16-16. UART Bus Interface Transaction Protocol Example

A standard UART bus transfer is composed of either three or four parts:

- START bit
- Data transfer (least significant bit is first data bit on the bus)
- Parity bit (optional)
- STOP bits

An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to drive the bits on SOUT.

The following sections describe the four components of the serial interface, the baud-rate generator, local loopback mode, different errors, and FIFO mode.

16.4.1.1 START Bit

A write to UTHR generates a START bit on the SOUT signal. [Figure 16-16](#) shows that the START bit is defined as a logic 0. The START bit denotes the beginning of a new data transfer which is limited to the bit length programmed in ULCR. When the bus is idle, SOUT is high.

16.4.1.2 Data Transfer

Each data transfer contains 5, 6, 7, or 8 bits of data. The ULCR data bit length for the transmitter and receiver UART devices must agree before a transfer begins; otherwise, a parity or framing error may occur. A transfer begins when UTHR is written. At that time, a START bit is generated followed by 5 to 8 of the data bits previously written to the UTHR. The data bits are driven from the least- to the most-significant bits. After the parity and STOP bits, a new data transfer can begin if new data is written to UTHR.

16.4.1.3 Parity Bit

The user has the option of using even, odd, no parity, or stick parity (see [Section 16.3.1.8, “Line Control Registers \(ULCR1 and ULCR2, ULCR3 and ULCR4\)”](#)). Both the receiver and transmitter parity definitions must agree before transferring data. When receiving data, a parity error can occur if an unexpected parity value is detected (see [Section 16.3.1.10, “Line Status Registers \(ULSR1 and ULSR2, ULSR3 and ULSR4\)”](#)).

16.4.1.4 STOP Bit

The transmitter device ends the write transfer by generating a STOP bit. The STOP bit is always high. The user can program the length of the STOP bit(s) in the ULCR. Both the receiver and transmitter STOP bit length must agree before attempting to transfer data. A framing error can occur if an invalid STOP bit is detected.

16.4.2 Baud-Rate Generator Logic

Each UART contains an independent programmable baud-rate generator, that is capable of taking the system clock input and dividing the input by any divisor from 1 to $2^{16} - 1$.

5. The baud rate is defined as the number of bits per second that can be sent over the UART bus. The formula for calculating baud rate is as follows:

$$\text{Baud rate} = (1/16) \times (\text{system clock frequency/divisor value})$$

Therefore, the output frequency of the baud-rate generator is 16 times the baud rate.

1. The divisor value is determined by the following two 8-bit registers to form a 16-bit binary number:
 - UART divisor most significant byte register (UDMB)
 - UART divisor least significant byte register (UDLB)

Upon loading either of the divisor latches, a 16-bit baud-rate counter is loaded.

The divisor latches must be loaded during initialization to ensure proper operation of the baud-rate generator. Both UART devices on the same bus must be programmed for the same baud rate before starting a transfer.

The baud clock can be passed to the performance monitor by enabling UAFR[BO]. This can be used to determine baud-rate errors.

16.4.3 Local Loopback Mode

Local loopback mode is provided for diagnostic testing. The data written to UTHR can be read from the receiver buffer register (URBR) of the same UART. In this mode, the MODEM control register UMCR[RTS] is internally tied to the MODEM status register UMSR[CTS]. The transmitter SOUT is set to a logic 1 and the receiver SIN is disconnected. The output of the transmitter shift register is looped back into the receiver shift register input. The $\overline{\text{CTS}}$ (input signal) is disconnected, $\overline{\text{RTS}}$ is internally connected to $\overline{\text{CTS}}$, and the $\overline{\text{RTS}}$ (output signal) becomes inactive. In this diagnostic mode, data that is transmitted is immediately received. In local loopback mode the transmit and receive data paths of the DUART can be verified. Note that in local loopback mode, the transmit/receive interrupts are fully operational and can be controlled by the interrupt enable register (UIER).

16.4.4 Errors

The following sections describe framing, parity, and overrun errors which may occur while data is transferred on the UART bus. Each of the error bits are usually cleared, as described below, when the line status register (ULSR) is read.

16.4.4.1 Framing Error

When an invalid STOP bit is detected, a framing error occurs and ULSR[FE] is set. Note that only the first STOP bit is checked. In FIFO mode, ULSR[FE] is set when the character at the top of the FIFO detects a framing error. An attempt to re-synchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit. ULSR[FE] is cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register.

16.4.4.2 Parity Error

When unexpected parity values are encountered while receiving data, a parity error occurs and ULSR[PE] is set. In FIFO mode, ULSR[PE] is set when the character with the error is at the top of the FIFO. ULSR[PE] is cleared when ULSR is read or when a new character is loaded into the URBR.

16.4.4.3 Overrun Error

When a new (overwriting character) STOP bit is detected and the old character is lost, an overrun error occurs and ULSR[OE] is set. In FIFO mode, ULSR[OE] is set after the receiver FIFO is full (despite the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. Data in the FIFO is not overwritten; only the shift register data is overwritten. Therefore, the interrupt occurs immediately. ULSR[OE] is cleared when ULSR is read.

16.4.5 FIFO Mode

The UARTs use an alternate mode (FIFO mode) to relieve the processor core from excessive software overhead. The FIFO control register (UFCCR) is used to enable and clear the receiver and transmitter FIFOs

and set the FIFO receiver trigger level `UFCCR[RTL]` to control the received data available interrupt `UIER[ERDAI]`.

The `UFCCR` also selects the type of DMA signaling. The `UDSR[RXRDY]` indicates the status of the receiver FIFO. `UDSR[TXRDY]` indicate when the transmitter FIFO is full. When in FIFO mode, data written to `UTHR` is placed into the transmitter FIFO. The first byte written to `UTHR` is the first byte onto the UART bus.

16.4.5.1 FIFO Interrupts

In FIFO mode, the `UIER[ERDAI]` is set when a time-out interrupt occurs. A receive data time-out generates a maskable interrupt condition (through `UIER[ERDAI]`). See [Section 16.3.1.4, “Interrupt Enable Registers \(UIER1 and UIER2, UIER3 and UIER4\)”](#).

`UIIR` indicates whether the FIFOs are enabled. `UIIR[IID3]` is set only for FIFO mode interrupts. The character time-out interrupt occurs when no characters have been removed from or input to the receiver FIFO during the last four character times and at least one character is in the receiver FIFO. The character time-out interrupt (controlled by `UIIR[IIDn]`) is cleared when `URBR` is read. See [Section 16.3.1.5, “Interrupt ID Registers \(UIIR1 and UIIR2, UIIR3 and UIIR4\)”](#).

`UIIR[FE]` indicates whether FIFO mode is enabled.

16.4.5.2 DMA Mode Select

`UDSR[RXRDY]` reflects the status of the receiver FIFO or `URBR`. In mode 0 (`UFCCR[DMS]` is cleared), `UDSR[RXRDY]` is cleared when at least one character is in the receiver FIFO or `URBR`; it is set when there are no more characters in the receiver FIFO or `URBR`. This occurs regardless of the `UFCCR[FEN]` setting. In mode 1 (`UFCCR[DMS]` and `UFCCR[FEN]` are set), `UDSR[RXRDY]` is cleared when the trigger level or a time-out has been reached; it is set when there are no more characters in the receiver FIFO.

`UDSR[TXRDY]` reflects the status of the transmitter FIFO or `UTHR`. In mode 0 (`UFCCR[DMS]` is cleared), `UDSR[TXRDY]` is cleared when there are no characters in the transmitter FIFO or `UTHR`; it is set after the first character is loaded into the transmitter FIFO or `UTHR`. This occurs regardless of the `UFCCR[FEN]` setting. In mode 1 (`UFCCR[DMS]` and `UFCCR[FEN]` are set), `UDSR[TXRDY]` is cleared when there are no characters in the transmitter FIFO or `UTHR`; it is set when the transmitter FIFO is full.

See [Section 16.3.1.13, “DMA Status Registers \(USDR1 and USDR2, USDR3 and USDR4\)”](#) for a complete description of the `USDR[RXRDY]` and `USDR[TXRDY]` bits.

16.4.5.3 Interrupt Control Logic

An interrupt is active when DUART interrupt ID register bit 7 (`UIIR[IID0]`), is cleared. `UIER` is used to mask specific interrupt types. See [Section 16.3.1.4, “Interrupt Enable Registers \(UIER1 and UIER2, UIER3 and UIER4\)”](#).

When the interrupts are disabled in `UIER`, polling software can not use `UIIR[IID0]` to determine whether the UART is ready for service. Software must monitor the appropriate `ULSR` and `UMSR` bits. `UIIR[IID0]` can be used for polling if the interrupts are enabled in `UIER`.

16.5 DUART Initialization/Application Information

The following requirements must be met for DUART accesses:

- All DUART registers must be mapped to a cache-inhibited and guarded area. (That is, the WIMG setting in the MMU needs to be 0b01x1.)
- All DUART registers are 1 byte wide. Reads and writes to these registers must be byte-length operations.

A system reset puts the DUART registers to a default state. Before the interface can transfer serial data, the following initialization steps are recommended:

1. Update the programmable interrupt controller (PIC) DUART channel interrupt vector source registers.
2. Set data attributes and control bits in the ULCR, UFCR, UAFR, UMCR, UDLB, and UDMB.
3. Set the data attributes and control bits of the external MODEM or peripheral device.
4. Set the interrupt enable register (UIER).
5. To start a write transfer, write to the UTHR.
6. Poll UIIR if the interrupts generated by the DUART are masked.



Chapter 17

JTAG/Testing Support

17.1 JTAG Overview

The device provides a JTAG (Joint Test Action Group) interface to facilitate boundary-scan testing. The JTAG interface complies to the IEEE 1149.1 boundary-scan specification. For additional information about JTAG operations, refer to the IEEE 1149.1 specification.

The JTAG interface consists of a set of five signals, three JTAG registers (see [Section 17.3, “JTAG Registers and Scan Chains,”](#)) and a test access port (TAP) controller, described in the following sections. A block diagram of the JTAG interface is shown in [Figure 17-1](#).

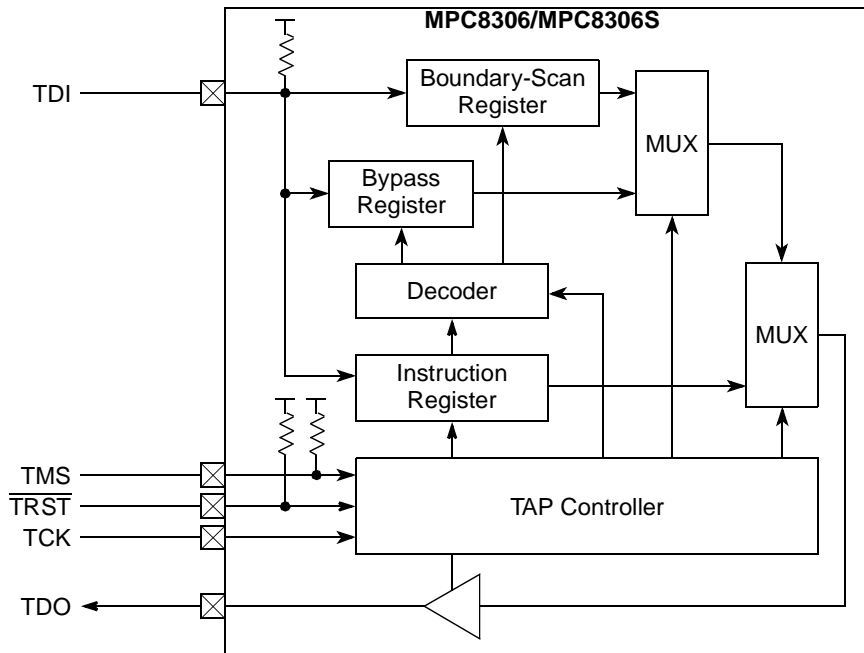


Figure 17-1. JTAG Interface Block Diagram

17.2 JTAG Signals

The device provides the following five dedicated JTAG signals:

- Test data input (TDI)
- Test data output (TDO)
- Test mode select (TMS)

- Test reset ($\overline{\text{TRST}}$)
- Test clock (TCK)

The TDI and TDO signals input and output all instructions and data to the JTAG scan registers. JTAG operations are controlled by the TAP controller through the TMS and TCK signals. Boundary-scan data is latched by the TAP controller on the rising edge of the TCK signal. The $\overline{\text{TRST}}$ signal is specified as optional by the IEEE 1149.1 specification, and is used to reset the TAP controller asynchronously. The assertion of the $\overline{\text{TRST}}$ signal at power-on reset ensures that the JTAG logic does not interfere with the normal operation of the device.

17.2.1 JTAG External Signal Descriptions

The JTAG signals are summarized in [Table 17-1](#).

Table 17-1. JTAG Test Signals Summary

Name	Description	Functional Block	Function	Reset Value	I/O
TCK	Test clock	Debug	Clock for JTAG testing.	—	I
TDI	Test data input		Serial input for instructions and data to the JTAG test subsystem. Internally pulled up.	—	I
TDO	Test data output		Serial data output for the JTAG test subsystem. High impedance except when scanning out data.	High impedance	O
TMS	Test mode select		Carries commands to the TAP controller for boundary scan operations. Internally pulled up.	—	I
$\overline{\text{TRST}}$	Test reset		Resets the TAP controller asynchronously. Internally pulled up.	—	I

[Table 17-2](#) shows detailed descriptions of the JTAG test signals.

Table 17-2. JTAG Test—Detailed Signal Descriptions

Signal	I/O	Description	
TCK	I	JTAG test clock.	
		State Meaning	Asserted/Negated—Should be driven by a free-running clock signal with a 30–70% duty cycle. Input signals to the TAP are clocked in on the rising edge. Changes to the TAP output signals occur on the falling edge. The test logic allows TCK to be stopped.
		Timing	See IEEE 1149.1 specification for more details.
TDI	I	JTAG test data input.	
		State Meaning	Asserted/Negated—The value present on the rising edge of TCK is clocked into the selected JTAG test instruction or data register. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 specification for more details.

Table 17-2. JTAG Test—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
TDO	O	JTAG test data output.	
		State Meaning	Asserted/Negated—The contents of the selected internal instruction or data register are shifted out on this signal on the falling edge of TCK. Remains in a high-impedance state except when scanning data.
		Timing	See IEEE 1149.1 specification for more details.
TMS	I	JTAG test mode select.	
		State Meaning	Asserted/Negated—Decoded by the internal JTAG TAP controller to distinguish the primary operation of the test support circuitry. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 specification for more details.
$\overline{\text{TRST}}$	I	JTAG test reset.	
		State Meaning	Asserted—Causes asynchronous initialization of the internal JTAG TAP controller. Must be asserted during power-on reset in order to properly initialize the JTAG TAP and for normal operation of the device. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor. Negated— Normal operation.
		Timing	See IEEE 1149.1 specification for more details.

17.3 JTAG Registers and Scan Chains

The bypass, boundary-scan, and instruction JTAG registers and their associated scan chains are mandatory for compliance with the IEEE 1149.1 specification.

- Bypass register. The bypass register is a single-stage register used to bypass the boundary-scan latches of the device during board-level boundary-scan operations involving components other than the device. The use of the bypass register reduces the total scan string size of the boundary-scan test.
- Boundary-scan registers. The JTAG interface provides a chain of registers dedicated to boundary-scan operations. To be JTAG-compliant, these registers cannot be shared with any functional registers of the device. The boundary-scan register chain includes registers controlling the direction of the input/output drivers, in addition to the registers reflecting the signal value received or driven.

The boundary-scan registers capture the input or output state of the device's signals during a Capture_DR TAP controller state. When a data scan is initiated following the Capture_DR state, the sampled values are shifted out through the TDO output while new boundary-scan register values are shifted in through the TDI input. At the end of the data scan operation, the boundary-scan registers are updated with the new values during an update_DR TAP controller state.

- Instruction register. The 8-bit JTAG instruction register serves as an instruction and status register. As TAP controller instructions are scanned in through the TDI input, the TAP controller status bits are scanned out through the TDO output.

- TAP controller. The device provides a standard JTAG TAP controller that controls instruction and data scan operations. The TMS signal controls the state transitions of the TAP controller.

Chapter 18

Delay Lock Loop (DLL)

This chapter describes the theory of operation of the delay lock loop (DLL) module in the integrated device. Additionally, the configuration, control, and status registers are described. Note that other chapters in this book describe additional specific initialization aspects for individual blocks.

18.1 DLL Introduction

The DLL unit consists of a phase detection circuit, adjustable delay unit, register file and control logic. A high-level block diagram of these elements is shown in [Figure 18-1](#).

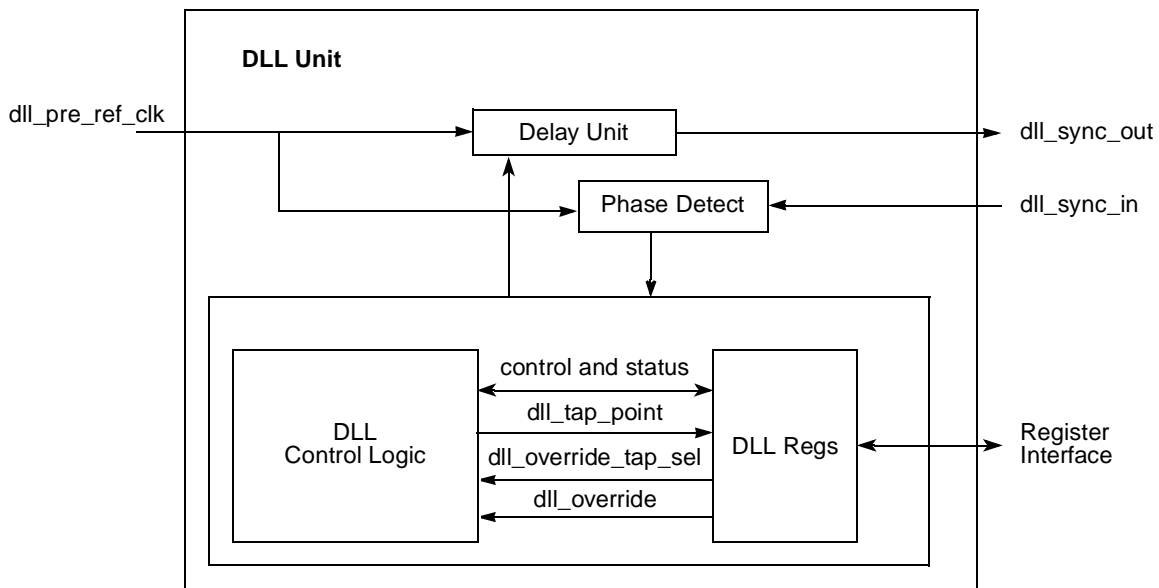


Figure 18-1. DLL Block Diagram

18.2 DLL Overview

The purpose of the DLL macro is to shift a reference clock to create an output clock to be used by external memory or I/O devices. The output clock is created by propagating the reference clock through a delay chain. A phase detect circuit compares the phase of the reference clock with that of a feedback clock. An output TAP point of the delay chain is chosen for the output clock such that the feedback clock is phase aligned with the reference clock. The DLL macro also contains override, debug, and error features.

DLL is used for LCLK (local bus clocks) skew elimination. Therefore, DLL_SYNC_OUT/DLL_SYNC_IN should be interpreted as LSYNC_OUT/LSYNC_IN, and DLL_CLK_OUT[0:n] should be interpreted as LCLK[0:n].

18.2.1 DLL Features

The DLL features are as follows:

- LOCK status reports when phase lock search is completed.
- WRAP status reports when the DLL wraps beyond either end of its delay chain during the search process. If WRAP is set while LOCK is set, this indicates a failure to find a lock during the search process.
- OVERRIDE mode allows a specific tap point to be selected.
- Current tap point is continuously reported.

18.2.2 DLL Modes of Operation

The DLL modes of operation are as follows:

- Normal mode: The DLL can operate in normal mode, in which the phase detector circuitry dynamically adjusts the tap point of the delay line.
- Override mode: The DLL can operate in override mode in which the tap point is specified by an input vector.

18.2.3 DLL External Signals

Table 18-1 displays the signals of the DLL macro.

Table 18-1. DLL Macro External Signals

Signal	I/O	Signal Description
DLL_SYNC_OUT	O	DLL output clock. Phase aligned with other clock outputs (DLL_CLK_OUT[0:n]).
DLL_SYNC_IN	O	DLL feedback clock, phased aligned with the internal logic clock.
DLL_CLK_OUT[0:n]	O	DLL output clocks to be used by external on-board devices.

18.3 DLL Initialization and Application Information

An example application using the DLL macro is shown in Figure 18-2. As this example shows, a memory or I/O controller generates the reference clock based on an internal knowledge of the ratio between the external clock and internal clk. DLL_CLK_OUT goes to the external memory or I/O devices, and the DLL_SYNC_OUT must be connected back to the DLL_SYNC_IN while keeping the same trace length as of DLL_CLK_OUT path. The DLL macro uses the phase detection and the delay chain to align the internal logic clock with the external device input clock, such that the integrated device's flip-flops and the external device's flip-flops are fed by clocks with minimal skew between them.

DLL lock status is used by the memory controller to hold off memory access until DLL search has been completed. However, if DLL override mode is used, DLL lock has no meaning.

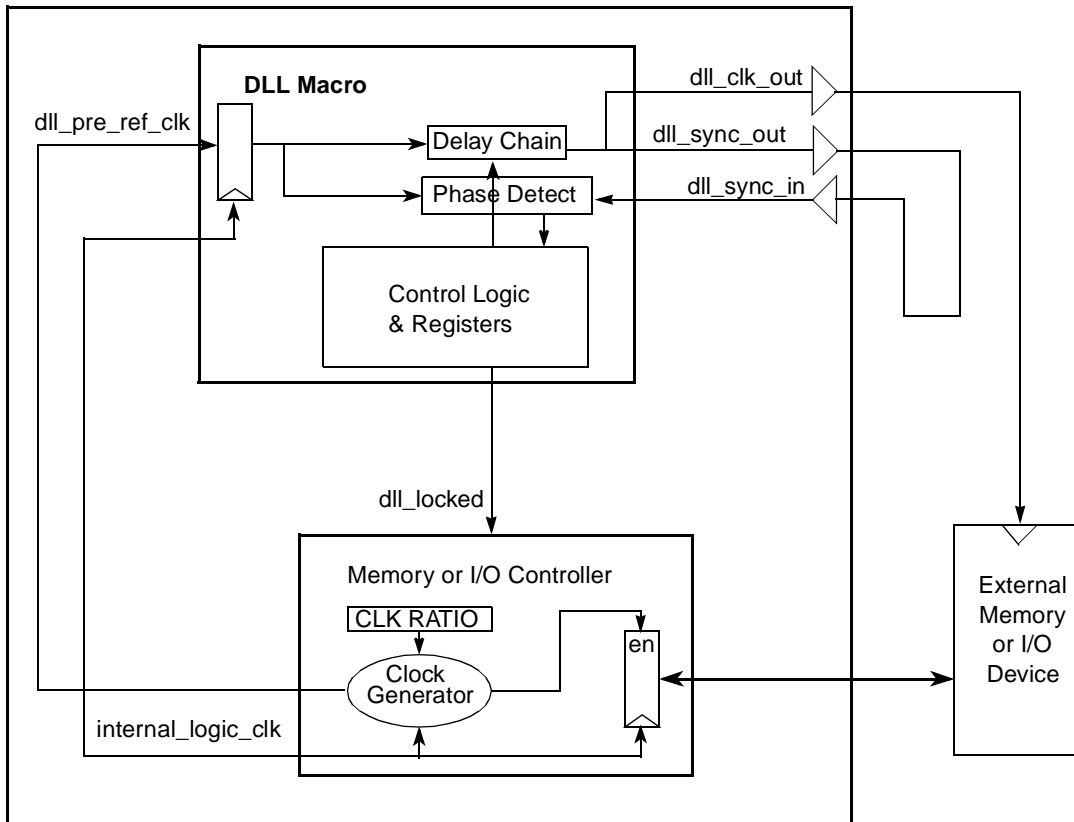


Figure 18-2. DLL Application Example

Use the following sequence to ensure DLL lock before starting to access the local bus devices:

1. Write `LCRR[DBYP] = 0` (DLL is enabled),
2. Sync,
3. Wait 1 μ s,
4. Poll the `DLLSR[LOCK]` bit until it becomes 1,
5. Access can now be gained to the local bus devices.

18.4 DLL Memory Map/Register Definition

The DLL programmable register map occupies 20 bytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All DLL registers are 32 bits wide located on 32-bit address boundaries. All addresses used in this chapter are offsets from the DLL starting address as defined in [Chapter 2, “Memory Map.”](#)

Table 18-2 shows the DLL registers.

Table 18-2. DLL Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x0_1100–0x0_1104	Reserved	—	—	—
0x08	DLL override register (DLLOVR)	R/W	All zeros	18.4.1/18-4
0x0C	DLL status register (DLLSR)	R	All zeros	18.4.2/18-5
0x10	DLL clock register (DLLCK)	R/W	0xFC00_0000	18.4.3/18-5
0x14–0xFF	Reserved	—	—	—

18.4.1 DLL Override Register (DLLOVR)

The DLL override register (DLLOVR), shown in [Figure 18-3](#), controls the override operation and the override value (coarse or fine) to force on the DLL in override mode.

Addr: 0x0_1108

Access: Read/Write

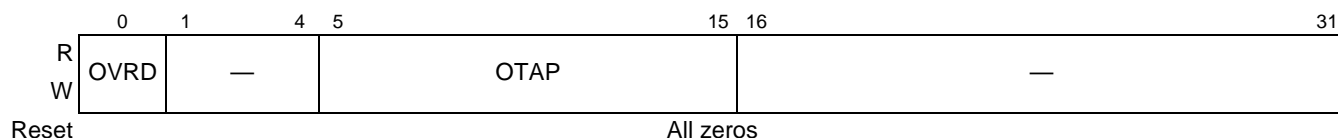


Figure 18-3. DLL Override Register

DLLOVR can be used to force a fixed delay value. It is recommended to wait until DLL lock is achieved, read the measured coarse and TAP delay values from the DLL status register, and then set the override delay values accordingly. [Table 18-3](#) describes DLLOVR fields.

Table 18-3. DLLOVR Field Description

Bits	Name	Description
0	OVRD	DLL override mode 0 Normal mode 1 Override mode
1–4	—	Reserved
5–15	OTAP	DLL override coarse and TAP select OTAP[5–7] Coarse delay select OTAP[8–15] TAP delay select
16–31	—	Reserved

18.4.2 DLL Status Register (DLLSR)

DLLSR shown in Figure 18-4, indicates the DLL status. It is a read only register.

Addr: 0x0_110C

Access: Read Only

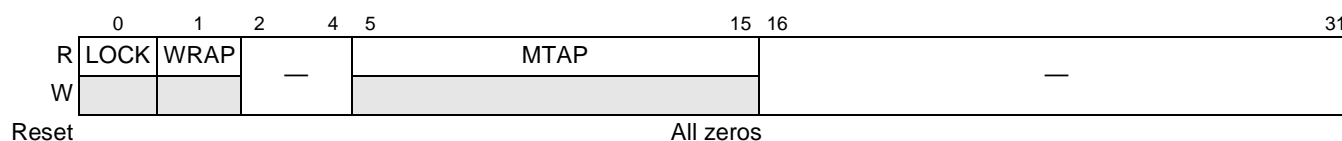


Figure 18-4. DLL Status Register

Table 18-4 describes DLSR fields.

Table 18-4. DLSR Field Description

Bits	Name	Description
0	LOCK	DLL locked. When the DLL finished its TAP point search, the LOCK bit is set. The LOCK bit must be considered together with the WRAP bit state.
1	WRAP	DLL wrapped. When the DLL LOCK is set, WRAP provides information for the DLL condition. If WRAP = 1, it indicates that the DLL search for a TAP point has completed unsuccessfully. When the DLL search was successful, the WRAP state remains clear while LOCK is set.
2–4	—	Reserved
5–15	MTAP	Measured coarse and tap delay by the DLL. MTAP[5–7] Coarse delay MTAP[8–15] Tap delay
16–31	—	Reserved

18.4.3 DLL Clock Register (DLLCK)

DLLCK shown in Figure 18-5, enables or disables the signals clock out.

Addr: 0x0_1110

Access: Read/Write

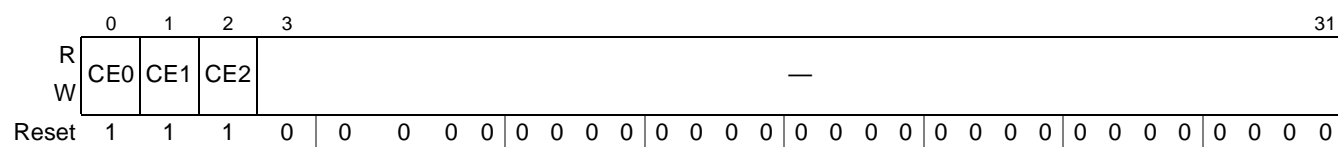


Figure 18-5. DLL Clock Register

Table 18-5 describes DLLCK fields.

Table 18-5. DLLCK Field Description

Bits	Name	Description
0	CE0	Enable/Disable LCLK[0] signal clock out 0 Disable LCLK[0] 1 Enable LCLK[0]
1	CE1	Enable/Disable LCLK[1] signal clock out 0 Disable LCLK[1] 1 Enable LCLK[1]
2	CE2	Enable/Disable LCLK[2] signal clock out 0 Disable LCLK[2] 1 Enable LCLK[2]
3–31	—	Reserved

Part IV

QUICC Engine Block

Part IV describes the QUICC Engine block of the MPC8360 integrated processor. The following chapter is included:

- [Chapter 19, “QUICC Engine Block on the MPC8360E](#), serves as both a general overview of the QUICC Engine block and a guide to the specific implementation of the QUICC Engine block on the MPC8360E. The *QUICC Engine Block Reference Manual with Protocol Interworking (QEIWRM)* describes all the functional units of the QUICC Engine block and must be used in conjunction with this device manual.
- [Appendix A, “MPC8358E,”](#) illustrates the MPC8358E and in particular the differences between it and the MPC8360E as described in this manual.
- [Appendix B, “Revision History,”](#) lists the major differences between revisions of this reference manual.

Chapter 19

QUICC Engine Block on the MPC8360E

The *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM) describes all the functional units of the QUICC Engine block and must be used in conjunction with this device manual and this chapter. The QEIWRM is a superset manual which includes some information not relevant to the MPC8360E. This chapter serves as both a general overview of the QUICC Engine block and a guide to the specific implementation of the QUICC Engine block on the MPC8360E.

- [Section 19.1, “QUICC Engine Block,”](#) gives a general overview of the QUICC Engine architecture and communication peripherals.
- [Section 19.2, “QUICC Engine Implementation Details for the MPC8360E,”](#) lists the chapters that do apply. Implementation-specific details for some chapters follow.

19.1 QUICC Engine Block

The QUICC Engine block is a versatile communications complex that integrates several communications peripheral controllers. It provides an on-chip system design that can be used as a building block for chip integration in a variety of applications, particularly in communications and networking systems.

The QUICC Engine block is the next generation of the Power QUICC II CPM and maintains a high level of compatibility with it.

The QUICC Engine block contains the following communication peripherals:

- Eight universal communication controllers (UCCs)
 - Ethernet, ATM, HDLC/HDLC bus and Transparent protocols (also known as fast protocols)
 - UART, BiSync, Async HDLC, Serial ATM and QMC (also known as slow protocols)
- Two UTOPIA-packet over SONET (POS) PHY L2 controllers (UPC) for 124/128 ports (only one UPC on the MPC8358E)
- Two serial peripheral controllers (SPI1 and SPI2). SPI2 can also be used for Ethernet PHY management.
- Multi channel controller (MCC) for 256 channels.
- Time slot assigner and serial interface (SI) for 8 TDMs and full duplex routing RAM of 512 entries.
- One universal serial bus controller (USB 2.0—full and low speed).

The UCCs are similar to the PowerQUICC II peripherals: SCC (BISYNC, UART, and HDLC bus), and FCC (fast Ethernet, HDLC, transparent, and ATM). In addition, 2×124 UTOPIA PHYs are supported in ATM mode. The QUICC Engine block presents enhanced flexibility by allowing the user to configure the UCCs to support a Layer-2 Ethernet switch.

Figure 19-1 shows the internal architecture and the interfaces provided by the QUICC Engine block. The QUICC Engine block contains two identical groups of four UCCs. Both groups are controlled by a RISC engine. A common multiuser RAM is used to store parameters for RISC engines. Each RISC has a ROM associated with it, which contains the code image. The instruction RAM is used to optionally run additional code.

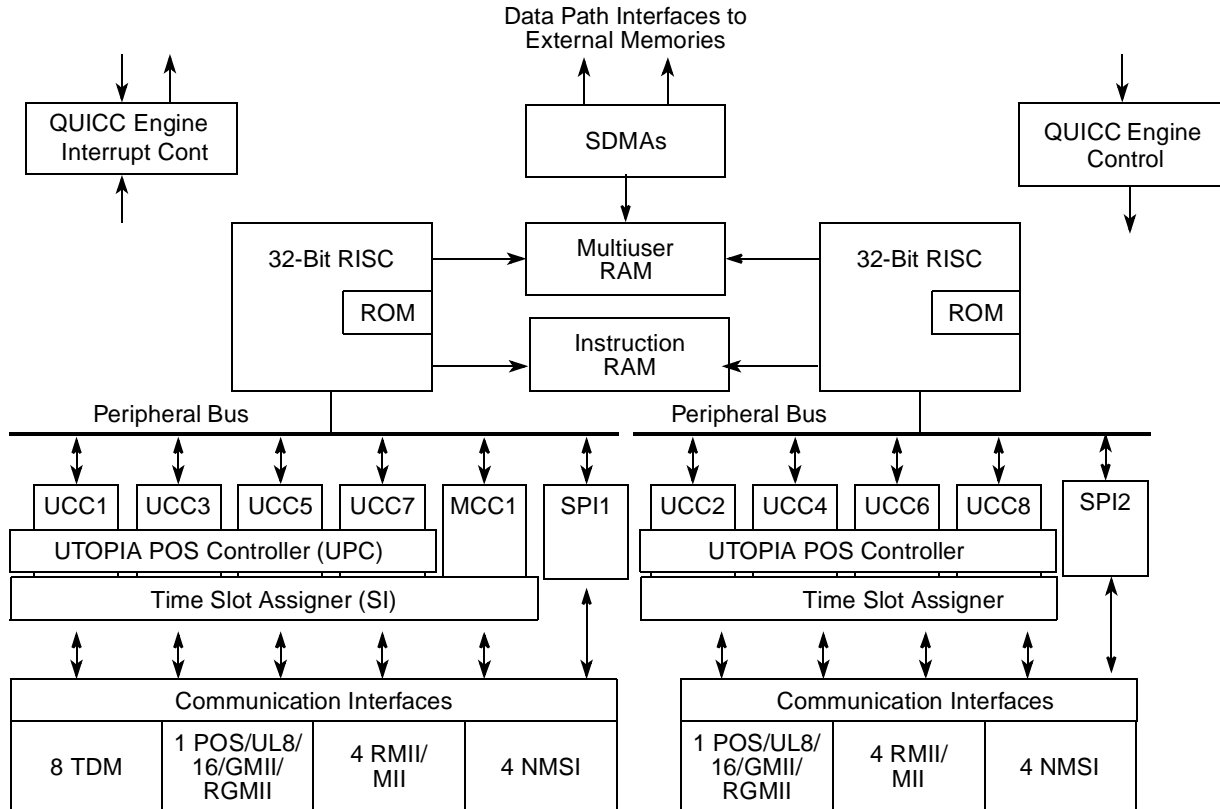


Figure 19-1. QUICC Engine Block Architectural Block Diagram

19.2 QUICC Engine Implementation Details for the MPC8360E

Most chapters of the QEIWRM apply to the MPC8360E without modification. However, some of these chapters have application differences that are pointed out in the QEIWRM. They are also given in this overview section. Although MPC8360E is used in this chapter, all the material applies to the MPC8358E device as well.

Note the chapters that have MPC8360E-specific details:

- [Section 19.2.1, “QUICC Engine System Interface](#)
- [Section 19.2.2, “QUICC Engine Block Control](#)
- [Section 19.2.3, “QUICC Engine Multiplexing and Timers](#)

While using the QEIWRM for the MPC8360E, apply this MPC8360E implementation-specific information in general:

- e300 core

- Two 32-bit RISCs

Table 19-1 lists the chapters from the *QUICC Engine Block Reference Manual with Protocol Interworking (QEIWRM)* and the chapters with implementation differences.

Table 19-1. QERM Chapters and MPC8360E Implementation

Chapters	MPC8360E Implementation
Part I “Introduction”	
System Interface	Applies to MPC8360E —see Section 19.2.1, “QUICC Engine System Interface,” for MPC8360E implementation.
Configuration	Applies to MPC8360E —see Section 19.2.2, “QUICC Engine Block Control,” for MPC8360E implementation.
Multiplexing and Timers	Applies to MPC8360E —see Section 19.2.3, “QUICC Engine Multiplexing and Timers,” for MPC8360E implementation.
Part II “Unified Communication Controllers (UCCs)”	
Unified Communications Controllers (UCCs)	Applies to MPC8360E
UCC for Fast Protocols	Applies to MPC8360E
UCC Ethernet (UEC)	Applies to MPC8360E
IEEE Standard 1588 Assist	Applies to MPC8360E
UTOPIA POS Bus Controller (UPC)	Applies to MPC8360E
ATM Controller AAL0, AAL1, and AAL5	Applies to MPC8360E
ATM Adaptation Layer 2	Applies to MPC8360E
UCC POS Controller (UPOS)	Applies to MPC8360E
HDLC Controller	Applies to MPC8360E
Transparent Controller	Applies to MPC8360E
UCC as Slow Communications Controllers	Applies to MPC8360E
UCC UART Mode and Asynchronous HDLC	Applies to MPC8360E
Serial Peripheral Interface (SPI)	Applies to MPC8360E
Universal Serial Bus Controller	Applies to MPC8360E
UCC BISYNC Mode	Applies to MPC8360E
Part III “Time Division Multiplex Support (TDM)”	
Serial Interface with Time-Slot Assigner	Applies to MPC8360E
Multi-Channel Controller (MCC)	Applies to MPC8360E
QMC (QUICC Multi-Channel Controller)	Applies to MPC8360E
Point-to-Point Protocol (PPP)	Applies to MPC8360E

Table 19-1. QERM Chapters and MPC8360E Implementation (continued)

Chapters	MPC8360E Implementation
Serial ATM Microcode	Applies to MPC8360E
Inverse Multiplexing for ATM (IMA)	Applies to MPC8360E
ATM AAL1 Circuit Emulation Service	Applies to MPC8360E
Part IV “Multiprotocol Interworking”	
Frame Parse and Lookup	Applies to MPC8360E
Protocol Interworking Programming Model	Applies to MPC8360E
Virtual Port	Applies to MPC8360E
IP Reassembly	Applies to MPC8360E
IPv4/UDP Header Compression	Applies to MPC8360E
Part V “Switching Functionality	
Enhanced MSP Microcode	Applies to MPC8360E
L2 Ethernet Switch	Applies to MPC8360E

The following subsections include MPC8360E-specific details for the given chapters of the QEIWRM.

19.2.1 QUICC Engine System Interface

The information in this MPC8360E-specific “System Interface” subsection applies to “System Interface” chapter of the QEIWRM.

19.2.1.1 System Interface—Serial DMA

In the “Serial DMA” subsection, the e300 core information applies for the MPC8360E.

The QUICC Engine module has two physical serial DMA (SDMA) channels. On the MPC8360E, one channel interfaces with coherent system bus, the other channel interfaces with the secondary bus.

19.2.1.2 System Interface—Data Paths

In the “Data Paths” subsection, use this e300 core information:

[Figure 19-2](#) is a simplified MPC8360E block diagram that shows the data paths. They may be configured with one DDR bus (32 or 64 bit data) or with two DDR buses (32-bit data). As shown in the figures, depending on the external bus configuration, the secondary bus of the QUICC Engine module may be connected to the second DDR and/or to the Local bus. Accesses of the QUICC Engine module to the secondary bus are not seen on the coherent system bus. This allows for concurrent bus transactions on the two buses (marked as ‘1’ and ‘2’ in the figure).

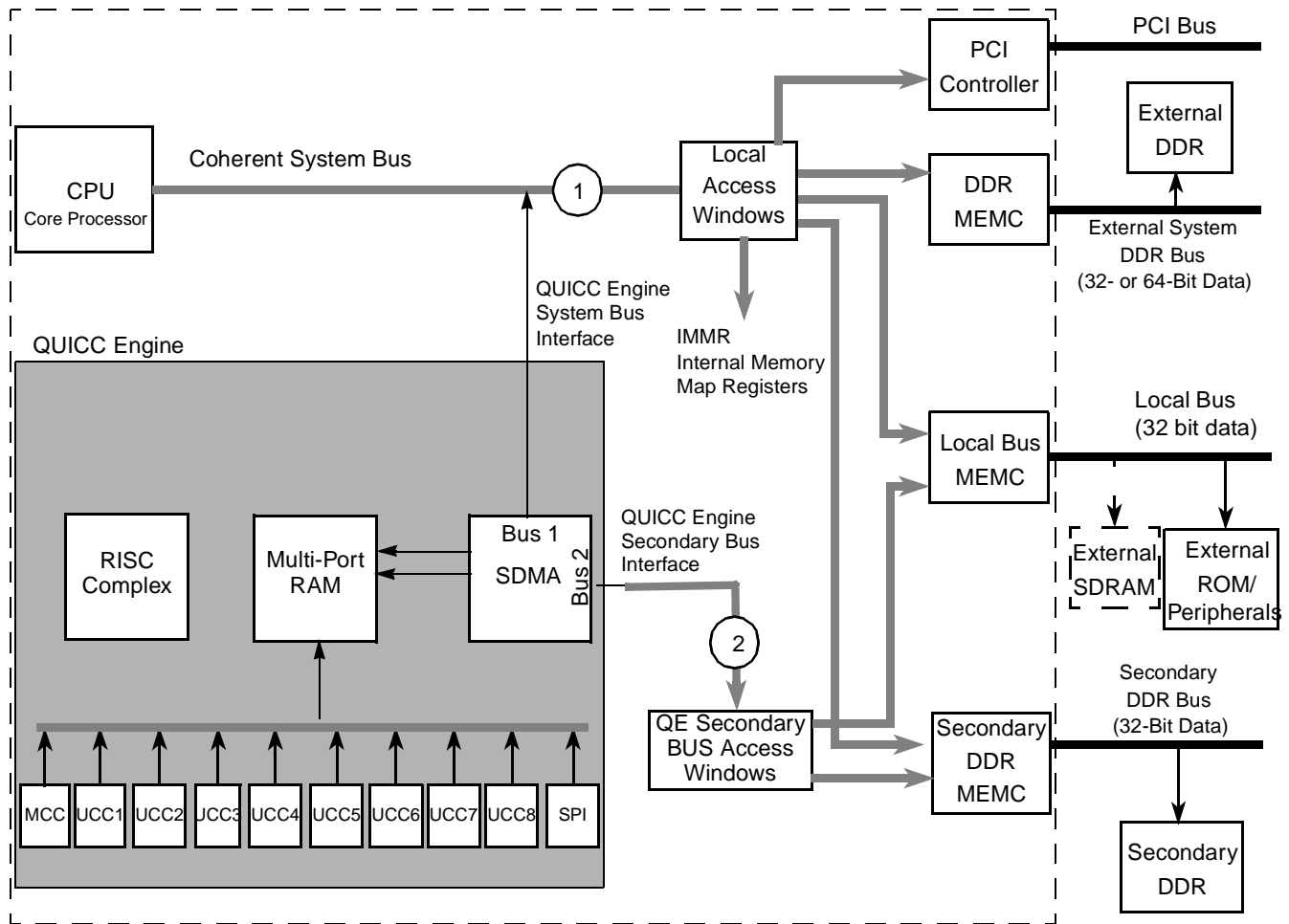


Figure 19-2. Data Paths

19.2.1.3 System Interface—SDMA and Bus Error

In the “SDMA and Bus Error” subsection, use the following e300 core information:

Under 3b. “On the MPC8360E, the device is reset by asserting the soft reset signal, SRESET, (the reset command to the QUICC Engine Command Register is not sufficient).”

19.2.1.4 System Interface—SDMA and Reset

In the “SDMA and Reset” subsection, use the following e300 core information:

During system reset (on the MPC8360E, $\overline{\text{SRESET}}$) all SDMA FIFOs are flushed and all outstanding transactions are stopped.

19.2.1.5 System Interface—Arbitration over the System Bus

In the “Arbitration over the System Bus” subsection, use the following e300 core information:

On the MPC8360E, the QUICC Engine block arbitrates over the system bus by requesting access to the bus from the system arbiter. The arbiter supports four levels of priorities. The QUICC Engine

module also asserts a REPEAT signal to the arbiter for transactions which are longer than one burst (32 bytes) on the System Bus. In this way, the arbiter can optimize bus grants to allow for Page Hits on the DRAMs, by allowing back to back cycles to subsequent addresses.

19.2.1.6 System Interface—Arbitration Over the Secondary Bus

In the “Arbitration Over the Secondary Bus” subsection, use the e300 core information:

On the MPC8360E, arbitration over the Secondary bus is not affected by internal state of the SDMA. Generally arbitration occurs in two places: Local Bus Memory Controller and Secondary DDR SDRAM Memory Controller. In both places arbitration is between the QUICC Engine module, which accesses the resource through its secondary bus, and another master which initiated a transaction on the System Bus. In the case of the Local Bus Memory Controller, the QUICC Engine module has the lower priority. In the case of the Secondary DDR SDRAM Controller arbitration is based on rotating priority.

19.2.2 QUICC Engine Block Control

The information in this MPC8360E-specific “QUICC Engine Block Control” subsection applies to “QUICC Engine Block Control” chapter of the QERM.

19.2.2.1 QUICC Engine Block Control—CERCR[CIR]

In the “QUICC Engine Block Control” chapter, the CERCR[CIR] bit under the “QUICC Engine RAM Control Register (CERCR)” section has this description for the MPC8360E.

Table 19-2. CERCR Field Descriptions

4	CIR	<p>Common instruction RAM. The size varies by device (either 48- or 64-Kbyte). Note: Check your device reference manual “QUICC Engine Block” chapter for implementation details.</p> <p>0 Each of the two RISC processors has its own instruction RAM. Performance may be better since RISCs are not competing on a common resource when fetching instructions. 1 Both RISC processors share a common instruction RAM</p> <hr/> <p>As implemented on the MPC8360E for 48-Kbyte common instruction RAM devices: 0 Each of the two RISC processors has its own 24-Kbyte instruction RAM. Performance may be better since RISCs are not competing on a common resource when fetching instructions. When CIR=0, the instruction ram is seen through IADD as follows:</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">RISC0</td> <td style="text-align: center;">RISC1</td> </tr> <tr> <td style="text-align: center;">0x8_0000–0x8_5FFF</td> <td style="text-align: center;">0x8_8000–0x8_DFFF</td> </tr> </table> <p>In case the two RISCs should run the same code, the code should be duplicated in the regions for RISC0 and RISC1. 1 Both RISC processors are sharing a common 48-Kbyte instruction RAM (must be used when instruction RAM code is more than 24 Kbyte). When CIR=1, the instruction RAM is seen as a consecutive 48-Kbyte region at addresses 0x80000–0x8BFFF.</p>	RISC0	RISC1	0x8_0000–0x8_5FFF	0x8_8000–0x8_DFFF
RISC0	RISC1					
0x8_0000–0x8_5FFF	0x8_8000–0x8_DFFF					

19.2.2.2 QUICC Engine Block Control—QUICC Engine Microcode Revision

The CEURNR[REV_NUM] bit in Section, “QUICC Engine Microcode Revision (CEURNR),” in the *QUICC Engine Block Reference Manual with Protocol Interworking*, has the following field descriptions for the MPC8360E.

Table 19-3. CEURNR Field Descriptions

Bits	Name	Description
0–31	REV_NUM	Microcode revision number. For MPC8360E, the value is 0xCE00_0000.

19.2.2.3 QUICC Engine Block Control—External Requests Device Specific Information

In the “External Requests Device Specific Information” subsection, use this MPC8360-specific information:

The MPC8360E supports the connection of a few system level communication peripherals to the QUICC Engine block. The user may configure the connection of the interrupt request signal to be connected either to the Programmable Interrupt Controller (PIC), or to be connected to the QUICC Engine block. Routing to the QUICC Engine block allows it to handle the interrupts, thus freeing up the CPU. In order to program the routing of the interrupt signals, the user programs registers CPCE_xR (x=1..4) (See the “Signal Description” chapter of *MPC8360E PowerQUICC II Pro Integrated Processor Family Reference Manual*.) The following table describes the mapping of the External Hardware Interrupts to the QUICC Engine External Requests (EXT[1..4]).

Table 19-4. Interrupt Routing and External Pins to QUICC Engine Block in MPC8360

Interrupt Source	SNUM Name ¹	Comment
According to configuration programmed in CPCE1R register	EXT1	See the “Communication Peripherals to QUICC Engine Mux Control Registers (CPCE1R–CPCE4R),” section in the “Signal Description” chapter of the <i>MPC8360E Integrated Processor Reference Manual</i>
According to configuration programmed in CPCE2R register	EXT2	
According to configuration programmed in CPCE3R register	EXT3	
According to configuration programmed in CPCE4R register	EXT4	

¹ See SNUM table

19.2.3 QUICC Engine Multiplexing and Timers

The information in this MPC8360E-specific “QUICC Engine Multiplexing and Timers” subsection applies to the “QUICC Engine Multiplexing and Timers” chapter of the QEIWRM.

19.2.3.1 QUICC Engine Multiplexing and Timers—CMXUCR n

In the QUICC Engine multiplexing and timers logic (CMX) UCC Clock Route (CMXUCR) registers, the HDLC bus mode fields [HBM n] vary by device. The location for CMXUCR n [HBM n] on the MPC8360E follows this pattern of as shown in the CMXUCR1 example using bit fields 7 and 23 in [Figure 19-3](#).

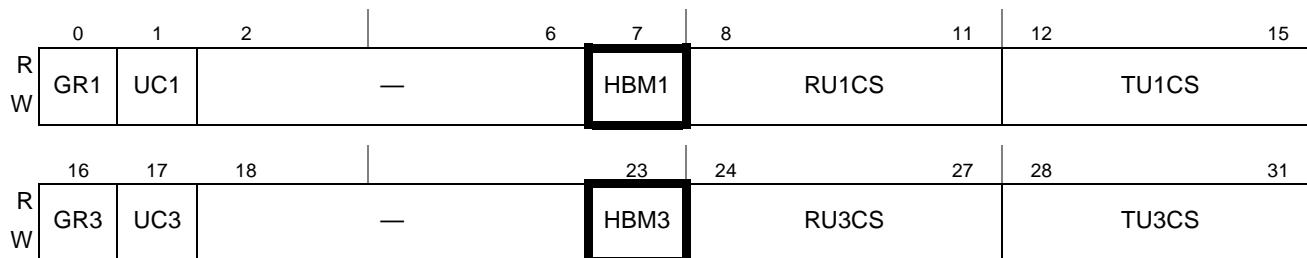


Figure 19-3. CMXUCR1[HBM1] and CMXUCR1[HBM3] location on the MPC8360E

For the MPC8360E, the locations for the HBM n fields are as follows for the four CMXUCR n registers:

- CMXUCR1[HBM1] is CMXUCR1[7] as shown in [Figure 19-3](#).
- CMXUCR1[HBM3] is CMXUCR1[23] as shown in [Figure 19-3](#).
- CMXUCR2[HBM5] is CMXUCR2[7].
- CMXUCR2[HBM7] is CMXUCR2[23].
- CMXUCR3[HBM2] is CMXUCR3[7].
- CMXUCR3[HBM4] is CMXUCR3[23].
- CMXUCR4[HBM6] is CMXUCR4[7].
- CMXUCR4[HBM8] is CMXUCR4[23].

19.2.3.2 QUICC Engine Multiplexing and Timers—Global Timer Module (GTM)

In the “Global Timer Module” subsection, note that for the MPC8360E the $\overline{\text{TGATE}}_n$, TIN_n , and $\overline{\text{TOUT}}_n$ signals are not pinned out. Use the MPC8360E platform GTM, if external signals are needed.

Appendix A

MPC8358E

This appendix illustrates the MPC8358E, and in particular, the differences between it and the MPC8360E as described in this manual.

A.1 Features

Major features of the MPC8358E are as follows:

- e300c1 PowerPC processor core
 - Enhanced version of the MPC603e processor core
 - High-performance, superscalar processor core
 - Floating-point, integer, load/store, system register, and branch processing units
 - 32-Kbyte instruction cache, 32-Kbyte data cache
 - Dynamic power management
 - Enhanced hardware program debug features
 - Software-compatible with the Freescale processor families implementing the PowerPC architecture
- QUICC Engine unit
 - Two 32-bit RISC controllers for flexible support of the communications peripherals
 - Serial DMA channel for receive and transmit on all serial channels
 - QE peripheral request interface (for SEC, PCI, IEEE Std 1588™)
 - Six UCCs supporting the following protocols and interfaces (not all of them simultaneously):
 - 10/100 Mbps Ethernet/IEEE Std 802.3™ through MII and RMII interfaces.¹
 - 1000 Mbps Ethernet/IEEE Std 802.3 through a media-independent interface (GMII, RGMII, RTBI, TBI) on UCC1 and UCC2
 - 10/100 Mbps Ethernet/IEEE Std 802.3 L2 switch port through MII and RMII interfaces.
 - IEEE Std 1588 protocol supported
 - 9.6K jumbo frames
 - ATM full-duplex SAR, up to 622 Mbps (OC-12/STM-4), AAL0, AAL1 and AAL5 in accordance ITU-T I.363.5
 - ATM AAL2 CPS, SSSAR, and SSTED up to 155 Mbps (OC-3/STM-1) Mbps full duplex (with 4 CPS packets per cell) in accordance ITU-T I.366.1 and I.363.2
 - ATM traffic shaping for CBR, VBR, UBR, and GFR traffic types compatible with ATM forum TM4.1 for up to 64K simultaneous ATM channels

¹.SMII or SGMII media-independent interface is not currently supported

- ATM AAL1 structured and unstructured circuit emulation service (CES 2.0) in accordance with ITU-T I.163.1 and ATM Forum af-vtoa-00-0078.000
- IMA (Inverse Multiplexing over ATM) for up to 31 IMA links over 8 IMA groups in accordance with the ATM forum AF-PHY-0086.000 (Version 1.0) and AF-PHY-0086.001 (Version 1.1)
- ATM Transmission Convergence layer support in accordance with ITU-T I.432
- ATM OAM handling features compatible with ITU-T I.610
- IP support for IPv4 and IPv6 packets including TOS, TTL and header checksum processing
- Ethernet over first mile IEEE Std 802.3ah™
- Shim header
- Ethernet-to-Ethernet/AAL5/AAL2 inter-working
- L2 Ethernet switching using MAC address or IEEE Std 802.1P/Q™ VLAN tags
- ATM (AAL2/AAL5) to Ethernet (IP) interworking in accordance with RFC2684 including bridging of ATM ports to Ethernet ports
- Extensive support for ATM statistics and Ethernet RMON/MIB statistics
- AAL2 protocol rate up to 4 CPS at OC-3/STM-1 rate
- Packet over Sonet (POS) up to 622-Mbps full-duplex 124 MultiPHY¹
- POS hardware; microcode must be loaded as an IRAM package
- Transparent up to 70-Mbps full-duplex
- HDLC up to 70-Mbps full-duplex
- HDLC BUS up to 10 Mbps
- Asynchronous HDLC
- UART
- BISYNC up to 2 Mbps
- User-programmable FIFO size
- QUICC Multichannel Controller (QMC) for 64 TDM channels
- One UTOPIA/POS interface supporting 31/124 MultiPHY
- Universal serial bus (USB) controller
- Two serial peripheral interfaces (SPI); SPI2 is dedicated to Ethernet PHY management
- Four TDM interfaces with 1-bit mode for E3/T3 rates in clear channel
- Sixteen independent baud rate generators and 30 input clock pins for supplying clocks to UCC serial channels
- Four independent 16-bit timers that can be interconnected as four 32-bit timers
- Interworking functionality:
 - Layer 2 10/100-Base T Ethernet switch
 - ATM-to-ATM switching (AAL0, 2, 5)
 - Ethernet-to-ATM switching with L3/L4 support
 - PPP interworking

- Security engine is optimized to handle all the algorithms associated with IPsec, SSL/TLS, SRTP, 802.11i, iSCSI, and IKE processing. The security engine contains four crypto-channels, a controller, and a set of crypto execution units (EUs).
 - Public key execution unit (PKEU) supporting the following:
 - RSA and Diffie-Hellman
 - Programmable field size up to 2048 bits
 - Elliptic curve cryptography
 - F2m and F(p) modes
 - Programmable field size up to 511 bits
 - Data encryption standard execution unit (DEU)
 - DES, 3DES
 - Two key (K1, K2) or three key (K1, K2, K3)
 - ECB and CBC modes for both DES and 3DES
 - Advanced encryption standard unit (AESU)
 - Implements the Rijndael symmetric key cipher
 - Key lengths of 128, 192, and 256 bits, two key
 - ECB, CBC, CCM, and counter modes
 - ARC four execution unit (AFEU)
 - Implements a stream cipher compatible with the RC4 algorithm
 - 40- to 128-bit programmable key
 - Message digest execution unit (MDEU)
 - SHA with 160-, 224-, or 256-bit message digest
 - MD5 with 128-bit message digest
 - HMAC with either SHA or MD5 algorithm
 - Random number generator (RNG)
 - Four crypto-channels, each supporting multi-command descriptor chains
 - Static and/or 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
 - Storage/NAS XOR parity generation accelerator for RAID applications
- DDR SDRAM memory controller
 - Programmable timing supporting DDR SDRAM
 - DDR bus can be configured as a 32-bit or 64-bit bus
 - 32- or 64-bit data interface
 - Four banks of memory, each up to 1 Gbyte
 - DRAM chip configurations from 64 Mbits to 1 Gigabit with x8/x16 data ports
 - Full ECC support
 - Page mode support (up to 16 simultaneous open pages for DDR1, up to 32 simultaneous open pages for DDR2)

- Contiguous or discontinuous memory mapping
- Read-modify-write support
- Sleep mode support for self refresh SDRAM
- Supports auto refreshing
- Supports source clock mode
- On-the-fly power management using CKE
- Registered DIMM support
- 2.5-V SSTL2 compatible I/O for DDR1, 1.8-V SSTL2 compatible I/O for DDR2
- External driver impedance calibration
- On-die termination (ODT)
- PCI interface
 - PCI Specification Revision 2.3 compatible
 - Data bus widths:
 - Single 32-bit data PCI interface that operates at up to 66 MHz
 - PCI 3.3-V compatible (not 5-V compatible)
 - PCI host bridge capabilities on both interfaces
 - PCI agent mode supported on PCI interface
 - Support for PCI-to-memory and memory-to-PCI streaming
 - Memory prefetching of PCI read accesses and support for delayed read transactions
 - Support for posting of processor-to-PCI and PCI-to-memory writes
 - On-chip arbitration, supporting five masters on PCI
 - Support for accesses to all PCI address spaces
 - Parity support
 - Selectable hardware-enforced coherency
 - Address translation units for address mapping between host and peripheral
 - Dual address cycle supported when the device is the target
 - Internal configuration registers accessible from PCI
- Local bus controller (LBC)
 - Multiplexed 32-bit address and data
 - Eight chip selects support eight external slaves
 - Up to eight-beat burst transfers
 - 32-, 16-, and 8-bit port sizes are controlled by an on-chip memory controller
 - Three protocol engines available on a per chip select basis:
 - General-purpose chip select machine (GPCM)
 - Three user programmable machines (UPMs)
 - Dedicated single data rate SDRAM controller

- Parity support
- Default boot ROM chip select with configurable bus width (8-, 16-, or 32-bit)
- Programmable interrupt controller (PIC)
 - Functional and programming compatibility with the MPC8260 interrupt controller
 - Support for 8 external and 35 internal discrete interrupt sources
 - Support for one external (optional) and seven internal machine checkstop interrupt sources
 - Programmable highest priority request
 - Four groups of interrupts with programmable priority
 - External and internal interrupts directed to communication processor
 - Redirects interrupts to external $\overline{\text{INTA}}$ pin when in core disable mode
 - Unique vector number for each interrupt source
- Dual industry-standard I²C interfaces
 - Two-wire interface
 - Multiple master support
 - Master or slave I²C mode support
 - On-chip digital filtering rejects spikes on the bus
 - System initialization data is optionally loaded from I²C-1 EPROM by boot sequencer embedded hardware
- DMA controller
 - Four independent virtual channels
 - Concurrent execution across multiple channels with programmable bandwidth control
 - All channels accessible by local core and remote PCI masters
 - Misaligned transfer capability
 - Data chaining and direct mode
 - Interrupt on completed segment and chain
 - DMA external handshake signals: $\overline{\text{DMA_DREQ}}[0:3]/\overline{\text{DMA_DACK}}[0:3]/\overline{\text{DMA_DONE}}[0:3]$. There is one set for each DMA channel. The pins are multiplexed to the parallel IO pins with other QE functions.
- DUART
 - Two 4-wire interfaces (RxD, TxD, RTS, CTS)
 - Programming model compatible with the original 16450 UART and the PC16550D
- System timers
 - Periodic interrupt timer
 - Real-time clock
 - Software watchdog timer
 - Eight general-purpose timers
- IEEE Std 1149.1™ compliant, JTAG boundary scan
- Integrated PCI bus and SDRAM clock generation

A.2 MPC8360E Features Not Present in the MPC8358E

MPC8360E features that are not present in the MPC8358E are as follows:

- UCC interfaces 6 and 7
- MCC
- Two UPCs
- TDM interfaces E, F, G, and H
- DDRC2

Table A-1 shows a comparison between the MPC8358E and the MPC8360E.

Table A-1. MPC8358E and MPC8360E Comparison

	MPC8358E	MPC8360E
Core	e300	e300
CPU Speed	Up to 400 MHz	Up to 667 MHz
QUICC Engine Speed	Up to 400 MHz	Up to 500 MHz
L1 Instruction/Data Cache	32 K Instruction 32 K Data	32 K Instruction 32 K Data
Memory Controller	1x32-bit or 1x64-bit DDR1	1x64-bit or 2x32-bit DDR1
Local bus	Yes	Yes
PCI	1- 32-bit up to 66MHz	1- 32-bit up to 66MHz
Ethernet	Up to 6- 10/100 Up to 2- 10/100/1000	Up to 8- 10/100 Up to 2- 10/100/1000
HDLC/TDMs	Up to 128/4 (QMC ucode)	Up to 256 / 8
UTOPIA	1x31 or 1x128 MPHY	2x128 MPHY
USB	Low/Full Speed	Low/Full Speed
Security Engine	E version only	E version only
UART	Dual	Dual
I²C	Dual	Dual
SPI	Dual	Dual
MCC	No	One
Interrupt Controller	Yes	Yes
Package	740 TBGA (37.5x37.5) 1 mm pitch	740 TBGA (37.5x37.5) 1 mm pitch

A.3 MPC8358E Block Diagram

Figure A-1 shows the MPC8358 block diagram.

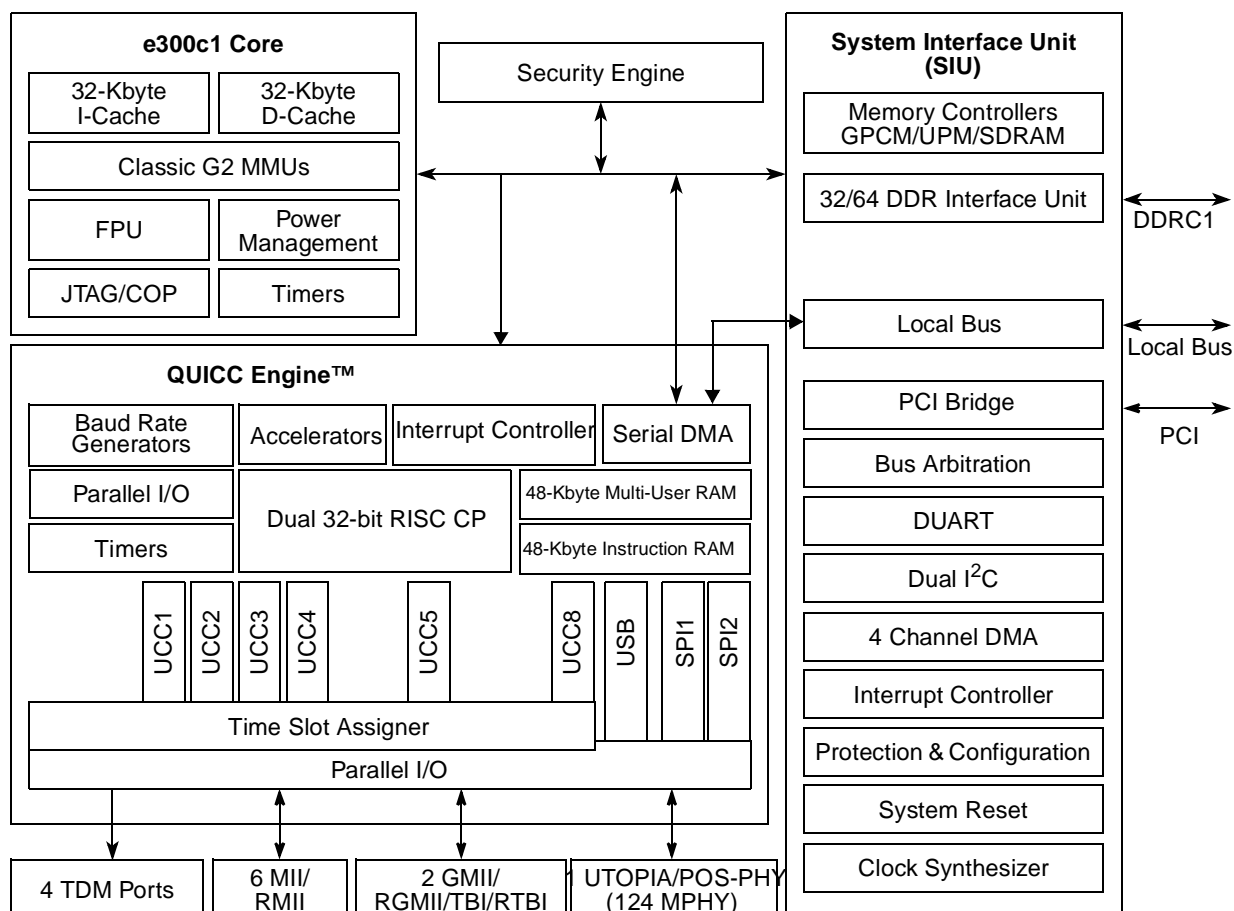


Figure A-1. MPC8358 Block Diagram

A.4 Implementation and Impact of Six UCCs

References to UCC6 and UCC7 are not applicable because the MPC8358 contains six UCCs (UCC1–5 and UCC8) instead of eight UCCs.

A.4.1 Unimplemented Signals

The following signals that support UCC6 and UCC7 are not implemented on MPC8358E:

- CMXUCR4[GR6]
- CMXUCR4[UC6]
- CMXUCR4[RU6CS]
- CMXUCR4[TU6CS]
- CMXUCR2[GR7]
- CMXUCR2[UC7]

- CMXUCR2[RU7CS]
- CMXUCR2[TU7CS]

NOTE

The alternative functionality of the associated port pins remains available, as described in [Chapter 3, “Signal Descriptions,”](#) and [Section 3.4, “Parallel I/O Ports.”](#)

A.4.2 Reserved Memory Map Areas

The following memory-mapped areas are reserved because UCC6 and UCC7 are not available:

- 0x0_2600–0x0_27FF
- 0x0_3400–0x0_35FF

A.4.3 UCC General Setup

Although most UCC registers contain configurations bits for eight UCCs, ensure that these are only configuring UCC1–5 and UCC8. See *QUICC Engine Block Reference Manual with Protocol Interworking*. Registers affected include CIPYCC, CMXGCR, CECR, MTC_TX_ATM_PRAM, MTC_RX_ATM_PRAM. If interrupts are used, ensure that only valid UCCE_x, UCCM_x, bits in CIMR and bits in CIPNR that correspond to UCC1–5 and UCC8 are manipulated and observed.

A.5 MCC

All references to MCC registers are not applicable since MPC8358 does not contain MCC. Thus, all SIRAM entries must be set for UCC (SIRAM[0] must always equal zero).

A.6 UPC Availability

Due to internal constraints, the MPC8358E supports only one UPC, either UPC1 or UPC2. Some features may become unavailable when one UPC is chosen over the other due to shared functionality on the port pins. Therefore, the user should choose either UPC1 or UPC2 depending on other features being utilized on the device. For example, some MII options are unavailable when UPC1 is chosen and PCI is unavailable when using UPC2, refer to [Section 3.4, “Parallel I/O Ports,”](#) for pin muxing limitations.

The CMXUPCR register contains configurations bits for both UPCs, ensure that these are only configuring one UPC. See *QUICC Engine Block Reference Manual with Protocol Interworking*.

NOTE

The alternative functionality of the associated port pins remains available, as described in [Chapter 3, “Signal Descriptions,”](#) and [Section 3.4, “Parallel I/O Ports.”](#)

A.7 Unimplemented TDM Registers

The following registers that support TDME-TDMH are not implemented on MPC8358E:

- CMXSI1CRH
- SIAMR
- SIBMR
- SICMR
- SIDMR
- SITERC
- SITFRC
- SITGRC
- SITHRC
- SIRERC
- SIRFRC
- SIRGRC
- SIRHRC
- SIGLMRL

NOTE

The alternative functionality of the associated port pins remains available, as described in [Chapter 3, “Signal Descriptions,”](#) and [Section 3.4, “Parallel I/O Ports.”](#)

A.8 DDR

The MPC8358 has only one DDR SDRAM memory controller (DDRC1). Thus, it can only support one 32-bit or one 64-bit bus. Therefore all references to the secondary DDR controller do not apply to the MPC8358E. In the memory map, these items do not apply:

- 0x00_D000–0x00_DFFF—secondary DDR memory controller
- 0x0_1804 SDMCSAR—secondary DDR memory controller start address register
- 0x0_1844 SDMCEA
- 0x0_1884 SDMCAR



Appendix B

Revision History

This appendix provides a list of the major differences between revisions of the *MPC8360E PowerQUICC II Pro Integrated Communications Processor Family Reference Manual*. Note that this list only covers the major changes.

B.1 Changes From Revision 2 to Revision 3

Section, Page No.	Changes
1.1/1-1	Changed maximum bit rate from 70 to 50 Mbps in the following sub-bullet: “HDLC/Transparent (bit rate up to 50 Mbps).”
2.3/2-1	In Table 2-1 , “IMMR Memory Map,” removed 0x00_0500–0x00_06FF, Reserved row.
2.3/2-1	Modified the DLL rows Table 2-1 , “IMMR Memory Map.”
2.3/2-1	In Table 2-2 , “Memory Map,” modified the DLL section to comprise Clock Control Secondary DDR, Clock Control Primary DDR, and Clock and DLL Control, LBC.
2.3/2-1	In Table 2-2 , “Memory Map,” added Communication Peripherals to QUICC Engine Mux Registers at offsets 0x0_14B8, 0x0_14BC, 0x0_14C0, and 0x0_14C4.
Chapter 3	Changed “MA[0:14]” to “MA[14:0]” throughout.
Chapter 4	Changed field name “LBIUCM” to “LBCM” throughout.
4.3.1.1/4-9	In Table 4-5 , “Reset Configuration Words Source,” updated RCW source option 010.
4.3.2.1/4-12	In Table 4-8 , “RCWLR Bit Settings,” updated RCWLR to show field SVCOD as bits 2–3.
4.3.2.1.1/4-13	Added Section 4.3.2.1.1 , “System PLL VCO Division.”
4.3.2.2/4-17	In Figure 4-4 , “Reset Configuration Word High Register (RCWHR),” and in Table 4-13 , “Reset Configuration Word High Bit Settings,” changed reserved bit 29 to LALE with the following bit field description: “Local bus LALE signal timing. See Section 4.3.2.2.7 , “LALE Configuration,” for more information.”
4.3.2.2.7/4-22	Added section heading “LALE Configuration,” and introductory sentence.
4.3.3.1/4-23	Added the following text: <p>“Note that LCS0 is asserted low during the assertion of PORESET and HRESET. Also, PORESET negation to LALE assertion is 36 cycles of PCI_SYNC_IN/PCI_CLK clock signal.”</p>

- 4.3.3.1/4-23 Updated [Figure 4-5](#), “Loading Reset Configuration Words from Local Bus,” and [Figure 4-6](#), “Loading Reset Configuration Words from Local Bus (continued).”
- 4.3.3.1/4-23 Added the following text to the end of the first paragraph: “+ LCS0 is the default for GPCM, so GPCM controlled is used to read the reset configuration word from EEPROM. /LGTA should be high to avoid unintended early termination of the read cycle.”
- 4.4.3/4-32 Added clarification/stipulation for the formula, $ce_clk = (\text{primary clock input} \times \text{CEPMF}) \div (1 + \text{CEPDF})$ and added additional formula.
- 4.4.3/4-32 Changed $\overline{\text{CFG_CLKIN_DIV}}$ to be active high (CFG_CLKIN_DIV).
- 4.5.1/4-34 In [Table 4-28](#), “Reset Configuration and Status Registers Memory Map,” for address 0x0_0910, changed “Reset status register (x)” to say “Reset status register (RSR).”
- 4.5.1.6/4-37 Changed the first sentence to the following: “RCR, shown in [Figure 4-13](#), can be used by software to initiate a hard reset sequence.”
- 4.5.2.1/4-39 In [Figure 4-15](#), “System PLL Mode Register,” changed reset value from “All zeros” to “*nnnn_nnnn_nnnn_nnnn*.”
- 4.5.2.3/4-41 In [Table 4-37](#), “SCCR Bit Descriptions,” added the encoding for “01” and a note to the ENCCM (bits 6–7) field description.
- 4.5.3/4-41 Added the following note:

NOTE

There is a DDR clock control register for each DDR controller. MCKENR1, for the primary controller, is at 0x00_1010 (offset 0x10 from the 0x0_1000); MCKENR2 (secondary controller) is at 0x00_0F10 (offset 0x10 from the 0x0_0F00).

- 4.5.3/4-41 Modified the MCKENR rows in [Table 4-38](#), “Clock Control DDR Register Address Map.”
- 4.5.3.1/4-42 Changed “MCK Enable Register (MCKENR)” to “MCK Enable Register (MCKENR n).”
- 5.4.2/5-24 In [Table 5-33](#), “System Configuration Register Memory Map,” updated SICRL reset value.
- 5.4.3.6/5-30 In second paragraph, changed “A value of 0b11 is illegal for all groups,” to “A value of 0b11 selects GPIO mode of the appropriate pin.”
- 5.4.3.7.1/5-32 Changed “either” to say “enter” in first sentence.
- 5.4.3.8/5-33 In first two paragraphs, changed 18 Ω to 18.2 Ω .
- 5.4.3.8/5-33 In [Table 5-43](#), “DDRCDR Field Descriptions,” added a note to DDR_TYPE (bit 13) field description:
- 5.4.3.9/5-35 In [Figure 5-26](#), “DDR Debug Status Register (DDRDSR),” made bits read-only to be consistent with the register’s read-only access.
- 5.5.5.2/5-41 Removed “This is the default value after soft reset” from bulleted list.

5.6.5.5/5-47	In Table 5-56 , updated AIF (bit 30) field description.
5.7.2/5-50	In bulleted list of features, changed “Maintains a 32-bit down-counter, clocked by a 16-bit prescaled input clock,” to say “Maintains a 32-bit down-counter, clocked by a 32-bit prescaled input clock.”
5.7.2/5-50	In bulleted list of features, removed “Provides maximum period of ~9.5 days (for 333-MHz system clock.”
5.7.5.5/5-54	In Table 5-65 , “PTEVR Bit Settings,” updated PIF (bit 31) field description.
5.8.1/5-56	Added footnote to Figure 5-48 , “Global Timers Block Diagram.”
5.8.2/5-57	In features list, updated two bulleted items as follows: <ul style="list-style-type: none"> • Maximum period of ~206 seconds (at 333-MHz bus clock in slow go mode, primary and secondary prescaler = 256) for 16-bit timer • Maximum period of ~3298 seconds (at 333-MHz bus clock and prescaler = 256) for 32-bit timer
5.8.3/5-57	Added bullet list.
5.8.4/5-58	In Table 5-67 , “GTM External Signals—Detailed Signal Descriptions,” for $\overline{\text{UART_RTS}}[1:2]$, changed sentence in description from: “Can be programmed to be automatically negated and asserted by either the receiver or transmitter” to: “Can be programmed to be negated and asserted by either the receiver or transmitter.”
5.8.4/5-58	In Table 5-67 , “GTM External Signals—Detailed Signal Descriptions,” updated the State Meaning descriptions of signal $\overline{\text{TGATE}}_n$ and $\overline{\text{TOUT}}_n$, and updated signal description of MA[14:0].
5.8.4/5-58	Added text in about external signals $\overline{\text{TGATE}}_n/\overline{\text{TIN}}_n/\overline{\text{TOUT}}_n$ not being pinned out.
5.8.5.4/5-66	Removed the word 'non-functional' from the introduction text to Figure 5-53 , “Global Timers Capture Registers (GTCPR1–GTCPR4).”
5.8.5.6/5-67	In Table 5-75 , “GTEVR _n Bit Settings,” changed “reached” to “exceeded” for the “1” encoding in the REF (bit 14) field description.
5.8.6.1/5-69	In the third paragraph, updated the sentence from: “The maximum period (when the reference value is all ones and the prescaler divides by 256) for one 16-bit timer is ~50 ms at 333 MHz” to: “The maximum period (when the reference value is all ones and the prescaler divides by 256) for one 16-bit timer is ~206 s at 333 MHz”
5.8.6.1/5-69	In the second paragraph following the bulleted list, changed 65,537 to 65,536.
5.8.6.3/5-69	In the Note, changed “the counter begins counting after...” to “the counter begins or stops after...”
5.9.2.3/5-75	In note underneath Table 5-81 , “PMCMR Bit Settings,” changed “SIMR_L[PMC]” to “SIMSR_L[PMC].”

Revision History

- 6.2.1/6-2 Reserved fields changed from “Write reserved, read = 0” to “Reserved, write should preserve reset value.”
- 6.2.6/6-8 In [Figure 6-6](#), “Arbiter Event Attributes Register (AEATR),” changed register access “Read/Write” to “Read-only.”
- 6.2.7/6-9 In [Figure 6-7](#), “Arbiter Event Address Register (AEADR),” changed register access “Read/Write” to “Read-only.”
- 6.3.1.3/6-13 Changed “After the completion of snoop copyback, the arbiter grants the bus back to the master that had its transaction ARTRYed” to:
 “After the completion of snoop copyback, the arbiter grants the bus to the most ahead master among those masters which have an active bus request signal at that time, which may or may not be the same master that had its transaction ARTRYed. Only when a transaction address phase is completed with no ARTRY (and no repeat conditions), the master moves to the end of the line.”
- 7.1.7.2/7-11 Changed “time base enable (*tben*) signal” to say “time base/decrementer clock base enable (*tben*) signal.”
- 7.3.1.3.3/7-18 In [Table 7-2](#), “e300 HID0 Bit Descriptions,” added the following note to EBA and EBD (bits 2 and 3, respectively) field descriptions: “Do not set this bit; the CSB does not have parity signals.”
- 7.3.1.3.3/7-18 Changed “[Table 7-3](#) shows how HID1[ECLK] and HID1[SBCLK] are used to configure the *clk_out* signal” to say:
 “[Table 7-3](#) shows how HID0[ECLK] and HID0[SBCLK] are used to configure the *clk_out* signal.”
- 7.3.4.2/7-30 In [Table 7-7](#), “Exceptions and Interrupts,” changed “DEC[31]” to “DEC[0]” in exception conditions for “Decrementer.”
- 8.1/8-1 Replaced last bullet on page with “External pins (IRQ[0:7]).”
- 8.1/8-1 Replaced next-to-last paragraph on page with the following:
 “The IPIC also manages an internal non-maskable machine-check processor (*mcp*) signal and the interrupt generated by the off-chip interrupt sources (IRQ[0:7]).”
- 8.2/8-4 Changed the seventh bullet to begin “Two programmable...”
- 8.4.1/8-5 Changed first sentence of section to the following:
 “The device has 8 distinct external interrupt request input signals (IRQ[0:7]) and one interrupt request output signal (PCI_INTA).”
- 8.4.1/8-5 In the first row of [Table 8-1](#), “IPIC Signal Properties,” changed entries in both Name and Port columns to read “IRQ[0:7].”
- 8.4.2/8-5 In the first row of [Table 8-2](#), “IPIC External Signals—Detailed Signal Descriptions,” changed the Signal column to read “IRQ[0:7]” and updated its State Meaning description.

8.5/8-6	In Table 8-3 , “IPIC Register Address Map,” changed Reset Value column of System error mask register (SERMR) at offset 0x44 to show only “0xFF00_0000.”
8.5.2/8-9	Removed second paragraph of note (begins with “Note that IVECx field....”).
8.5.8/8-18	Updated bit field lengths in Table 8-17 , “SEPNR Field Descriptions.”
8.5.11/8-20	Updated bit field lengths in Table 8-20 , “SEMSR Field Descriptions.”
8.5.12/8-21	Updated bit field lengths in Table 8-21 , “SECNR Field Descriptions.”
8.5.17/8-26	Updated bit field lengths in Table 8-28 , “SEFCR Field Descriptions.”
8.5.19/8-27	Removed third paragraph of section (begins with words “Note that the CVEC _x field specifies....”).
8.5.20/8-27	Removed third paragraph of section (begins with words “Note that MVEC _x field specifies....”).
8.6.2/8-31	Removed second and third bulleted items completely (both begin with words “The relative priority of the Reserved, Reserved...”).
8.6.2/8-31	Replaced existing fourth bulleted item with the following: “The relative priority of the UART1, UART2, I2C1, I2C2, and SEC internal interrupt signals can be modified.”
9.3.2.2/9-9	In Table 9-4 , “Clock Signals—Detailed Signal Descriptions,” modified MCKE description.”
9.3.2.3/9-10	Updated cross-reference to “ Section 5.3.2.7, “Debug Configuration.” ”
9.4/9-10	In Table 9-5 , “DDR Memory Controller Memory Map,” added footnote to DDR_IP_REV2—DDR IP block revision 2, reset 0x00nn_00nn.
9.4.1.2/9-12	In Table 9-7 , “CS _n _CONFIG Field Descriptions,” added “or DDR3” to ODT_RD_CFG (bits 9–11) and ODT_WR_CFG (bits 13–15) field descriptions.
9.4.1.7/9-21	In Table 9-12 , “DDR_SDRAM_CFG Field Descriptions,” modified the note for field 8_BE (bit 13).
9.4.1.7/9-21	In Table 9-12 , “DDR_SDRAM_CFG Field Descriptions,” added a new programming requirement for HSE (bit 28) such that this bit should not be set if using automatic calibration.
9.4.1.17/9-31	In Figure 9-18 , “DDR IP Block Revision 2 (DDR_IP_REV2),” changed reset value to match reset value in Table 9-5 , “DDR Memory Controller Memory Map.”
9.4.1.27/9-37	In Table 9-33 , “CAPTURE_ATTRIBUTES Field Descriptions,” added bit field description for TSIZ (bits 5–7), and modified TSRC (bits 11–15) field description.
9.4.1.23/9-34	In Figure 9-24 , “Memory Data Path Read Capture ECC Register (CAPTURE_ECC),” and Table 9-29 , “CAPTURE_ECC Field Descriptions,” extended bit field for ECE from 24:31 to 16:31, and added detailed bit field description after generic ECE statement.

9.5/9-39	<p>Updated: “Bank sizes up to 2 Gbytes (maximum total physical memory size of 4 Gbytes) are supported, providing up to a maximum of 4 Gbytes of DDR main memory.”</p> <p>to: “Bank sizes up to 2 Gbits (maximum total physical memory size of 4 Gbytes) are supported, providing up to a maximum of 4 Gbits of DDR main memory per chip select.”</p>
9.5.6/9-58	Added note after first paragraph clarifying system board requirements when using registered DIMMs.
9.5.11/9-64	<p>Added the following text to the end of the section:</p> <p>“In 32-bit mode, Table 9-49 is split into 2 halves. The first half, consisting of rows 0–31, is used to calculate the ECC bits for the first 32 data bits of any 64-bit granule of data. This always applies to the odd data beats on the DDR data bus. The second half of the table, consisting of rows 32–63, is used to calculate the ECC bits for the second 32 bits of any 64-bit granule of data. This always applies to the even data beats on the DDR data bus.”</p>
9.5.12/9-66	Changed the first sentence of the third paragraph to say the following: “If a multi-bit error is detected for a read, the DDR memory controller logs the error and generates the interrupt, and transfer error acknowledge (TEA) is asserted internally on the CSB bus (if enabled, as described in Section 9.4.1.25 , “ Memory Error Disable (ERR_DISABLE) ”).”
9.6.1/9-68	In Table 9-53 , “Programming Differences between Memory Types,” for ODT_PD_EXIT, changed it to be set to 0001 for DDR1; for FOUR_ACT, changed it to be set for 00001 for DDR1.
10.4.2.3/10-47	Added Figure 10-33 , “External Termination of GPCM Access (PLL Bypass Mode).”
10.4.4/10-60	Added the following sentence to end of first paragraph: “A gap of 2 dead LCLK cycles is present on the UPM interface between UPM transactions.”
10.4.4.2/10-64	<p>Add the following text to end of section, immediately before section 10.4.4.2.1:</p> <p>“For proper signalling, the following guidelines must be followed while programming UPM RAM words:</p> <ul style="list-style-type: none"> • For UPM reads, program UTA and LAST in the same or consecutive RAM words.”
10.4.4.4.1/10-67	In Table 10-30 , “UPM RAM Word Field Descriptions,” added note “AMX must not changed values in any RAM word which begins a loop” to “LOOP” and “AMX” descriptions.
10.4.4.4.7/10-73	Added the following note to the end of the section: “AMX must not change values in any RAM word which begins a loop.”
10.4.4.4.10/10-74	<p>Added the following text:</p> <p>“Setting the WAEN bit for the first RAM word does not have any effect since the first RAM word signifies the start of a new bus cycle and the initial values of the signals driven onto the bus should be present.”</p>

10.5.5.3/10-100	In paragraph beginning, “The remaining issue is the synchronization of the UPM cycles...” changed parenthetical in final sentence from “(GPL[0:4] are 1 when inactive, GPL5 is 0 when inactive)” to “(LGPL n are 1 when inactive).”
11.4.1/11-3	In Table 11-2 , “POTAR n Field Descriptions,” updated TA field description.
12.4.1/12-4	In Figure 12-2 , “Outbound Message Interrupt Status Register (OMISR),” changed bits 0 and 1 to “w1c” and register access to “Mixed.”
12.4.2/12-5	Changed sentence “OMIMR can be read from the CSB or the PCI bus, but it can be cleared only from the PCI bus” to say “OMIMR can be read from the CSB or the PCI bus, but it can be written only from the PCI bus.”
12.4.8.1/12-10	In Table 12-11 , “DMAMR n Field Descriptions,” modified PRC bits 11–10) and TEM (bit 3) field descriptions. In addition, removed the note “The address hold feature is not supported when external hardware control is used” from DAHE and SAHE (bits 13 and 12, respectively) field descriptions.
12.4.8.2/12-12	In Table 12-12 , “DMASR n Field Descriptions,” modified CB (bit 2) and TE (bit 7) field descriptions.
12.5.3/12-17	Updated paragraph beginning “Accesses to CSB memory depend...”.
13.3.2.11/13-22	Added the following text to the end of the PITAR n description: “Inbound and outbound windows for the same bus should not overlap. Therefore, situations where an inbound window translation points back into an outbound window, or where an outbound translation window points back into an inbound window, are not allowed.”
13.3.2.11/13-22	In Table 13-18 , “PITAR n Field Descriptions,” updated TA field description.
13.4.2/13-43	In Table 13-43 , “PCI Command Definitions,” added the following note to the Memory write command (0b0111) definition: “Note that for inbound writes less than 4-bytes, the PCI controller splits the transaction into single byte writes to the target. Thus, the PCI interface cannot be used to perform single beat writes to 16-bit devices on the local peripheral interfaces.”
13.4.8/13-57	Added section with Section 13.4.8 , “Byte Ordering.”
14.5.6.9.1/14-78	Updated the bullets.
15.3.1.5/15-9	In Table 15-8 , “I2CnDR Field Descriptions,” in DATA description, changed last sentence to say “Note that in both master receive and slave receive modes, the very first read is always a dummy read.”
15.5.5/15-23	Removed sentence “For 1-byte transfers, a dummy read should be performed by the interrupt service routine (see Figure 15-11).”
16.3.1.3/16-7	Changed calculating percent error value step 1 calculation from “AFI = baud rate \times 16” to “AFI = baud rate \times 16 \times divisor”
Part IV	Removed QUICC Engine chapters from MPC8360ERM. See <i>QUICC Engine Block Reference Manual with Protocol Interworking (QEIWRM)</i> , Rev. 3.

Chapter 19, “QUICC Engine Block on the MPC8360E,” serves as a general overview.

B.2 Changes From Revision 1 to Revision 2

Section, Page No.

Changes

- Global The EFM chapter will be removed in future reference manuals. This functionality will be supported in the future via a RAM-based microcode package.
- 2.3/2-1 Added the following after the third paragraph:
Unless stated otherwise in a particular block, all accesses to and from the memory mapped registers must be made with 32-bit accesses. There is no support for accesses of sizes other than 32 bits.
- 2-3, 2-2 Added global timer blocks to [Table 2-1](#), “IMMR Memory Map,” and [Table 2-2](#), “Memory Map,” as shown below. Note that the cross-references refer to this document only and are subject to change in the next revision of the document.

Table 2-1. IMMR Memory Map

Address	Use	Actual Size	Window	Cross Reference
0x00_0500–0x00_05FF	Global timers module 1	64 bytes	256 bytes	Table 2-2
0x00_0600–0x00_06FF	Global timers module 2	64 bytes	256 bytes	Table 2-2

Table 2-2. Memory Map

Offset	Register	Access	Reset	Section/Page
Global Timers Module 1				
0x0_0500	GTCFR1—Timer 1 and 2 global timers configuration register	R/W	0x00	5.8.5.1/5-10
0x0_0501–0x0_0503	Reserved, should be cleared	—	—	—
0x0_0504	GTCFR2—Timer 3 and 4 global timers configuration register	R/W	0x00	5.8.5.1/5-10
0x0_0505–0x0_050F	Reserved, should be cleared	—	—	—
0x0_0510	GTMDR1—Timer 1 global timers mode register	R/W	0x0000	5.8.5.2/5-14
0x0_0512	GTMDR2—Timer 2 global timers mode register			
0x0_0514	GTRFR1—Timer 1 global timers reference register	R/W	0x0000	5.8.5.3/5-15
0x0_0516	GTRFR2—Timer 2 global timers reference register			
0x0_0518	GTCPR1—Timer 1 global timers capture register	R/W	0x0000	5.8.5.4/5-15
0x0_051A	GTCPR2—Timer 2 global timers capture register			
0x0_051C	GTCNR1—Timer 1 global timers counter register	R/W	0x0000	5.8.5.5/5-16
0x0_051E	GTCNR2—Timer 2 global timers counter register			
0x0_0520	GTMDR3—Timer 3 global timers mode register	R/W	0x0000	5.8.5.2/5-14
0x0_0522	GTMDR4—Timer 4 global timers mode register			

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0524	GTRFR3—Timer 3 global timers reference register	R/W	0x0000	5.8.5.3/5-15
0x0_0526	GTRFR4—Timer 4 global timers reference register			
0x0_0528	GTCPR3—Timer 3 global timers capture register	R	0x0000	5.8.5.4/5-15
0x0_052A	GTCPR4—Timer 4 global timers capture register			
0x0_052C	GTCNR3—Timer 3 global timers counter register	R/W	0x0000	5.8.5.5/5-16
0x0_052E	GTCNR4—Timer 4 global timers counter register			
0x0_0530	GTEVR1—Timer 1 global timers event register	Special	0x0000	5.8.5.6/5-16
0x0_0532	GTEVR2—Timer 2 global timers event register			
0x0_0534	GTEVR3—Timer 3 global timers event register			
0x0_0536	GTEVR4—Timer 4 global timers event register			
0x0_0538	GTPSR1—Timer 1 global timers prescale register	R/W	0x0003	5.8.5.7/5-17
0x0_053A	GTPSR2—Timer 2 global timers prescale register			
0x0_053C	GTPSR3—Timer 3 global timers prescale register			
0x0_053E	GTPSR4—Timer 4 global timers prescale register			
Global Timers Module 2				
0x0_0600	GTCFR1—Timer 1 and 2 global timers configuration register	R/W	0x00	5.8.5.1/5-10
0x0_0601– 0x0_0603	Reserved, should be cleared	—	—	—
0x0_0604	GTCFR2—Timer 3 and 4 global timers configuration register	R/W	0x00	5.8.5.1/5-10
0x0_0605– 0x0_060F	Reserved, should be cleared	—	—	—
0x0_0610	GTMDR1—Timer 1 global timers mode register	R/W	0x0000	5.8.5.2/5-14
0x0_0612	GTMDR2—Timer 2 global timers mode register			
0x0_0614	GTRFR1—Timer 1 global timers reference register	R/W	0x0000	5.8.5.3/5-15
0x0_0616	GTRFR2—Timer 2 global timers reference register			
0x0_0618	GTCPR1—Timer 1 global timers capture register	R/W	0x0000	5.8.5.4/5-15
0x0_061A	GTCPR2—Timer 2 global timers capture register			
0x0_061C	GTCNR1—Timer 1 global timers counter register	R/W	0x0000	5.8.5.5/5-16
0x0_061E	GTCNR2—Timer 2 global timers counter register			
0x0_0620	GTMDR3—Timer 3 global timers mode register	R/W	0x0000	5.8.5.2/5-14
0x0_0622	GTMDR4—Timer 4 global timers mode register			
0x0_0624	GTRFR3—Timer 3 global timers reference register	R/W	0x0000	5.8.5.3/5-15
0x0_0626	GTRFR4—Timer 4 global timers reference register			
0x0_0628	GTCPR3—Timer 3 global timers capture register	R	0x0000	5.8.5.4/5-15
0x0_062A	GTCPR4—Timer 4 global timers capture register			

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_062C	GTCNR3—Timer 3 global timers counter register	R/W	0x0000	5.8.5.5/5-16
0x0_062E	GTCNR4—Timer 4 global timers counter register			
0x0_0630	GTEVR1—Timer 1 global timers event register	Special	0x0000	5.8.5.6/5-16
0x0_0632	GTEVR2—Timer 2 global timers event register			
0x0_0634	GTEVR3—Timer 3 global timers event register			
0x0_0636	GTEVR4—Timer 4 global timers event register			
0x0_0638	GTPSR1—Timer 1 global timers prescale register	R/W	0x0003	5.8.5.7/5-17
0x0_063A	GTPSR2—Timer 2 global timers prescale register			
0x0_063C	GTPSR3—Timer 3 global timers prescale register			
0x0_063E	GTPSR4—Timer 4 global timers prescale register			

- 3.1, 3-2 In Figure 3-1, made IRQ0/MCP_IN input only.
- 4.3.2.1/4-12 In [Table 4-8](#), “Reset Configuration Word Low Bit Settings,” changed DDRCM bit description from
 “The 2:1 mode is useful mostly when for a 32-bit data bus memory device.”
 to
 “The 2:1 mode is useful mostly when for a 32-bit data bus width.”
- 4.3.2.2/4-16 In [Figure 4-4](#), “Reset Configuration Word High Register (RCWHR),” changed bit 29 (LALE) to be ‘Reserved’.
- 4.3.2.2/4-16 In [Table 4-12](#), “Reset Configuration Word High Bit Settings,” changed bit 29 (LALE) to be ‘Reserved’.
- 4.3.3.2.1, 4-24 Replaced the note with the following:
- NOTE**
- When reset configuration words are loaded from an I²C EEPROM, an I²C serial EEPROM of extended addressing type must be used.
- 4.3.3.3.2, 4-28 In [Table 4-24](#), changed the Name from ‘Reserved, should be cleared’ to Reserved’ for bits 1, 12-15, 16-19, and 20-27.
- 4.3.3.3.2/4-28 Reformatted [Table 4-24](#), “Hard-Coded Reset Configuration Word High Field Values.”
- 4.4.3/4-31 Removed overbar from the two instances of CFG_CLKIN_DIV.
- 4.4.3, 4-32 Removed the first row (I²C1) from [Table 4-26](#), “Configurable Clock Units.”
- 4.5.3/4-40 In [Table 4-37](#), “Clock Control DDR,” added a footnote for offset 0x10.
- 5.4.3.4/5-27 In [Table 5-39](#), “SPCR Bit Settings,” added note to PCIPR (bits 6-7):
 “DMA has the same priority as PCI.”
- 5.4.3.8/5-33 In [Table 5-43](#), “DDRCDR Field Descriptions,” revised field description bit settings for DDRCDR[DSO_NZ] and DDRCDR[DSO_PZ].

5.5.3, 5-37 Replace text of section (up to 5.5.4) with the following:

The WDT unit can operate in the following modes:

- WDT enable/disable mode:

If the software watchdog timer is not needed, the user can disable it. The 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)

- 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 the value of SWCRR[SWRI], the 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 hard 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 the SWCRR[SWPR] bit that controls the divide-by-65,536 of the WDT counter.

— Prescale mode (SWCRR[SWPR] = 1)

The WDT clock is prescaled.

— Non-prescale mode (SWCRR[SWPR] = 0)

The WDT clock is not prescaled.

5.8, 5-57 Added new [Section 5.8, “Global Timers,”](#) as described below. Note that any cross-references refer to this document only and are subject to change in the next revision of the reference manual.

5.8 General-Purpose Timers (GTM)

The following sections describe theory of operation of the two general purpose (global) timer modules, including a definition of the external signals and the functionality. Additionally, the configuration, control, and status registers are described. Note that individual chapters in this book describe additional specific initialization aspects for each individual block.

5.8.1 Overview

Each global timer module (GTM) includes four identical 16-bit general-purpose timers, two 32-bit timers or one 64-bit timer. Each GTM timer consists of a timer prescale register (GTPSR), a timer mode register

(GTMDR), a timer capture register (GTCPR), a timer counter register (GTCNR), a timer reference register (GTRFR), a timer event register (GTEVR), and a timer global configuration register (GTCFR). The GTPSRs and the GTMDRs contain the primary and secondary prescalers, programmed by the user.

Note that while the MPC8360E has two global timer modules, signals `GTM2_TOUT2` and `GTM2_TOUT4` are not available externally.

Figure 5-1 shows the functional GTM block diagram.

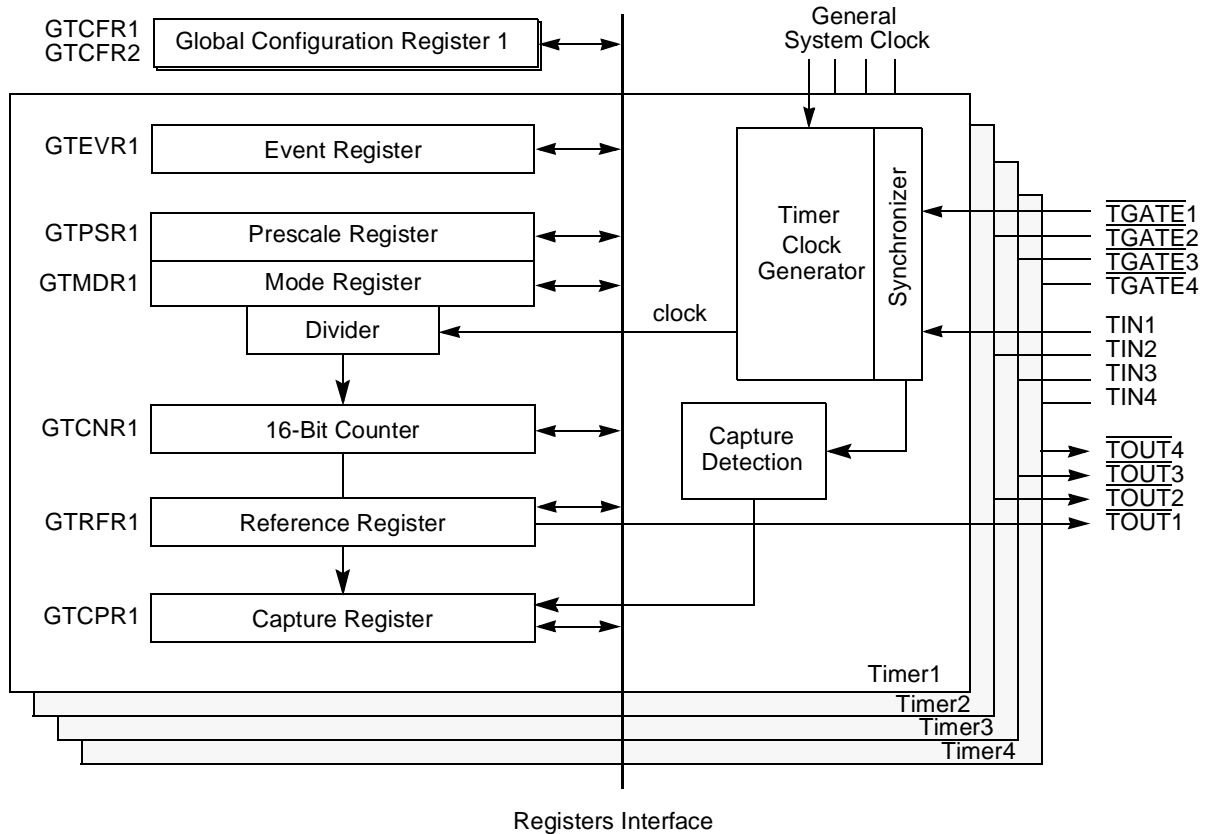


Figure 5-1. Global Timers Block Diagram

5.8.2 Features

The key features of the timer include the following:

- The maximum input clock is the system bus clock
- Four 16-bit programmable timers
- Two timers cascaded internally or externally to form a 32-bit timer
- One timer cascaded internally or externally to form a 64-bit timer
- Maximum period of ~50 msecond (at 333 MHz bus clock) for 16-bit timer
- Maximum period of ~12.8 second (at 333 MHz bus clock) for 32-bit timer
- Maximum period of thousand of year (at 333 MHz bus clock) for 64-bit timer
- 3-nanosecond timer resolution (at 333 MHz bus clock)

- Three programmable input clock sources for the timer prescalers
- Input capture capability
- Output compare with programmable mode for the output pin
- Free run and restart modes
- Functional and programming compatibility with MPC8260 timers

5.8.3 Modes of Operation

The GTM unit can operate in the following modes:

5.8.3.1 Cascaded Modes

$GTCFR_n[PCAS]$ and $GTCFR_2[SCAS]$ are used to put the timers into different cascaded modes:

- Non-cascaded mode: Each timer (timer 1, timer 2, timer 3 and timer 4), function as a independent 16-bit timer with a 16-bit GTRFR, GTCPR, GTMDR and GTCNR. In this mode, the non-cascaded GTRFR, GTCPR, and GTCNR should be referenced with corresponding 16-bit bus cycles.
- Pair-cascaded mode: In this mode, two 16-bit timers can be internally cascaded to form a 32-bit counter: timer 1 can be internally cascaded to timer 2 and timer 3 may be internally cascaded to timer 4. Because the decision to cascade timers is made independently, the user has the option of selecting two 16-bit timers and one 32-bit timer, or two 32-bit timers. When working in the pair-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with 32-bit bus cycles.
- Super-cascaded mode: In this mode, all four 16-bit timers can be internally cascaded to form a 64-bit counter. When working in the super-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with two 32-bit bus cycles.

5.8.3.2 Clock Source Modes

The clock input to the timer's prescaler can be selected from three sources:

- The system clock
- The system slow go clock (system bus clock internally divided by 16)
- The corresponding TINx pin

5.8.3.3 Reference Modes

Each timer can be configured to count until a reference is reached and then either begin a new time count immediately or continue to run. The FRR bit of the corresponding GTMRR selects each mode.

- Free run reference mode. The corresponding timer count continues to increment after the reference value is reached.
- Reset reference mode. The corresponding timer count is reset immediately after the reference value is reached.

5.8.3.4 Capture Modes

Each timer has a 16-bit field in GTCPR, used to latch the value of the counter when a defined transition of TINx is sensed by the corresponding input capture edge detector.

- Normal gate mode enables the count on a falling edge of the $\overline{\text{TGATE}}$ pin and disables the count on the rising edge of $\overline{\text{TGATE}}$. This mode allows the timer to count conditionally, based on the state of $\overline{\text{TGATE}}$.
- The restart gate mode performs the same function as normal mode, except it also resets the counter on the falling edge of the $\overline{\text{TGATE}}$ pin. This mode has applications in pulse interval measurement and bus monitoring.

5.8.4 External Signal Description

The following sections provide an overview and detailed descriptions of the GTM signals.

5.8.4.1 Overview

There are four distinct external input timer capture signals (TIN1, TIN2, TIN3 and TIN4), four distinct external input timer get signals ($\overline{\text{TGATE1}}$, $\overline{\text{TGATE2}}$, $\overline{\text{TGATE3}}$ and $\overline{\text{TGATE4}}$), and four distinct external timer output signals ($\overline{\text{TOUT1}}$, $\overline{\text{TOUT2}}$, $\overline{\text{TOUT3}}$ and $\overline{\text{TOUT4}}$). The GTM interface signals are defined in [Table 5-1](#).

Table 5-1. GTM Signal Properties

Name	Port	Function	I/O	Reset	Require Pull Up
TIN1	TIN1	Global timer 1 capture control signal	I	0	No
TIN2	TIN2	Global timer 2 capture control signal	I	0	No
TIN3	TIN3	Global timer 3 capture control signal	I	0	No
TIN4	TIN4	Global timer 4 capture control signal	I	0	No
$\overline{\text{TGATE1}}$	$\overline{\text{TGATE1}}$	Global timer 1 counter gate control signal	I	0	No
$\overline{\text{TGATE2}}$	$\overline{\text{TGATE2}}$	Global timer 2 counter gate control signal	I	0	No
$\overline{\text{TGATE3}}$	$\overline{\text{TGATE3}}$	Global timer 3 counter gate control signal	I	0	No
$\overline{\text{TGATE4}}$	$\overline{\text{TGATE4}}$	Global timer 4 counter gate control signal	I	0	No
$\overline{\text{TOUT1}}$	$\overline{\text{TOUT1}}$	Global timer 1 counter output signal	O	1	No
$\overline{\text{TOUT2}}$	$\overline{\text{TOUT2}}$	Global timer 2 counter output signal	O	1	No
$\overline{\text{TOUT3}}$	$\overline{\text{TOUT3}}$	Global timer 3 counter output signal	O	1	No
$\overline{\text{TOUT4}}$	$\overline{\text{TOUT4}}$	Global timer 4 counter output signal	O	1	No

5.8.4.2 Detailed Signal Descriptions

Table 5-2 provides detailed descriptions of the external GTM signals.

Table 5-2. GTM External Signals—Detailed Signal Descriptions

Signal	I/O	Description
TIN _n	I	Global timer capture control signal. Used to latch the value of the counter when a defined transition of TIN _n is sensed by the corresponding input capture edge detector.
		State Meaning Asserted/Negated—According to the programmed polarity by the corresponding GTMDR _n [CE]. Each timer has a 16-bit GTCPR used to latch the value of the counter when a defined transition of TIN _n is sensed by the corresponding input capture edge detector. Upon a capture or reference event, the corresponding GTEVR bit is set and a maskable interrupt request is issued to the interrupt controller.
		Timing Assertion/Negation—Asynchronous to internal bus clock. TIN _n is internally synchronized to the system bus clock. If TIN _n meets the asynchronous input setup time, the value of counter is captured after one system bus clock when working with the internal clock.
$\overline{\text{TGATE}}_n$	I	Global timer counter gate control signal. Used to gate/restart the counter when a defined transition of $\overline{\text{TGATE}}_n$ is sensed by the corresponding input capture edge detector.
		State Meaning Asserted/Negated—According to the programmed polarity by the corresponding GTCFR[GMx] bits. In a reset gate mode (GTCFR[GM _n] = 0), the $\overline{\text{TGATE}}_n$ pin is used to enable/disable count. A falling $\overline{\text{TGATE}}_n$ pin enables and restarts the count and a rising edge of $\overline{\text{TGATE}}_n$ disables the count. In a normal gate mode (GTCFR[GM _n] = 1), the $\overline{\text{TGATE}}_n$ have similar functionality, except the falling edge of $\overline{\text{TGATE}}_n$ does not restart the appropriate count value in GTCNR _n [CNV _n].
		Timing Assertion/Negation—Asynchronous to internal bus clock. $\overline{\text{TGATE}}_n$ is internally synchronized to the system bus clock. If $\overline{\text{TGATE}}_n$ meets the asynchronous input setup time, the counter begins counting after one system bus clock when working with the internal clock.
$\overline{\text{TOUT}}_n$	O	Global timer counter output signal. The GTM output a signal on the timer output pin $\overline{\text{TOUT}}_n$ when the reference value is reached.
		State Meaning Asserted/Negated—According to the programmed polarity by the corresponding GTMDR _n [OM _n]. <ol style="list-style-type: none"> Active-low pulse on $\overline{\text{TOUT}}_n$ for one timer input clock cycle as defined by the GTMDR_n[ICLK_n] bits (GTMDR_n[OM_n] = 1). Thus, $\overline{\text{TOUT}}_n$ may be low for one general system clock period, one general system slow go clock period, or one TIN_n pin clock cycle period. Toggle the $\overline{\text{TOUT}}_n$ pin (GTMDR_n[OM_n] = 0). $\overline{\text{TOUT}}_n$ changes occur on the rising edge of the system clock.
		Timing Assertion/Negation— $\overline{\text{TOUT}}_n$ changes occur on the rising edge of the system clock.

5.8.5 Memory Map/Register Definition

The GTM programmable register map occupies 64 bytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All GTM registers are 8 or 16 bits wide, located on 8-bit or 16-bit address boundaries, and should only be accessed as 8-bit or 16-bit quantities. All addresses used in this chapter are offsets from GPT Base, as defined in Chapter 2, “Memory Map.”

Table 5-3 shows memory map of the GTM.

Table 5-3. GTM Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00	Timer 1 and 2 global timers configuration register (GTCFR1)	R/W	0x00	5.8.5.1/5-10
0x01–0x03	Reserved	—	—	—
0x04	Timer 3 and 4 global timers configuration register (GTCFR2)	R/W	0x00	5.8.5.1/5-10
0x05–0x0F	Reserved	—	—	—
0x10	Timer 1 global timers mode register (GTMDR1)	R/W	0x0000	5.8.5.2/5-14
0x12	Timer 2 global timers mode register (GTMDR2)			
0x14	Timer 1 global timers reference register (GTRFR1)	R/W	0xFFFF	5.8.5.3/5-15
0x16	Timer 2 global timers reference register (GTRFR2)			
0x18	Timer 1 global timers capture register (GTCPR1)	R/W	0x0000	5.8.5.4/5-15
0x1A	Timer 2 global timers capture register (GTCPR2)			
0x1C	Timer 1 global timers counter register (GTCNR1)	R/W	0x0000	5.8.5.5/5-16
0x1E	Timer 2 global timers counter register (GTCNR2)			
0x20	Timer 3 global timers mode register (GTMDR3)	R/W	0x0000	5.8.5.2/5-14
0x22	Timer 4 global timers mode register (GTMDR4)			
0x24	Timer 3 global timers reference register (GTRFR3)	R/W	0xFFFF	5.8.5.3/5-15
0x26	Timer 4 global timers reference register (GTRFR4)			
0x28	Timer 3 global timers capture register (GTCPR3)	R	0x0000	5.8.5.4/5-15
0x2A	Timer 4 global timers capture register (GTCPR4)			
0x2C	Timer 3 global timers counter register (GTCNR3)	R/W	0x0000	5.8.5.5/5-16
0x2E	Timer 4 global timers counter register (GTCNR4)			
0x30	Timer 1 global timers event register (GTEVR1)	w1c	0x0000	5.8.5.6/5-16
0x32	Timer 2 global timers event register (GTEVR2)			
0x34	Timer 3 global timers event register (GTEVR3)			
0x36	Timer 4 global timers event register (GTEVR4)			
0x38	Timer 1 global timers prescale register (GTPSR1)	R/W	0x0003	5.8.5.7/5-17
0x3A	Timer 2 global timers prescale register (GTPSR2)			
0x3C	Timer 3 global timers prescale register (GTPSR3)			
0x3E	Timer 4 global timers prescale register (GTPSR4)			

5.8.5.1 Global Timers Configuration Registers (GTCFR n)

The global timers configuration registers (GTCFR1 and GTCFR2), shown in [Figure 5-2](#) and [Figure 5-3](#), contain configuration parameters used by the timers. These registers allow simultaneous starting, stopping

and resetting of a pair of timers (1 and 2 or 3 and 4) or of a groups of timers (1, 2, 3, and 4) if one bus cycle is used. GTCFR is cleared by reset.

NOTE

For proper operation of the timers, do not change the modes of operation and enable the timer in the same register write operation. The modes can be changed when $GTCFR_n[RST_n]$ is cleared. However, when $GTCFR_n[RST_n]$ are set, they are the only bits that can be changed.

Offset 0x00

Access: Read/Write

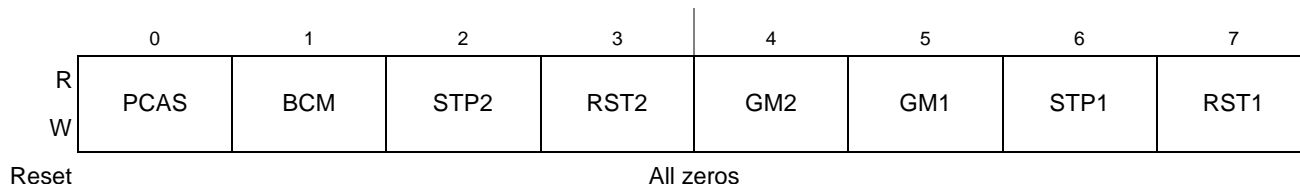


Figure 5-2. Global Timers Configuration Register 1 (GTCFR1)

Table 5-4 defines the bit fields of GTCFR1.

Table 5-4. GTCFR1 Bit Settings

Bits	Name	Description
0	PCAS	Pair-cascade mode 0 Normal operation 1 Timers 1 and 2 cascade to form a 32-bit timer. Note: This bit is ignored in super-cascade mode ($GTCFR2[SCAS]=1$). Note: It is allowed to change the value of this bit only when the corresponding timers are in reset mode. Thus, the user should first clear the RST1 and RST2 bits (without changing PCAS) and then, <u>in a separate write to the register</u> , change the value of PCAS.
1	BCM	Backward compatible mode 0 Provide backward compatibility to PowerQUICC II family timers. In this mode $GTCFR1[GM2]$ bit will control the gate mode for timers 1 and 2 and $GTCFR2[GM4]$ bit will control the gate mode for timers 3 and 4. $GTCFR1[GM1]$ and $GTCFR2[GM3]$ bits are ignored. 1 Normal operational mode
2	STP2	Stop timer 2 0 Normal operation 1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 2, except the Register Interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
3	RST2	Reset timer 2 0 Reset the timer 2, including GTMDR2, GTRFR2, GTCNR2, GTCPR2 and GTEVR2 (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP2 bit is cleared.
4	GM2	Gate mode for $\overline{TGATE2}$ 0 Restart gate mode. The $\overline{TGATE2}$ pin is used to enable/disable count. A low level of $\overline{TGATE2}$ enables and a falling edge of $\overline{TGATE2}$ restarts the count (reset the dynamic counter's count value to 0) and a high level of $\overline{TGATE2}$ disables the count. 1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{TGATE2}$ does not restart the appropriate count value in $GTCNR2[CNV2]$.

Table 5-4. GTCFR1 Bit Settings (continued)

Bits	Name	Description
5	GM1	<p>Gate mode for $\overline{\text{TGATE1}}$</p> <p>0 Restart gate mode. The $\overline{\text{TGATE1}}$ is used to enable/disable count. A low level of $\overline{\text{TGATE1}}$ enables and a falling edge of $\overline{\text{TGATE1}}$ restarts the count (reset the dynamic counter's count value to 0) and a high level of $\overline{\text{TGATE1}}$ disables the count.</p> <p>1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE1}}$ does not restart the appropriate count value in GTCNR1[CNV1].</p> <p>Note: In backward compatible mode (GTCFR1[BCM]=0) this bit is ignored. GTCFR1[GM2] bit will control the gate mode for timers 1 and 2.</p>
6	STP1	<p>Stop timer 1</p> <p>0 Normal operation</p> <p>1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 1, except the register interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.</p>
7	RST1	<p>Reset timer 1</p> <p>0 Reset the timer 1, including GTMDR1, GTRFR1, GTCNR1, GTCPR1 and GTEVR1 (a software reset is identical to an external reset).</p> <p>1 Enable the corresponding timer if the STP1 bit is cleared.</p>

The GTCFR2 register is shown in [Figure 5-3](#).

Offset 0x04

Access: Read/Write

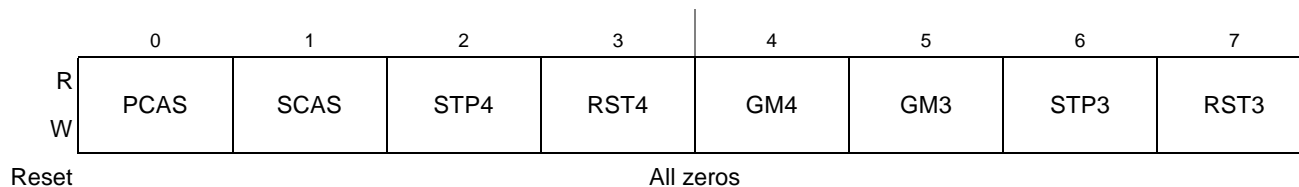


Figure 5-3. Global Timers Configuration Register 2 (GTCFR2)

Table 5-5 defines the bit fields of GTCFR2.

Table 5-5. GTCFR2 Bit Settings

Bits	Name	Description
0	PCAS	Pair-cascade mode 0 Normal operation. 1 Timers 3 and 4 cascade to form a 32-bit timer. Note: This bit is ignored in super-cascade mode (GTCFR2[SCAS]=1). Note: It is allowed to change the value of this bit only when the corresponding timers are in reset mode. Thus, the user should first clear the RST3 and RST4 bits (without changing PCAS) and then, in a separate write to the register, change the value of PCAS.
1	SCAS	Super cascade mode 0 Normal operation 1 Timers 1, 2, 3 and 4 cascade to form a 64-bit timer. Note: In super-cascade mode (GTCFR2[SCAS]=1) the pair-cascade mode bits are ignored, (GTCFR1/2[PCAS]=Don't Care). Note: It is allowed to change the value of this bit only when the corresponding timers are in reset mode. Thus, the user should first clear the RST1, RST2, RST3 and RST4 bits (without changing SCAS) and then, in a separate write to the register, change the value of SCAS.
2	STP4	Stop timer 4 0 Normal operation 1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 4, except the register interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
3	RST4	Reset timer 4 0 Reset the timer 4, including GTMDR4, GTRFR4, GTCNR4, GTCPR4 and GTEVR4 (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP4 bit is cleared.
4	GM4	Gate mode for $\overline{\text{TGATE4}}$ 0 Restart gate mode. The $\overline{\text{TGATE4}}$ is used to enable/disable count. A low level of $\overline{\text{TGATE4}}$ enables and a falling edge of $\overline{\text{TGATE4}}$ restarts the count (reset the dynamic counter's count value to 0) and a high level of $\overline{\text{TGATE4}}$ disables the count. 1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE4}}$ does not restart the appropriate count value in GTCNR4[CNV4].
5	GM3	Gate mode for $\overline{\text{TGATE3}}$ 0 Restart gate mode. The $\overline{\text{TGATE3}}$ is used to enable/disable count. A low level of $\overline{\text{TGATE3}}$ enables and a falling edge of $\overline{\text{TGATE3}}$ restarts the count (reset the dynamic counter's count value to 0) and a high level of $\overline{\text{TGATE3}}$ disables the count. 1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE3}}$ does not restart the appropriate count value in GTCNR3[CNV3]. Note: In backward compatible mode (GTCFR1[BCM]=0) this bit is ignored. The GTCFR2[GM4] bit controls the gate mode for timers 3 and 4.
6	STP3	Stop timer 3 0 Normal operation 1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 3, except the register interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
7	RST3	Reset timer 3 0 Reset the timer 3, including GTMDR3, GTRFR3, GTCNR3, GTCPR3 and GTEVR3 (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP3 bit is cleared.

5.8.5.2 Global Timers Mode Registers (GTMDR1–GTMDR4)

The global timers mode registers (GTMDR1, GTMDR2, GTMDR3 and GTMDR4) are shown in Figure 5-4.

Erratic behavior may occur if GTCFR1 and GTCFR2 are not initialized before the GTMDR n . Only GTCFR n [RST n] and GTCFR n [STP n] can be modified at any time.

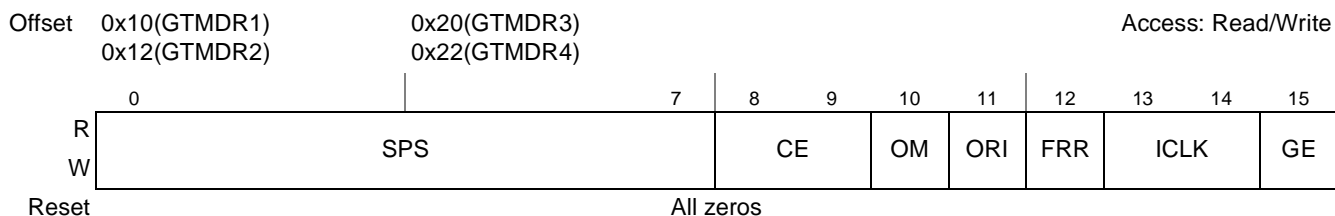


Figure 5-4. Global Timers Mode Registers (GTMDR1–GTMDR4)

Table 5-6 defines the bit fields of GTMDR.

Table 5-6. GTMDR Bit Settings

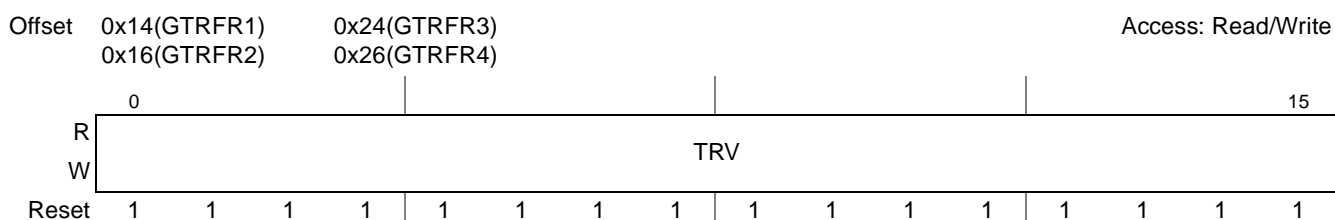
Bits	Name	Description
0–7	SPS	Secondary prescaler value The secondary prescaler is programmed to divide the clock input to corresponding timer by values from 1 to 256. The value 0x00 divides the clock by 1 and 0xFF divides the clock by 256.
8–9	CE	Capture edge and enable interrupt 00 Disable interrupt on capture event; capture function is disabled 01 Capture on rising TIN n edge only and enable interrupt on capture event. 10 Capture on falling TIN n edge only and enable interrupt on capture event. 11 Capture on any TIN n edge and enable interrupt on capture event. Note: The frequency of TIN n should be slower than system clock (TIN n is sampled internally by system clock to detect TIN n 's rising/falling edge before updating the counter)
10	OM	Output mode 0 Toggle $\overline{\text{TOUT}}_n$ every time when the corresponding timer matches its reference value. 1 Active-low pulse on $\overline{\text{TOUT}}_n$ for one timer input clock cycle (4 input clock cycles for the system clock) as defined by the ICLK n bits. Thus, $\overline{\text{TOUT}}_n$ may be low for four general system clocks, one general system slow go clock period, or one TIN n pin clock cycle period. Note: $\overline{\text{TOUT}}_n$ changes are internally synchronized to the rising edge of the system clock
11	ORI	Output reference interrupt enable 0 Disable interrupt for reference reached (does not affect interrupt on capture function). 1 Enable interrupt upon reaching the reference value.
12	FRR	Free run/restart mode 0 Free run. The timer count continues to increment after the reference value is reached. 1 Restart. The timer count is reset immediately after the reference value is reached.

Table 5-6. GTMDR Bit Settings (continued)

Bits	Name	Description
13–14	ICLK	Input clock source for the timer. 00 Internally cascaded input. This selection means: For ICLK1, the timer 1 input is the output of timer 2; For ICLK2, the timer 1 input is the output of timer 2, the timer 2 input is the output of timer 3, the timer 3 input is the output of timer 4; For ICLK3, the timer 3 input is the output of timer 4; For ICLK4 this selection means no input clock is provided to the timer. 01 Internal general system bus clock. 10 Internal slow go clock (divided by 16 system bus clock). 11 TIN n : corresponding TIN1, TIN2, TIN3 or TIN4 pin (falling edge).
15	GE	Gate enable 0 The $\overline{\text{TGATE}}_n$ signal is ignored. 1 The $\overline{\text{TGATE}}_n$ signal is used to control the timer.

5.8.5.3 Global Timers Reference Registers (GTRFR1–GTRFR4)

Global timers reference registers, shown in [Figure 5-5](#), are 16-bit memory-mapped, read/write registers containing the 16-bit reference values for each timer’s timeout. The reference value is not reached until $\text{GTCNR}_n[\text{CNV}]$ increments to the value in $\text{GTRFR}_n[\text{TRV}]$.


Figure 5-5. Global Timers Reference Registers (GTRFR1–GTRFR4)

[Table 5-7](#) defines the bit fields of GTRFR.

Table 5-7. GTRFR Bit Settings

Bits	Name	Description
0–15	TRV	Timeout reference value. 16-bit timeout reference value for the corresponding timer. Set to all ones by reset.

5.8.5.4 Global Timers Capture Registers (GTCPR1–GTCPR4)

Global timers capture registers (GTCPR_1 , GTCPR_2 , GTCPR_3 and GTCPR_4), shown in [Figure 5-6](#), are used to latch the value of the counters according to $\text{GTMDR}_n[\text{CE}]$.

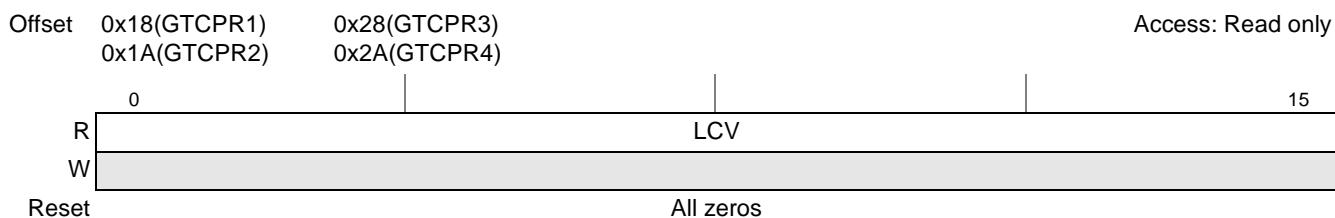

Figure 5-6. Global Timers Capture Registers (GTCPR1–GTCPR4)

Table 5-8 defines the bit fields of $GTCPR_n$.

Table 5-8. $GTCPR_n$ Bit Settings

Bits	Name	Description
0–15	LCV	Latched counter value. Corresponding timer's 16-bit latched value.

5.8.5.5 Global Timers Counter Registers ($GTCNR1$ – $GTCNR4$)

Global timers counter registers ($GTCNR1$, $GTCNR2$, $GTCNR3$ and $GTCNR4$), shown in Figure 5-7, are four 16-bit, memory-mapped, read/write up-counters. A read cycle to a $GTCNR_n[CNV]$ fields yields the current value of the appropriate timer but does not affect the counting operation. A write cycle to a $GTCNR_n[CNV]$ field sets the register to the written value, causing its corresponding primary and secondary prescaler counters to be reset.

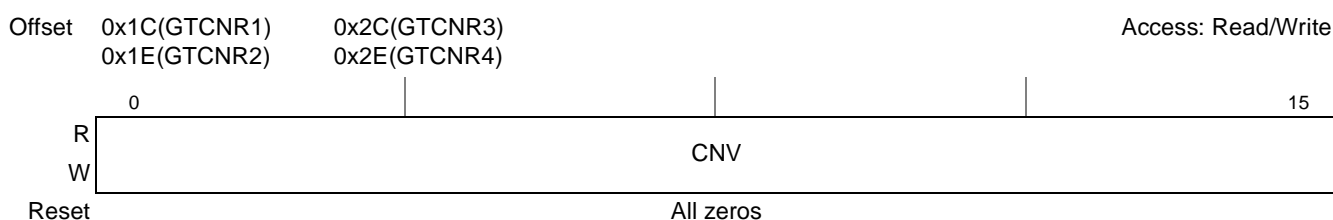


Figure 5-7. Global Timers Counter Registers ($GTCNR1$ – $GTCNR4$)

Table 5-9 defines the bit fields of $GTCNR$.

Table 5-9. $GTCNR$ Bit Settings

Bits	Name	Description
0–15	CNV	Counter value. Corresponding timer's 16-bit read/write up-counter value.

5.8.5.6 Global Timers Event Registers ($GTEVR1$ – $GTEVR4$)

Global timers event registers ($GTEVR1$, $GTEVR2$, $GTEVR3$ and $GTEVR4$), shown in Figure 5-8, are used to report events recognized by any of the timers. On recognition of an output reference event, the appropriate timer sets $GTEVR_n[REF]$, regardless of the corresponding $GTMDR_n[ORI]$. The capture event is only set if it is enabled by $GTMDR_n[CE]$. $GTEVRs$ appear as memory-mapped registers to users, which can be read at any time.

$GTEVR_n$ bits are cleared by writing ones to them (writing zeros does not affect bit values). Both bits must be reset before the timer negates the interrupt to the interrupt controller.

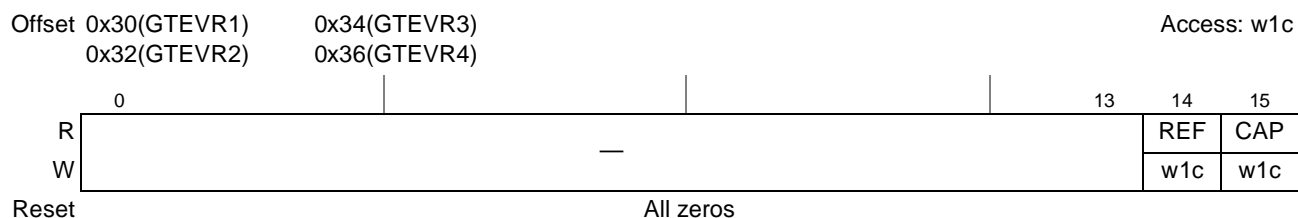


Figure 5-8. Global Timers Event Registers ($GTEVR1$ – $GTEVR4$)

Table 5-10 defines the bit fields of $GTEVR_n$.

Table 5-10. $GTEVR_n$ Bit Settings

Bits	Name	Description
0–13	—	Reserved, should be cleared.
14	REF	Output reference event 0 No event 1 The counter reached the $GTRFR_n[TRV]$ value. $GTMDR_n[ORI]$ is used to enable the interrupt request caused by this event.
15	CAP	Counter capture event Corresponding timer's 16-bit read/write up-counter value. 0 No event 1 The counter value has been latched into the $GTCPR_n[LCV]$. $GTMDR_n[CE]$ is used to enable generation of this event.

5.8.5.7 Global Timers Prescale Registers (GTPSR1–GTPSR4)

The global timers prescale registers (GTPSR1, GTPSR2, GTPSR3 and GTPSR4) are shown in Figure 5-9.

Erratic behavior may occur if $GTPSR_n$ is not initialized before the corresponding $GTMDR_n$.

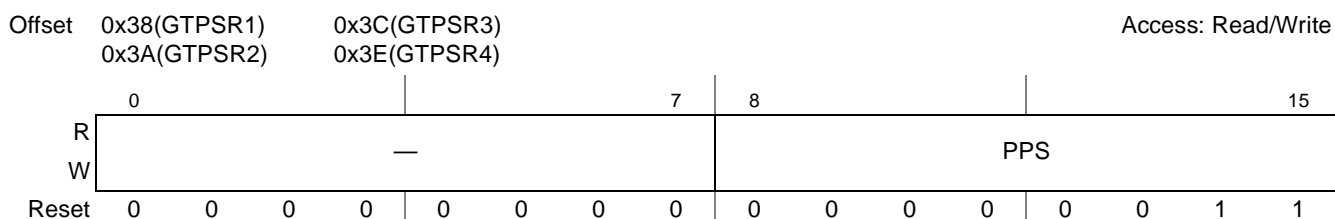


Figure 5-9. Global Timers Prescale Registers (GTPSR1–GTPSR4)

Table 5-11 defines the bit fields of $GTPSR_n$.

Table 5-11. $GTPSR_n$ Bit Settings

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–15	PPS	Primary prescaler bits The primary prescaler is programmed to divide the clock input to corresponding timer by values from 1 to 256. The value 0x00 divides the clock by 1 and 0xFF divides the clock by 256.

NOTE

The total timer prescale value is calculated as follows:

$$GTM_{n_{\text{prescaler}}} = (GTPSR_n[PPS] + 1) \cdot (GTMDR_n[SPS] + 1)$$

This gives a total prescale range from 1 ($GTPSR_n[PPS] = 0x00$, $GTMDR_n[SPS] = 0x00$) to 65,536 ($GTPSR_n[PPS] = 0xFF$, $GTMDR_n[SPS] = 0xFF$).

5.8.6 Functional Description

5.8.6.1 General-Purpose Timer Units

The clock input to the timer's prescaler can be selected from the following sources:

- The system clock (*ipg_clock*)
- The system slow go clock (*ipg_clock* internally divided by 16)

The general system clock is generated in the clock synthesizer and defaults to the system frequency. However, the general system clock has the option to be divided before it leaves the clock synthesizer. This mode, called slow go, is used to save power. Whatever the resulting frequency of the general system clock, the user can either choose that frequency or the frequency divided by 16 as the input to the prescaler of each timer. Alternatively, the user may prefer TIN_n to be the clock source. TIN_n is internally synchronized to the internal clock. If the user has chosen to internally cascade two 16-bit timers to a 32-bit timer, then a timer can use the clock generated by the output of another timer.

The clock input source is selected by the corresponding $GTMDR_n[ICLK]$ bits. The prescalers ($GTMDR_n[SPS]$ and $GTPSR_n[PPS]$) can be programmed to divide the clock input by values from 1 to 65,537 and the output of the prescaler is used as an input to the 16-bit counters. The best resolution of the timer is one clock cycle (3 ns at a 333-MHz system clock, for example). The maximum period (when the reference value is all ones) for one 16-bit timer is ~50 ms at 333-MHz.

5.8.6.2 Reference Modes

Each timer can be configured to count until a reference is reached and then either begin a new time count immediately or continue to run. The FRR bit of the corresponding $GTMRR$ selects each mode.

- Free run reference mode ($GTMDR_n[FRR]=0$)
The corresponding timer count continues to increment after the reference value is reached.
- Reset reference mode ($GTMDR_n[FRR]=1$)
The corresponding timer count is reset immediately after the reference value is reached.

Upon reaching the reference value, the corresponding $GTEVR_n[REF]$ bit is set and an interrupt is issued if $GTMDR_n[ORI] = 1$. The timers can output a signal on the timer output pin \overline{TOUT}_n if the reference value is reached (selected by the corresponding $GTMDR_n[OM]$). This signal can be an active-low pulse or a toggle of the current output. The output can also be connected internally to the input of another timer, resulting in a 32- or 64-bit timer.

5.8.6.3 Capture Modes

In addition, each timer has a 16-bit field in $GTCPR$, used to latch the value of the counter when a defined transition of TIN_n is sensed by the corresponding input capture edge detector. The timers may be gated/restarted by an external gate signals (\overline{TGATE}_n) that controls the timers. The type of transition triggering the capture is selected by the corresponding $GTMDR_n[CE]$ bits. Upon a capture or reference

event, corresponding $GTEVR_n[REF]$ or $GTEVR_n[CAP]$ is set and a maskable interrupt request is issued to the interrupt controller.

- Normal gate mode enables the count on a falling edge of \overline{TGATE} and disables the count on the rising edge of \overline{TGATE} . This mode allows the timer to count conditionally, based on the state of \overline{TGATE} .
- The restart gate mode performs the same function as normal mode, except it also resets the counter on the falling edge of \overline{TGATE} .

This mode has applications in pulse interval measurement and bus monitoring as follows:

- Pulse measurement—The restart gate mode can measure a low pulse on \overline{TGATE} . The rising edge of \overline{TGATE} completes the measurement and if \overline{TGATE}_n is connected externally to TIN_n , it causes the timer to capture the count value and generate a rising-edge interrupt.
- Bus monitoring—The restart gate mode can detect a signal that is stuck abnormally low. The bus signal should be connected to \overline{TGATE} . The timer count is reset on the falling edge of the bus signal and if the bus signal does not go high again within the number of user-defined clocks, an interrupt can be generated.

The gate function is enabled in the $GTMDR$; the gate operating mode is selected in the $GTGCR$.

NOTE

\overline{TGATE} is internally synchronized to the system clock. If \overline{TGATE} meets the asynchronous input setup time, the counter begins counting after one system clock when working with the internal clock.

5.8.6.4 Cascaded Modes

$GTCFR_n[PCAS]$ and $GTCFR2[SCAS]$ are used to put the timers into different cascaded modes:

- Non-cascaded mode ($GTCFR_n[PCAS] = 0$ and $GTGCF2[SCAS] = 0$)
If $GTCFR_n[PCAS] = 0$ and $GTCFR2[SCAS] = 0$, the each timer (timer 1, timer 2, timer 3 and timer 4), function as a independent 16-bit timer with a 16-bit $GTRFR$, $GTCPR$, $GTMDR$ and $GTCNR$ for each one (Figure 5-10). When working in the none-cascaded mode, the non-cascaded $GTRFR$, $GTCPR$, and $GTCNR$ should be referenced with appropriate 16-bit bus cycles.

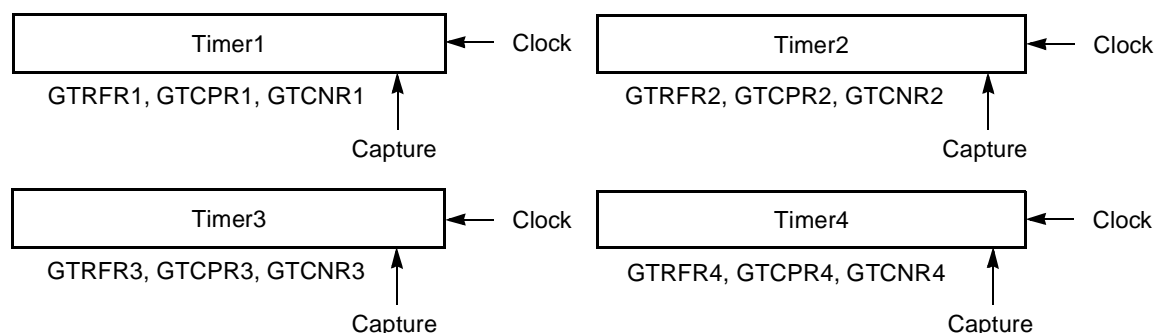


Figure 5-10. Timers Non-Cascaded Mode Block Diagram

- Pair-cascaded mode ($GTCFR1[PCAS] = 1$ and/or $GTCFR2[PCAS] = 1$, $GTCFR2[SCAS] = 0$)
 In this mode, two 16-bit timers can be internally cascaded to form a 32-bit counter: timer 1 may be internally cascaded to timer 2 and timer 3 may be internally cascaded to timer 4, as shown in [Figure 5-11](#). Since the decision to cascade timers is made independently, the user has the option of selecting two 16-bit timers and one 32-bit timer ($GTCFR1[PCAS] = 1$, $GTCFR2[PCAS] = 0$ or $GTCFR1[PCAS] = 0$, $GTCFR2[PCAS] = 1$), or two 32-bit timers ($GTCFR1[PCAS] = 1$ and $GTCFR2[PCAS] = 1$).

If $GTCFR1[PCAS] = 1$ and/or $GTCFR2[SCAS] = 1$, the two 16-bit timers (timer 1 and timer 2 or timer 3 and timer 4) function as a 32-bit timer with a 32-bit GTRFR, GTCPR, and GTCNR. In this case, GTMDR1/GTMDR3 is ignored, and the modes and functions are defined using GTMDR2/GTMDR4 and GTCFR1/GTCFR2. The capture are controlled from TIN2, and the interrupts are generated from GTEVR2. When working in the pair-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with 32-bit bus cycles.

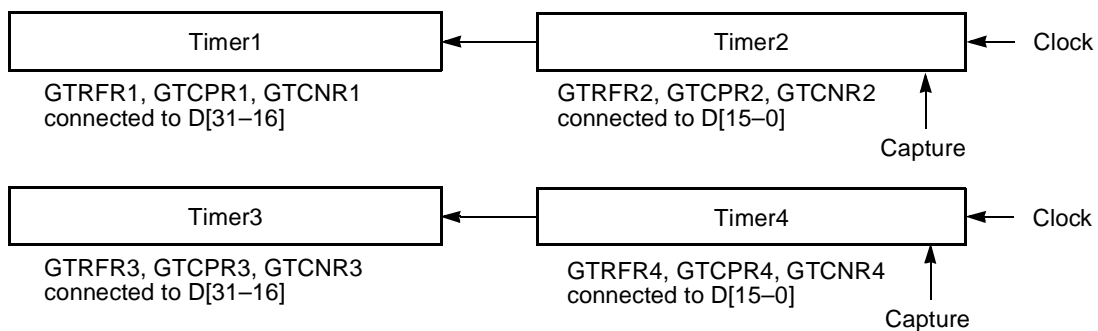


Figure 5-11. Timer Pair-Cascaded Mode Block Diagram

- Super-cascaded mode ($GTCFR2[SCAS] = 1$)
 In this mode, all four 16-bit timers can be internally cascaded to form a 64-bit counter, as shown in [Figure 5-12](#).
 If $GTCFR2[SCAS] = 1$, the all four 16-bit timers function as a 64-bit timer with a cascaded 32-bit GTRFR, GTCPR, and GTCNR. In this case, registers GTMDR1, GTMDR2 GTMDR3 and GTCFR1 are ignored, and the modes and functions are defined using GTMDR4 and GTCFR2 only. The capture are controlled from TIN4, and the interrupts are generated from GTEVR4. When working in the super-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with two 32-bit bus cycles.

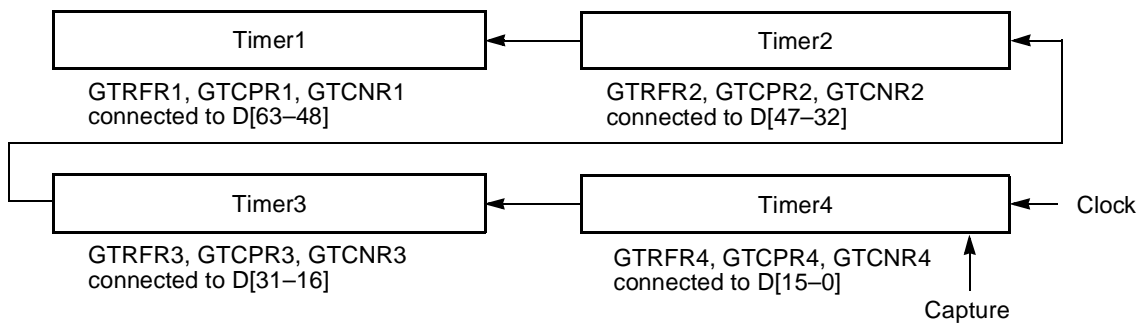


Figure 5-12. Timers Super-Cascaded Mode Block Diagram

5.8.7 Initialization/Application Information

5.8.7.1 Programming Guidelines

5.8.7.1.1 GTM Registers

The following initialization sequence of GTM is recommended:

- Write to $GTCFR_n$ in order to reset, to stop or to configure the appropriate timer's operation: cascaded timers configuration, gate mode configuration.
- Write to $GTPSR_n[PPS]$ fields in order to program the appropriate timer's clock primary prescaler.
- Write to $GTMDR_n$ in order to choose an input clock, to program the secondary prescaler and to set a desirable appropriate timer's operational mode.

NOTE

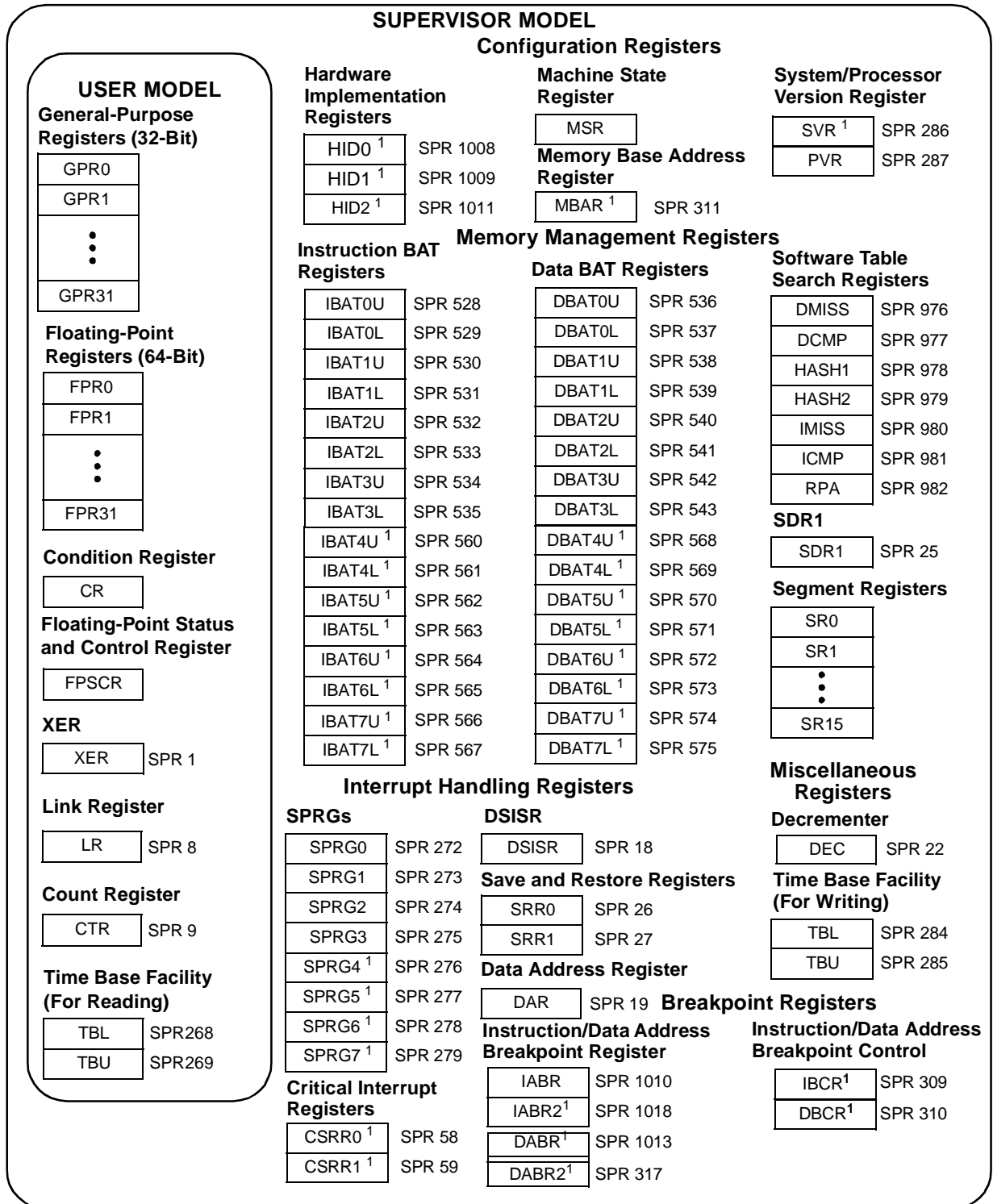
Erratic behavior may occur if $GTGCR$ and $GTPSR$ is not initialized before the $GTMDR$. Only $GTGCR[RST]$ can be modified at any time

- Clear $GTEVR_n[REF]$ and $GTEVR_n[CAP]$ by writing 1's in order to clear the previous events.
- Write to $GTRFR$ and to $GTCNR_n$ according to appropriate timer's $GTMDR_n$ programming.

NOTE

A write cycle to a $GTCNR_n[CNV]$ fields sets the register to the written value, causing its corresponding primary and secondary prescalers, ($GTPSR_n[PPS]$ and $GTMDR_n[SPS]$), to be reset.

- Write to $GTGCR_n[STP]$ and to $GTGCR_n[RST]$ in order to initialize the appropriate timer's operation.



¹ These registers are e300 core implementation-specific (not defined by the PowerPC architecture).

Figure 5-13. e300 Programming Model—Registers

- 7.3.1.3.3, 7-20 Added new table to show how HID0[ECLK] and HID0[SBCLK] are used to set the frequency of the *clk_out* signal.

Table 7-3. Using HID0[ECLK] and HID0[SBCLK] to Configure *clk_out*

<i>hreset</i>	ECLK	SBCLK	<i>clk_out</i>
Asserted	x	x	Bus clock (small pulse for every rising edge of sysclk)
Negated	0	0	Clock output off
	0	1	Core clock/2
	1	0	Core clock
	1	1	Bus clock

- 7.4, 7-38 Removed Table 7-8, “Difference Between e300 Cores”
- 8.5.16/8-25 In [Table 8-26](#), “SIFCR_H Field Descriptions” and [Table 8-27](#), “SIFCR_L Field Descriptions”, changed INT n bit description of “corresponds to an external interrupt source” to “corresponds to an internal interrupt source”.
- 9.1/9-1, 9.2.1/9-3 Modified text to clarify that statements about setting 8_BE when in 32-bit bus mode applies to DDR1 only.
- 9.3.2.1, 9-6 Updated description of MDQS[0:8] in [Table 9-3](#) to read as follows:
Data strobes. Inputs with read data, outputs with write data.
- 9.4/9-9 In [Table 9-5](#), “DDR Memory Controller Memory Map,” changed reset value for DDR_IP_REV2 to 0x00nn_00nn to defer implementation-specific values to register description in [Section 9.4.1.17](#), “DDR IP Block Revision 2 (DDR_IP_REV2).”
- 9.4.1.6, 9-19 Modified descriptions of TIMING_CFG_2 fields CPO and FOUR_ACT as follows:

Table 9-11. TIMING_CFG_2 Register Field Descriptions

Bits	Name	Description
4–8	CPO ¹	MCAS-to-preamble override. Defines the number of DRAM cycles between when a read is issued and when the corresponding DQS preamble is valid for the memory controller. For these decodings, “READ_LAT” is equal to the CAS latency plus the additive latency. 00000 READ_LAT + 1 01100 READ_LAT + 5/2 00001 Reserved 01101 READ_LAT + 11/4 00010 READ_LAT 01110 READ_LAT + 3 00011 READ_LAT + 1/4 01111 READ_LAT + 13/4 00100 READ_LAT + 1/2 10000 READ_LAT + 7/2 00101 READ_LAT + 3/4 10001 READ_LAT + 15/4 00110 READ_LAT + 1 10010 READ_LAT + 4 00111 READ_LAT + 5/4 10011 READ_LAT + 17/4 01000 READ_LAT + 3/2 10100 READ_LAT + 9/2 01001 READ_LAT + 7/4 10101 READ_LAT + 19/4 01010 READ_LAT + 2 10110–11111 Reserved 01011 READ_LAT + 9/4

Table 9-11. TIMING_CFG_2 Register Field Descriptions (continued)

Bits	Name	Description
26–31	FOUR_ACT	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 cycle 000010 2 cycles 000011 3 cycles 000100 4 cycles ... 010011 19 cycles 010100 20 cycles 010101–111111 Reserved

¹ For CPO decodings other than 00000 and 11111, 'READ_LAT' is rounded up to the next integer value

- 9.4.1.7/9-20 In [Table 9-12](#), “DDR_SDRAM_CFG Field Descriptions,” added notes to DDR_SDRAM_CFG[RD_EN] and DDR_SDRAM_CFG[RD_EN] field descriptions stating that these fields must not both be set at the same time.
- 9.4.1.8, 9-23 Changed setting 01 of DDR_SDRAM_CONFIG_2[DQS_CFG] to reserved.
- 9.4.1.14/9-29 The MPC8360 does not support 1/8-clock adjust. Modified the end of the first sentence to read ‘...along with a 1/4-cycle clock adjustment.’
- 9.5.3/9-49 Added sentence to 6th bullet (mode register set):
For DDR2 in 32-bit bus mode, all 32-byte burst accesses from the platform are split into two 16-byte (that is, 4 beat) accesses to the SDRAMs in the memory controller.
- 9.6.1, 9-69 Changed DDR2 setting for DQS_CFG (in [Table 9-53](#)) to ‘Should be set to 00.’
- 10.3.1.3, 10-18 Updated MAR (memory address register) to show that field A is 32 bits wide (bits 0–31).
- 12.4.6/12-9 Changed IMISR access to mixed—two of the bits are w1c. Updated bits to reflect w1c access. Also updated memory map.
- 12.4.6/12-9,
12.4.7/12-11 Removed the following text from introductory paragraph of IMISR and IMIMR: IMISR should be accessed only from the CSB and only in agent mode. Accesses while in host mode or from the PCI bus have undefined results.
- 12.4.8.1/12-12 In [Table 12-11](#), “DMAMR_n Field Descriptions,” added the following to bit fields DAHE and SAHE:
The DMA does not support address hold when the external trigger mode is selected (EMSEN = 1).
- 12.4.8.1, 12-13 Added the following note to DMAMR_n[DAHE] and DMAMR_n[SAHE]:
Note that the address hold feature is not supported when external hardware control is used.

address may still be used once before the new start address takes effect. Therefore it is not advisable to change this address while a peripheral is enabled.

The new start address is programmed in the CECDR register, the relevant SNUM is programmed in CECR[7:14], and the scheduler request is enabled by setting CECR[17].

	0	1		6	7		14	15	
Field	0	—				SNUM			1
Reset	0000_0000_0000_0000								
R/W	R/W								
Addr	CE_base + 0x00100								
	16	17	18		25	26		31	
Field	—	—	—			001111			
Reset	0000_0000_0000_0000								
R/W	R/W								
Addr	CE_base + 00102								

Figure 20-3. CECR for PushSched Command

- 21.10.5.1/21-51 Modified last paragraph.
- 21.10.5.3/21-51 Made correction to Pair-cascade mode section, second paragraph.
- 23.3.1/23-5 In [Table 23-3](#), “UCC Registers,” removed UCC Data Synchronization Register row.
- 23.3.3/23-7 Changed ‘Transmit Polling Timer’ to ‘UCC Transmit Polling Timer’ and ‘UFTP’ to ‘UTPT’.
- 24.2.4/24-7 Changed ‘UFTP’ to ‘UTPT’.
- 24.4.6.3/24-14 After step 1, add, ‘If a protocol switch is performed from fast protocol, follow step 2 in Section 24.4.6.5.’
- 24.4.6.5/24-14 Add the following as step 2. ‘If a protocol switch is performed from fast protocol: Issue the PushSched host command for UCC Transmitter (CECDR value 0x80), Issue the PushSched host command for UCC Receiver (CECDR value 0x82).’
- 27.4.1/27-4 In [Table 27-1](#), “UCC Memory Map (Fast Mode),” changed ‘UFTP’ to ‘UTPT’.
- 27.4.3/27-10 Changed ‘UFTP’ to ‘UTPT’.
- 29.3.3/29-17 Modified third bullet under ‘The collision-detection mechanism supports only:’ to the following:
Open-drain connection via port pin configuration or via external transceivers.
- 30.5.1.1/30-29 In [Table 30-5](#), “UPSMR Ethernet Field Descriptions,” made change to UPSMR[AUFC] bit field description.
- 30.5.1.17/30-47 Added description of UEMPR[Extension Header]:
Can contain any user-defined data. Note that this does not extend the MAC

- parameter value, which would cause more delay when a flow control frame is detected.
- 30.5.2.2/30-56 In [Table 30-27](#), Offset +0, bit 15, change, ‘if UPSMR[IPCHK]=1,’ should be ‘if REMODER[IPCHK]=1.’
- 30.6.2.6.5/30-91 In [Table 30-56](#), “TAD Field Descriptions,” added note to Rej field description: For proper operation, if TAD[Rej]=1 the user must program bits TAD[VPriority,CFI,VID]=0.
- 30.7.5/30-97 In [Table 30-59](#), “UCC Statistics,” added the following sentence to the field descriptions for RBCA, RMCA, TMCA, and TBCA counters: Frames longer than 1518 bytes (if untagged) or 1522 bytes (if tagged) are not counted.
- 30.8/30-110 In [Table 30-79](#), removed ‘or transmitter error (underrun, retransmission limit reached, or late collision)’ from RESTART TRANSMIT description.
- 31.11.3/31-32 In [Table 31-25](#), “1588 RTC External Signals,” added parallel port signal pins columns as shown below:

[Table 31-25](#) describes the IEEE 1588 Real Time Clock external signals:

Table 31-25. 1588 RTC External Signals

Signal name	I/O	Description	Parallel Port Signal Pins	
			Primary Option	Secondary Option
PTP_PPS1	Output	Pulse per second output signal generated by configuring the FIPER1 register. Each time the FIPER1 value is expired one RTC clock period pulse is generated	CE_PC21	CE_PE20
PTP_PPS2	Output	Pulse per second output signal generated by configuring the FIPER2 register. Each time the FIPER2 value is expired one RTC clock period pulse is generated	CE_PC17	CE_PD19
PTP_PPS3	Output	Pulse per second output signal generated by configuring the FIPER3 register. Each time the FIPER3 value is expired one RTC clock period pulse is generated	CE_PC5	CE_PF29
PTP_ALARM1	Output	Alarm output triggers, set if the timer value reaches the alarm1 register	CE_PC0	CE_PE14
PTP_ALARM2	Output	Alarm output triggers, set if the timer value reaches the alarm2 register	CE_PC23	CE_PE6
PTP_REF_CLK	Output	Divided output clock, by configuring the TMR_PRSC register	CE_PC12	CE_PE26
PTP_CLK	Input	External oscillator Real Time Clock	CE_PC30	

Table 31-25. 1588 RTC External Signals (continued)

Signal name	I/O	Description	Parallel Port Signal Pins	
			Primary Option	Secondary Option
PTP_EXT_TRIG1	Input	Input trigger to capture timestamps. The captured timestamp is stored in TMR_ETSS1L/TMR_ETSS1H	CE_PB4	CE_PC14
PTP_EXT_TRIG2	Input	Input trigger to capture timestamps. The captured timestamp is stored in TMR_ETSS2L/TMR_ETSS2H	CE_PC11	CE_PC27

36.6.4/36-18

Add the following table as Table 36-9:

Table 36-9. SI Mode Register Description

Bits	Name	Description
0-3	SADx	<p>Starting address for the RAM of TDMx. These four bits define the starting address of the SI RAM section that belongs to TDMx channel. The last entry of a certain TDM is determined by the LST bit in the SI RAM entry. The user must set LST within the entries of SI RAM blocks for every TDM used i.e. before the starting address of the next TDM.</p> <p>0000 Entries 0-31, bank0 0001 Entries 32-63, bank0 0010 Entries 64-95, bank1 0011 Entries 96-127, bank1 0100 Entries 128-159, bank2 0101 Entries 160-191, bank2 0110 Entries 192-223, bank3 0111 Entries 224-255, bank3 1000 Entries 256-287, bank4 1001 Entries 288-319, bank4 1010 Entries 320-351, bank5 1011 Entries 352-383, bank5 1100 Entries 384-415, bank6 1101 Entries 416-447, bank6 1110 Entries 448-479, bank7 1111 Entries 480-511, bank7</p> <p>Note: Each bank should be addressed by a single TDM only</p>
4-5	SDMx	<p>SI Diagnostic Mode for all eight TDMs</p> <p>00 normal operation. 01 Automatic echo. In this mode, the channel_x transmitter automatically retransmits the TDM received data on a bit-by-bit basis. The receive section operates normally, but the transmit section can only retransmit received data. In this mode, the L1GRx line is ignored. 10 Internal loopback. In this mode, the TDM transmitter output is internally connected to the TDM receiver input (L1TXDx is connected to L1RXDx). The receiver and transmitter operate normally. The data appears on the L1TXDx pin and in this mode, the L1RQx line is asserted normally. The L1GRx line is ignored. 11 Loopback control. In this mode, the TDM transmitter output is internally connected to the TDM receiver input (L1TXDx is connected to L1RXDx). The transmitter output (L1TXDx) and the L1RQx pin is inactive. This mode is used to accomplish loopback testing of the entire TDM without affecting the external serial lines.</p>

Tablen 36-9. SI Mode Register Description (continued)

Bits	Name	Description
6–7	RFSDx.	Receive frame sync delay for all eight TDM. Determines the number of clock delays between the receive sync and the first bit of the receive frame. Even if CRTx is set, these bits do not control the delay for the transmit frame. 00 No bit delay. The first bit of the frame is transmitted/received on the same clock as the sync. 01 1-bit delay. Use for IDL. 10 2-bit delay 01 3-bit delay
8	Reserved	Reserved. Should be cleared.
9	CRTx	Common receive and transmit pins for all TDM. Useful when the transmit and receive sections of a given TDM use the same clock and sync signals. In this mode, L1TCLKx and L1TSYNCx pins can be used as general-purpose I/O pins. 0 Separate pins. The receive section of this TDM uses L1RCLKx and L1RSYNCx pins for framing and the transmit section uses L1TCLKx and L1TSYNCx for framing. 1 Common pins. The receive and transmit sections of this TDM use L1RCLKx as clock pin of channel x and L1RSYNCx as the receive and transmit sync pin. Use for IDL. RFSD and TFSD are independent of one another in this mode.
10	SLx	Sync level for all TDM's. 0 The L1RSYNCx and L1TSYNCx signals are active on logic "1". 1 The L1RSYNCx and L1TSYNCx signals are active on logic "0".
11	CEx	Clock edge for all TDM's. 0 The data is transmitted on the rising edge of the clock and received on the falling edge (use for IDL). 1 The data is transmitted on the falling edge of the clock and received on the rising edge
12	FEx	Frame sync edge for all TDM's. Determines whether L1RSYNCx and L1TSYNCx pulses are sampled with the falling/rising edge of the channel clock. 0 Falling edge. Use for IDL. 1 Rising edge.
13	GMx	Grant mode for all TDM's. 0 No grant mode is supported. 1 IDL mode. A GRANT mechanism is supported if the corresponding CMXSCR[GRx] is set. The grant is a sample of L1GRx while L1TSYNCx is asserted. This grant mechanism implies the IDL access controls for transmission on the D channel.
14–15	TFSDx	Transmit frame sync delay for all TDM's. Determines the number of clock delays between the transmit sync and the first bit of the transmit frame. 00 No bit delay. The first bit of the frame is transmitted/received on the same clock as the sync. 01 1-bit delay 10 2-bit delay 11 3-bit delay

- 32.3.8.1/36-117 Added explanation of how to add multicast members to hierarchical scheduling and the special host command for doing so.
Updated the V-TCT table as well.
- 33.6.5.1/36-29 [Table 33-13](#), “UPDCx in ATM Protocol Field Descriptions,” added reserved row for bits 22-23.
- 36.1/36-1 In [Figure 36-1](#), “SI Block Diagram,” modified note at bottom of figure.

Revision History

37.19.7/37-74	In Table 37-36 and Table 37-33 , updated MMR by adding Not_Active bit to indicate to the MUX process that this queue is not active. Add missing GBL bit in MPT.
38.3.1.1/38-15	Change Figure 38-9 , bit 14 from Reserved to RAD.
38.3.1.1/38-15	Change Table 38-2 , bit 14 from Reserved to RAD with this bit description: Receive ATM Disable 0 Pass cells, except filtered cells, to the ATM or IMA microcode. 1 No cells are passed to the ATM or IMA microcode. The MTC will maintain the cell delineation algorithm and counters will be updated.
42.3.2.8/42-6	Bolded user initialized parameters in Table 42-1 , “Global Multichannel Parameters”: MCBASE, QMCSTATE, MRBLR, Tx_S_PTR, RxPTR, GRFTHR, CRFCNT, INTBASE, INTPTR, Rx_S_PTR, TxPTR, C_MASK32, TSATRx, TSATTx, C_MASK16, and QMC_Global_Channel_specific_base.
42.3.4.1/42-16	Bolded user initialized parameters in Table 42-4 , “Channel-Specific HDLC Parameters,”: TBASE, CHAMR, TSTATE, TBPTR, ZISTATE, INTMSK, RBASE, MFLR, RSTATE, RBPTR, and ZDSTATE.
42.3.4.2/42-20	Bolded user initialized parameters in Table 42-8 , “Channel-Specific Transparent Parameters,”: TBASE, CHAMR, TSTATE, TBPTR, ZISTATE, INTMSK, RBASE, TMBLR, RSTATE, RBPTR, ZDSTATE, and TRNSYNC.
43.5.3.5/43-52	Change step 8 to read as follows: Enable the corresponding link/PHY at the TC layer for example if using SAM as the TC layer set MTC_MODE[RXEN], or clear MTC_MODE[RAD] if the MTC is already in SYNC state but not passing cells up. If an external TC layer is used, set the corresponding enable bit for the external TC layer device.
43.5.3.5/43-52	Revised step 1 of the Rx steps for LDS section to clarify that after GDS failure ADD_NEW and ADD_NEW_M are correctly configured if the link that caused the failure is to be added after it reaches the working state.
43.5.3.10/43-58	Updated section to clarify that IFSD and GDS events do not require link removals.
43.5.3.10/43-58	Updated DCB Synchronization Lost (DSL) bulleted section to explain how to perform the fast recovery.
43.5.3.6.1/43-55	Change step 8 to read as follows: Inhibit reception of cells over the dropped link by disabling the TC layer such that it does not pass any more cells to the IMA layer. Disable the TC layer, for example, if using the SAM clear MTC_MODE[RXEN]. If the TC layer is connected on the UTOPIA bus, disable it by ensuring that it passes no more cells onto the UTOPIA bus for this link. If the application requires that the TC layer remains cell delineated (no LOCD) at this step then software should adhere to the following steps: a) Leave MTC_MODE[RXEN] = 1 b) Set MTC_MODE[RAD]

B.3 Changes From Revision 0 to Revision 1

Section, Page No.	Changes
1.2.1, 1-8	<p>Replace the first sentence of the second paragraph on page 1-8 with the following: The core is a superscalar processor that can issue three instructions (two plus a branch) and completes and retires as many as two instructions per clock cycle.</p> <p>Replace the fourth sentence of the fourth paragraph on page 1-8 with the following: The caches use a pseudo least recently used (PLRU) replacement algorithm; the TLBs use a least recently used (LRU) replacement algorithm.</p>
2-3, 2-2	<p>Add the following text to the beginning of this section: Reading from address locations which appear as reserved in the memory map table is not guaranteed to return predictable data. Writing to address locations which appear as reserved in the memory map table is not allowed and could lead to unpredictable behavior of the device. Reserved bits in non-reserved registers will be read as zero unless the reset value of those bits is different due to internal logic considerations.</p> <p>When writing to registers with reserved bits, those reserved bits should be cleared. By doing so, existing software would be able to run on a future modified device in which some reserved bits were allocated for enhanced modes. This would allow for maintaining the legacy functionality when set to zero.</p> <p>In certain specific cases, reserved bits should not be cleared but should keep their reset value. Thus, the software should perform a ‘read-modify-write’ and make sure that it does not change the reset value of those bits. The description of the specific bits will indicate when this is needed.</p>
2.3, 2-5	<p>In Table 2-2, the reset value of SWCRR (system watchdog control register) should appear as 0xFFFF_0007/0xFFFF_0003 (either value possible). A footnote was added to explain as follows: SWCRR[SWEN] reset value directly depends on RCWHR[SWEN] (reset configuration word high).</p>
2.4, 2-35	<p>In Table 2-4, the size of IPGIFG (interframe gap register) should be 4 bytes.</p>
2.4, 2-36	<p>In Table 2-4, the section “Ethernet Statistics Counters” was modified.</p>
3.4.8, 3-28	<p>Added POS I/O signals to Table 3-11 through Table 3-17.</p>
3.4.8, 3-34	<p>In Table 3-12, the input row for PB4 (CPPARBx[SELn]=11) should contain: UPC2:RxAddr[0] UTOPIA slave UPC2:RADR[0] POS slave (Secondary Option)</p>
3.4.8, 3-55	<p>In Table 3-17, the input row for PG3 (CPPARGx[SELn]=01) should contain:</p>

	UPC2:RxAddr[0] UTOPIA slave UPC2:RADR[0] POS slave (Primary Option)
4.1.2, 4-3	Add the following note to the descriptions of both CLKIN and PCI_CLK/PCI_SYNC_IN: NOTE: If PCI is not used, the device clock source must be connected to PCI_CLK/PCI_SYNC_IN, not CLKIN.
4.3.1.1, 4-10	In the meaning of CFG_RESET_SOURCE[0:2] (Table 4-5) for option 010, replace the last sentence with the following: Reset configuration word is loaded from an I ² C EEPROM. PCI_CLK/PCI_SYNC_IN is valid for any PCI frequency up to 66.666 MHz (range of 24–66.666 MHz).
4.3.3.1.1, 4-23	Add the following paragraph after the first paragraph: In these figures, CLOCK/32 is an internal clock. The LCLK _n signals are not active during the power-on reset sequence.
5.2.4.1.1, 5-6	Add the following bullet below the existing one: <ul style="list-style-type: none">• When the e300 core is writing to IMMRBAR, it should use the following sequence:<ul style="list-style-type: none">– Read the current value of IMMRBAR using a load word instruction followed by an isync. This forces all accesses to configuration space to complete.– Write the new value to IMMRBAR.– Perform a load of an address that does not access configuration space or the on-chip SRAM, but has an address mapping already in effect (for example, boot ROM). Follow this load with an isync.– Read the contents of IMMRBAR from its new location, followed by another isync.
5.4.1.1, 5-28	Add the following note to the description of <u>PCI_MODE</u> : NOTE: If PCI is not used, the device clock source must be connected to PCI_CLK/PCI_SYNC_IN, not CLKIN.
5.5.4, 5-41	In Table 5-46, the reset value of SWCRR should appear as 0xFFFF_0007/0xFFFF_0003 (either value possible). A footnote was added to explain as follows: SWCRR[SWEN] reset value directly depends on RCWHR[SWEN] (reset configuration word high).
5.5.4.1, 5-42	Added text to description of SWCRR[SWEN]: The reset value directly depends on the value of RCWHR[SWEN] bit.
8.5.19, 8-27	The offset of SCVCR should be 0x60.
8.5.20, 8-28	The offset of SMVCR should be 0x64.
9.4.1.16, 9-30	Remove this section.

9.4.1.30, 9-38	Remove this section.
14.6.1.4, 14-91	<p>Replace paragraph three with the following:</p> <p>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 will be read to launch the next descriptor. Writing a descriptor pointer to the fetch FIFO while the FIFO is full will result in a single overflow interrupt to advise the user that the descriptor pointer was not successfully written to the fetch FIFO. The channel will continue processing and software can check the fetch FIFO counter in the crypto-channel pointer status register before attempting to re-enqueue the descriptor pointer. If a second descriptor pointer is written to the fetch FIFO before the single the single overflow error is cleared, the channel will generate a double overflow error interrupt and stop processing descriptors. The channel can be restarted by setting the Continue bit in the crypto-channel configuration register, or completely reset by writing the Reset bit in the same register.</p>
19.1.8.9, 19-14	<p>In Table 19-9, the description of BA should read as follows:</p> <p>Temporary buffer base address in multi-user RAM. The address must be 4KB aligned.</p>
21.4, 21-7	<p>Add the following sentence to footnote 2 of Table 21-1:</p> <p>The UCC1 Tx clock is also the 125MHz reference clock for Gigabit Ethernet when the UCC is configured for GMII/RGMII/TBI/RTBI.</p> <p>Add the following sentence to footnote 4 of Table 21-1:</p> <p>The UCC2 Tx clock is also the 125MHz reference clock for Gigabit Ethernet when the UCC is configured for GMII/RGMII/TBI/RTBI.</p>
27.4.2.1, 27-7	In Table 27-2, in the description of DIAG, remove the note “In Ethernet mode program MACCFG1[8] bit for loopback mode. DIAG bits have no effect..”
27.4.2.1, 27-10	<p>In Table 27-2, in the description of ENR, change the words (in the note) “set the MACCFG1 bit 2” to “set the MACCFG1 bit 29.”</p> <p>In the description of ENT, change the words (in the note) “set the MACCFG1 bit 0” to “set the MACCFG1 bit 31.”</p>
27.5.3, 27-14	The second sentence in the description of field RFET should begin “The recommended value for RFET (for most applications) is”
27.5.4, 27-15	The third sentence in the description of field RSFET should begin “The recommended value for RSFET”
29.3.3, 29-17	<p>Replace the last two sentences of the third paragraph with the following:</p> <p>If the echo bit is ever 0 when the transmit bit is 1, a collision occurs between terminals; the station(s) that sent a one stops transmitting. The station that sent a zero continues as normal.</p>
30.5.1.1, 30-29	<p>Replace the description of PTPE with the following:</p> <p>PTP Enable</p>

- 0 - Disable IEEE1588 assist connection to MAC I/F
1 - Enable IEEE1588 assist connection to MAC I/F
- 30.5.1.6, 30-38 Add the following sentence to the description of Minimum IFG Enforcement (Table 30-12):
Zero is not a legal value.
- 30.5.1.8, 30-41 Replace the description of Mgmt Clock Select in Table 30-14 with the following (note that the bit settings remain unchanged):
This field determines the clock frequency of the Mgmt Clock (CE_MDC). Its default value is 111. The source clock frequency is equal to the QUICC Engine clock divided by 16.
- 30.5.1.10, 30-42 In Table 30-16, “MIIMADD Field Descriptions,” replace the text in PHY Address description with the following:
This field represents the 5-bit PHY address field of Mgmt cycles. Up to 31 PHYs can be addressed, with one address reserved for internal TBI registers (corresponding to the value of TBIPA[TBIPA] whose default value is 0x00). Re-programming of TBIPA[TBIPA] to a non-zero value allows PHY address to then be set to 0.
- 30.5.3.3.2, 30-63 Replace the second row of Table 30-33 (reserved area) with the following:

Offset	Bits	Name	Description (Data structure must be 32 bytes aligned).	Initialized by
0x04-0x07		Reserved	Set to zero	QE
0x08-0x09	0:31	BDRConsumerAddress	Points to the next entry in the BD ring to be executed. This field must be initialized after reset to the first entry in the BD ring.	QE
0x0A - 0x0B		Reserved		QE

- 30.5.3.3.3, 30-64 In Table 30-34, add the following reference to the descriptions of NorTSRByteTime (offset 0x50) and FracSiz (offset 0x52):
Also see example calculations in Section 30.15, “Traffic Shaper Programming Considerations.”
- 30.5.3.5, 30-68 In Table 30-36, add the following to the description of LossLessFCPtr:
This base address must be programmed if Lossless Flow Control is enabled in REMODER[LossLessFCEn] OR if Advanced Queue Management is enabled by programming TAD[IWCT Index]>0.
If Lossless Flow Control feature is enabled, the user must allocate 8..64 bytes for this data structure depending on the number of queues.

30.6.2.6.5, 30-90 Replace the existing Table 30-56 with the following:

Table 30-56. TAD Field Descriptions

Offset	Bits	Name	Description
0x0	0-15		See description in REMODER register. Section 30.5.3.7, "Rx Ethernet Mode Register (REMODER)
0x2	0-15	QTag	QTag (VPriority,CFI,VID) to be inserted.
0x4		Reserved	Set to zero

- 30.7.5, 30-97 Table 30-59 was updated extensively.
- 30.7.7, 30-99 Register names were modified throughout this section.
- 30.7.7.16, 30-107 The bits of RxDiscOv should be numbered from 16–31.
- 30.7.7.18, 30-109 Bits 0–15 of CMR are reserved; the non-reserved fields are in bits 16–31. (MT64 is bit 16, MT65 is bit 17, MT128 is bit 18, ...)
- 30.15.1.8, 30-163 In Table 30-124, the traffic rate of queue number 5 should be 1/10
- 31.9.1, 31-17 In Table 31-12, the names of the fields ALM2P and ALM1P should be swapped to match the register figure. In the description of setting 11 for CKSEL, change “reserved by QUICC Engine” to “NA for QUICC Engine.”
- 32.3.8, 32-118 Remove the fourth sentence of the second paragraph on this page (stating that the table is 64-byte aligned).
- 33.4.1.1, 33-19 Table 33-3 was revised to include pull up and tri-state requirements.
- 33.4.1.2, 33-20 Table 33-4 was revised to include tri-state requirements.
- 33.4.1.3, 33-21 Table 33-5 was revised to include pull up and tri-state requirements.
- 33.4.1.4, 33-22 Table 33-6 was revised to include tri-state requirements.
- 33.6.1, 33-26 Change footnote 1 of Table 33-8 to read as follows:
 Note on backwards compatibility: the programing model regarding the LPB mode have changed, instead of programing the GUMR[DIAG] as in the 8260, it is done in through UPGCR[DIAG].
- 43.5.3.6.1, 43-55 In the TX Link Removal, updated step 8 to accomodate those cases where the TC layer must remain in SYNC state in order to send the correct state of link back to the FE.